For my mother, the memory of my father, and my dear friend Mary.

Learning as Knowledge Integration

by

Kenneth S Murray, B.S., M.S.C.S

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May, 1995

Acknowledgments

This research has benefited immeasurably from the support and contributions of many people. I am grateful to each of my committee members for their encouragement, enthusiasm, and patience. Foremost among them is Bruce Porter, my advisor, who suggested such a wonderful topic and who provided many technical insights as well as several years of financial support that allowed my dedicated pursuit of this research. Most importantly, I am grateful for the intuitions that I have developed under, and as a result of, his mentoring about essential issues in Artificial Intelligence and conducting research. I am indebted to Diane Schallert whose provocative insights helped to develop and clarify some of the most important intuitions on which this research is based, and whose encouragement was both a joy and an enduring source of motivation. I appreciate Ben Kuipers for suggesting what turned out to be a very interesting (and challenging!) extension to one of the implemented learning scenarios and for his enthusiasm for this research. I am indebted to Woody Bledsoe whose high standards, enlightened view of Artificial Intelligence, and support for exploratory research, coupled with his enthusiasm and encouragement for this research, provided an important and enduring source of motivation. I also appreciate the comments of Ray Mooney and Robert Holte on this research and for their significant contributions to Machine Learning. The contributions of these mentors have been invaluable and have made this research possible. Their enthusiasm and support for this research was a significant motivation for me to persevere through the difficult times.

I am very grateful to Doug Lenat for providing an exceptional research environment as well as much support, encouragement and inspiration. This research is just one example of the many research opportunities made possible by his fundamental contributions to the pursuit of artificial intelligence.

Discussions with many current and former colleagues have helped shape this work, including Liane Acker, Ray Bareiss, Brad Blumenthal, Karl Branting, James Crawford, Mark Derthick, Adam Farquhar, R.V. Guha, James Lester, Jeff Rickel, Wei-Min Shen, and Loren Tervene. I am indebted to Liane Acker, Lila Ghemri, Mary Janecek, Bill Jarrold, Kathy Mitchell, Karen Pittman, Jeff Rickel, and Jamie Stephens for their generous participation in the imperical evaluation of this research. I am grateful to Liane Acker, Brad Blumenthal, Tom Jones, James Lester, Karen Pittman, and Art Souther for their contributions to creating the Botany Knowledge Base.

I am grateful to Chris Merritt for her prolific suggestions on improving the clarity and simplicity of the writing. Paul Blair, Mark Derthick, Mary Janecek, Kathy Mitchell, Deborah Nichols, and Laurie Sines have also been exceptionally generous with their time in reading drafts of this dissertation and offering suggestions, and each has contributed significantly to improving the quality of my presentation of this research.

I will be forever indebted to my parents for their generosity, encouragement, and the virtues of their values. I am grateful to my father for bestowing such a high regard for the pursuit of knowledge and tenacity. I am grateful to my mother for her uncompromised love and encouragement. I will always be grateful to Mary Janecek whose unwavering support and pure empathy have been invaluable; I'll never know a better partner. I will also always be grate-

ful to Laurie Sines, whose care and friendship have been among the greatest joys in my life. I deeply appreciate the following friends for their companionship and endless encouragement: Mark Derthick, Melanie McNeely, Joanna McNeely, Laura Maurer, Kay Plavidal, Deborah Nichols, Kathy Mitchell, Bill Jarrold, Venu Rao, Sandra Henry, Steve Davidsen, Matt Kaufmann, James Lester, Kathy Mitchell, Kelly, Muffet, Jack, Cloe, and the Thunks. With the care of such friends, the joy of life never wanes too much!

I am grateful to MCC for providing facilities and support for this research and to many of its employees whose care, advice, and talents benefited me over the years, including Judy Bowman, Mary Shepherd, David Gadbois, and Mark Caffey. I thank the members of the Cyc staff, both current and past. This very talented group has, over the span of many years, aided my efforts tremendously with their generosity, encouragement, assistance, and kindness.

Gloria Ramirez is an angel!

This research was funded by the Air Force Human Resources Laboratory under grant BAA-90-04 and RICIS grant ET.14, by the Army Research Office under grant ARO-DAAG29-84-K0060, and by the National Science Foundation under grant IRI-8620052.

Thanks to one and all!

Kenneth S Murray

The University of Texas at Austin

May, 1995

Learning as Knowledge Integration

Publication No. _____

Kenneth S Murray, Ph.D.
The University of Texas at Austin, 1995

Supervising Professor: Bruce Porter

A fundamental challenge for Artificial Intelligence is developing methods to build and maintain knowledge-based systems. *Knowledge integration* is the task of identifying how new and prior knowledge interact while incorporating new information into a knowledge base. This task is pervasive because substantial knowledge bases must be developed incrementally: segments of knowledge are added separately to a growing body of knowledge. This task is difficult because new and prior knowledge may interact in very subtle and surprising ways, and unanticipated interactions may require changes to the knowledge base. Performing knowledge integration involves determining and effecting these changes. This research investigates knowledge integration as a machine learning task. Its contributions include formalizing knowledge integration as a machine learning task, developing a computational model for performing knowledge integration, and instantiating the computational model as an implemented machine learning program.

The study of knowledge integration and methods that perform it is important both for pragmatic concerns of building knowledge-based systems and for theoretical concerns of understanding learning systems. By identifying subtle conflicts and gaps in knowledge, knowledge integration facilitates building knowledge-based systems. By avoiding unnecessary restrictions on learning situations, knowledge integration reveals important sources of learning bias and permits learning behaviors that are more opportunistic than do traditional machine learning tasks.

REACT is a computational model that identifies three essential activities for performing knowledge integration. *Elaboration* assesses how new and prior knowledge interact. The system's limited capacity to explore the interactions of new and prior knowledge requires methods to focus its attention. This focus is achieved by restricting elaboration to consider only selected segments of prior knowledge. *Recognition* selects the prior knowledge that is considered during elaboration. By identifying the consequences of new information for relevant prior knowledge, recognition and elaboration reveal learning opportunities, such as inconsistencies and gaps in the extended knowledge base. *Adaptation* exploits these learning opportunities by modifying the new or prior knowledge.

KI is a machine learning program that implements the REACT model. Empirical studies demonstrate that KI provides significant assistance to knowledge engineers while integrating new information into a large knowledge base.

Table of Contents

Acknowledgments iii

Table of Contents viii

1. Introduction 1

1.1 What is knowledge integration? 1

1.2 Why study knowledge integration? 3

1.2.1 Facilitating the construction of knowledge-based systems 4

1.2.2 Enhancing learning opportunism 9

1.3 Dissertation organization 15

2. REACT: a Computational Model for Knowledge Integration 17

2.1 Knowledge Integration as Human Learning 17

2.1.1 Learning actively relates new and prior knowledge 18

2.1.2 Limited resources force selective learning 23

2.2 REACT 24

2.2.1 Comprehension: relating new and prior knowledge 26

2.2.2 Adaptation – detecting and exploiting learning opportunities: 36

2.2.3 Discussion 38

2.3 Summary 42

3. KI: A Tool for Knowledge Integration 44

3.1 Overview 44

3.2 The Botany Knowledge Base 48
3.2.1 Representing concepts 48
3.2.2 Ontology 49
3.2.3 Inference 51
3.3 Interpreting new information 53
3.4 Elaboration: determining the consequences of new information . 56
3.4.1 Initializing the learning context 56
3.4.2 Extending the learning context via inference 58
3.4.3 Distinguishing consequences of new information 61
3.4.4 Maintaining justifications of inferred facts 62
3.5 Adaptation: detecting and exploiting learning opportunities . . 65
3.5.1 Resolving inconsistencies 67
3.5.2 Inference-path compilation 73
3.5.3 Abduction: acquiring rules to explain new beliefs 79
3.6 Discussion 80

4. Recognition: Focusing Attention with Views 82

4.1 The problem of focusing attention 82
4.2 The approach: determining *what* to reason about 84
4.3 A model of relevance 85
4.4 Views: contexts of mutually relevant beliefs 88
4.4.1 Creating views with view types 91
4.4.2 Selecting views 102
4.5 Discussion 114
4.5.1 Why views work 114
4.5.2 Views qua learning bias 116

4.5.3 Views qua schemas 117

5. Identifying Deep Consequences of New Information 121

5.1 Consequences for the leaf epidermis as a container 122

5.2 Consequences for the leaf's use of carbon dioxide 126

5.3 Resolving anomalies 129

5.3.1 Identifying knowledge-base revisions to resolve anomalies 132

5.3.2 Propagating the revision throughout the learning context 137

5.3.3 Propagating the revision through the adaptation methods 140

5.4 Teleological learning 141

5.4.1 Representing goals 142

5.4.2 Identifying teleological explanations 143

5.4.3 Generalizing teleological explanations 145

5.5 Knowledge deepening: explaining "shallow" rules 156

5.5.1 Identifying physiological functions 158

5.5.2 Identifying proofs of rules 161

6. An Empirical Analysis of KI 166

6.1 The knowledge base 168

6.2 Analyzing elaboration 168

6.2.1 The productivity of elaboration 168

6.2.2 The explanation level 171

6.3 Analyzing recognition 173

6.3.1 The coverage provided by view types 173

6.3.2 Sensitivity of view selection 174

6.3.3 Utility for selecting relevant background knowledge . . . 174

6.4 Analyzing Adaptation 179
6.4.1 Diversity of learning behaviors 179
6.4.2 Utility of conflict-resolution hierarchies 180
6.4.3 Measuring learning gain 181
6.4.4 Measuring learning utility 183

7. Related Work **187**

7.1 Belief Revision 187
7.1.1 Belief revision and belief sets: The AGM model 188
7.1.2 Belief revision and belief bases 192
7.1.3 Belief revision and coherence 193
7.2 Elaboration 195
7.2.1 FIE: Integrating propositions into a theorem prover . . . 195
7.2.2 Inference in an implementation of AGM belief revision . 196
7.3 Recognition 197
7.3.1 Focus spaces 198
7.3.2 Perspective hierarchies 200
7.3.3 Romper 201
7.3.4 View dimensions 204
7.3.5 View Retriever 205
7.3.6 Compositional modeling 208
7.3.7 Tripel 210
7.3.8 Composite model fragments 212
7.4 Adaptation 213
7.4.1 Explanation-based learning 213
7.4.2 Theory refinement and inductive-logic programming . . . 214

7.4.3 Knowledge acquisition 220
7.4.4 Teleological learning 224

8. Conclusions **229**

8.1 Summary 229
8.2 Research contributions 231
8.3 Agenda for further research 234
8.3.1 Integrating top-down and bottom-up control 235
8.3.2 Next generation views and view types 236
8.3.3 Evaluation via field tests 238

A.A Characterization of Learning **239**

A.1 The learning environment 239
A.1.1 The target domain 240
A.1.2 The knowledge source 241
A.1.3 The learning agent: a knowledge-based system 243
A.1.4 The task source 246
A.1.5 Knowledge transitions 251
A.1.6 Component interactions 257
A.1.7 Discussion 258
A.2 What is learning? 258
A.2.1 Learning as enhancing task performance 259
A.2.2 Learning as knowledge acquisition 262
A.2.3 Learning as response-function evolution through knowledge acquisition 263
A.2.4 Teaching vs. learning 266
A.3 Learning as test incorporation 268

B. The Learning Task of Knowledge Integration **272**

B.1 A formal definition of knowledge integration 272

B.2 General learning goals 275

B.2.1 Generic learning goals 276

B.2.2 Other learning goals 280

B.2.3 Discussion 283

C. Interpreting Semantic Networks **285**

C.1 The input language: specifying training with semantic networks 286

C.1.1 Logical networks 287

C.1.2 Disablement subgraphs 288

C.2 Parsing 289

C.3 Translating tuples 290

C.3.1 Identifying figurative references 293

C.3.2 Handling new constants 294

C.3.3 Establishing quantification 299

C.4 Adaptation: integrating translations 300

C.5 Summary 308

C.6 Lessons learned 309

D. A Model of Interestingness **311**

BIBLIOGRAPHY **314**

Vita

Chapter 1

Introduction

A fundamental challenge for Artificial Intelligence is developing knowledge-based systems, computer programs that exploit computational representations of knowledge in one or more domains. The task of integrating new knowledge into a body of existing knowledge is essential for developing knowledge-based systems. Substantial knowledge bases must be designed and constructed incrementally; discrete fragments of knowledge are identified, formalized, and then added to the growing knowledge base. Because new and existing knowledge may interact in surprising ways, unanticipated interactions may require changes to either the new or existing knowledge. Determining and affecting these changes are the goals of *knowledge integration.*

1.1 What is knowledge integration?

Performing knowledge integration involves identifying and evaluating the interaction between new and existing knowledge. The responsibility for performing knowledge integration is often placed on the teaching agent. For example, a knowledge engineer adding new information to a knowledge base must consider how new and existing knowledge will interact and then adapt them accordingly. In such cases knowledge integration is a teaching or authoring task. Alternatively, knowledge integration may be performed by the learning agent; in such cases it becomes a learning task. This research investi-

Teacher: The epidermis of the plant leaf is covered by the leaf cuticle, which is composed of cutin.

Student: Cutin is impermeable to gases, so the cuticle restricts water loss from the leaf.

Teacher: Yes, that's right.

Student: By reducing water loss, the cuticle helps the leaf avoid dehydration. Other plant organs that transpire would also benefit from a cuticle. Do stems, fruits, and flowers have cuticles?

Teacher: Yes.

Student: But the cuticle would also cause the leaf to starve.

Teacher: Explain.

Student: The cuticle is impermeable to gases. This prevents carbon dioxide in the atmosphere from passing through the leaf's epidermis. Without carbon dioxide, the leaf cannot conduct photosynthesis and starves.

Teacher: Well, the cuticle is impermeable to carbon dioxide; however, the leaf survives.

Student: Does the cuticle only partially cover the epidermis? Or, are there portals in the epidermis that permit restricted gas flow?

Teacher: Yes, the epidermis does have portals. They're called stomata.

Figure 1.1: Learning about leaf cuticle

gates knowledge integration as a machine learning task.

Knowledge integration occurs as a learning agent strives to comprehend new information. In Figure 1.1 a teacher presents a student with new information about the anatomy of a plant leaf. The student reacts to this new information, investigating its consequences and responding with several observations on the physiological effects of the leaf's cuticle covering the leaf's epidermis. The student thus acquires additional knowledge beyond the explicit content of the new information. For example, the student generalizes the new information: not only does the leaf have a cuticle, so do the other parts of the plant's shoot system. Furthermore, the student's responses reveal to the teacher the existing state of the student's knowledge, thus enabling the teacher to provide follow-up comments that resolve the student's questions and misconceptions.

1. learning identifies how new and prior knowledge interact
   (a) learning identifies how new and prior knowledge conflict
   (b) learning identifies how prior knowledge explains new information
2. learning acquires knowledge beyond the explicit content of new information
   (a) learning generalizes new information
   (b) learning resolves conflicts between new and prior knowledge
3. learning is reactive
   (a) learning reveals the state of the learner's knowledge
   (b) learning solicits additional information
4. learning is opportunistic
   (a) the learner need not anticipate the content of new information
   (b) the learner need not anticipate the precise uses of acquired knowledge

A summary of important aspects of the learning scenario illustrated in Figure 1.1.

Figure 1.2: Aspects of Learning as Knowledge Integration

Figure 1.2 summarizes some of the significant aspects of the learning scenario illustrated in Figure 1.1. While none of these aspects are strictly necessary for learning, each can be beneficial, as the following sections discuss. A primary goal of this research is to develop a computational model that exhibits the aspects of learning presented in Figure 1.2.

1.2 Why study knowledge integration?

There are both pragmatic and theoretical goals to the study of knowledge integration and the methods of performing it. Performing knowledge integration addresses critical issues that arise during the development of knowledge bases. Furthermore, knowledge integration builds on existing approaches to machine learning in order to apply in a wider variety of learning situations.

1.2.1 Facilitating the construction of knowledge-based systems

One of the fundamental goals of machine learning is facilitating the construction and maintenance of knowledge-based systems [Sim83]. This section discusses the significance of knowledge integration for two pervasive problems confronting developers of knowledge-based systems.

The interdependence of knowledge: Knowledge bases are built by incrementally adding or modifying discrete fragments of knowledge. As each fragment is added or changed, the correctness and utility of other fragments of knowledge becomes questionable. This was noted by Hayes [Hay85]:

> Since at any intermediate stage of theory construction there will be tokens not yet axiomatized, the process of formalizing those concepts may force changes in their correspondence to intuition and these changes might require our earlier partial theories to be rewritten.

Because this problem bears such close relation to the frame problem, introduced by McCarthy and Hayes [MH69] and extensively discussed in Knowledge Representation literature, it will be refered to as the *learning frame problem.*

The traditional frame problem is pervasive during reasoning because any change to the state of a represented world calls into question the accuracy of every statement describing that world. Solving the frame problem requires determining how those changes affect previous statements. The learning frame problem is pervasive during knowledge-base development because any modification (e.g., the addition of new information that extends the ontology) of

a knowledge base calls into question the correctness and utility of every existing statement in the knowledge base. Solving the learning frame problem requires determining how those modifications affect the truth of existing beliefs or the utility of existing concepts in the ontology. In other words, adding new information to an existing body of knowledge requires determining how the new and existing knowledge interact. Both frame problems are difficult for the same reason: the inferential path by which a change affects other beliefs can be arbitrarily long, so determining which beliefs are affected by a change can be arbitrarily difficult.

It is important to assess how new information interacts with existing knowledge because knowledge-base modifications intended to correct shortcomings may conflict with existing knowledge and introduce problems. Identifying such implicit conflicts is the necessary first step in resolving them; resolving each conflict may suggest additional knowledge not explicitly contained in the new information. Three common types of implicit conflict that occur during learning are:

1. New information introduces competing problem-solving objectives. For example, extending the drug therapy advisor MYCIN to minimize the number of drugs prescribed to each patient conflicts with other therapy goals, such as maximizing the number of symptoms covered by the prescribed treatment [MS86].

2. New information violates tacit simplifying domain assumptions. For example, the naive physics included in a botanical knowledge base may hold that some botanical objects (e.g., leaves, fruit) drop to the ground when they become disconnected (e.g., through abscission). This rule reflects

many implicit assumptions: that the objects are located in the earth's gravitational field; that a hungry bear has not just torn the objects away from the plant in order to swallow them; that the objects are not tied to a fleet of tiny little blimps that support them against gravity; etc. ad nauseam. Such simplifying assumptions are unavoidable in knowledge-based systems; their necessity is due to the *qualification problem* [McC77]. However, new information may violate some of the multitudes of tacit simplifying assumptions to which the knowledge base has committed (e.g., new information describing aquatic plants when only terrestial plants were tacitly expected), and such violations introduce conflicts (e.g., the disconnected leaves of aquatic plants float rather than drop.)

3. New information interferes with control knowledge implicitly encoded in some structure imposed on the knowledge base, such as the order in which rules appear. For example, it has been noted that simply adding new rules to MYCIN caused existing rules to apply erroneously or not at all [Die82, page 331]. [1]

These conflicts illustrate one general category of interaction that occurs between new and prior knowledge. New information can also interact synergistically with existing knowledge. Each synergistic interaction suggests new knowledge not explicitly contained in the new information. Two common types of synergistic interaction are:

1. Existing knowledge explains the new information. In Figure 1.1, the student's background knowledge determines that the leaf's cuticle restricts

[1] In contrast, learning in Waterman's poker player carefully placed new rules within the ordered list of rules to block application of faulty rules [Wat70].

water lost through the leaf and thus has an important physiological function. This teleological explanation suggests that other components of the shoot system (such as stems, flowers, and fruit), which also require some mechanism to restrict water loss, would benefit from having cuticles. Such teleological explanations are essential to understanding aspects of many domains [Sim81, DeK85, KC85, Dow90, Fra93].

2. New information explains existing knowledge. For example, adding the fact that chloroplasts contain the green pigment chlorophyll to a botanical knowledge base helps explain the existing default beliefs that the leaves of plants are green and capable of conducting photosynthesis [Mur90]. This explanation enables the system to justify its prior belief: *"Leaves are green <u>because</u> they contain chlorophyll."* Furthermore, this explanation suggests additional knowledge: *"Leaves that are not green do not contain chlorophyll and probably cannot conduct photosynthesis."* Explaining new information with prior knowledge is an important characteristic of comprehension [Gag85], and the ability to explain beleifs and conclusions is recognized as essential for knowledge-based systems and their development [SS89, Mor89].

Determining the interaction of new and prior knowledge facilitates the acquisition of knowledge beyond that explicitly contained in the new information. Identifying conflicts enables acquiring additional information to resolve the conflicts. Identifying synergistic interactions between new and prior knowledge enables acquiring other types of knowledge, such as generalizations of new information and explanations of new or prior beliefs.

The fidelity fallacy: Knowledge engineers often mistakenly conceive of symbols within a knowledge base as the domain (e.g., real world) concepts they represent rather than as what the symbols denote by virtue of their definitions within the formal system. Unlike the examples of conflicts among beliefs within a knowledge base discussed above, these fallacies transcend the system: they are misconceptions in the minds of the system's developers and result from limitations in the developers' understanding of the system's knowledge.

Such misconceptions lead to blunders in knowledge engineering. Often knowledge engineers tacitly assume properties of the represented concepts when formalizing a new fragment of knowledge. For example, when formalizing the knowledge that each vertebrate animal has a head, it is easy to omit the default constraint that each has precisely one head. Similarly, when formalizing knowledge about occupational situations, it is surprisingly easy to omit the default constraints that employees are clothed and living. The more "second nature" an expectation is, the more it resembles common sense, the more ingrained it is by our culture or experiences, the harder it can be to conceive and articulate explicitly when formalizing new knowledge.

These types of mistakes are examples of the *fidelity fallacy*, one of the most pervasive problems in developing knowledge-based systems [LG90, McD81]. Under this fallacy, an observer believes that what a symbol actually denotes within a formal system is what the observer expects it to denote. All too often the observer expects the symbols to denote some convenient subset of those aspects of the domain concept that the symbol is intended to represent.

To avoid this fallacy, knowledge engineers must keep their conceptions of what symbols denote within the formal system as accurate as possible. Consequently, it is important to show knowledge engineers the actual uses of

symbols in the formal system, what can and cannot be inferred about them, and what inferences they do and do not support. The dialogical aspect of learning exposes to the knowledge engineer the actual uses of symbols within the formal system. This exposure includes both symbols referenced by new information and relevant symbols in the knowledge base not referenced by new information. It enables knowledge engineers to identify precisely what a symbol denotes within the formal system, compare this to their expectations for what the symbol should denote, and thereby determine how the symbol fails to represent the intended domain concept. Thus, by actively investigating the consequences of new information and presenting those investigations to the knowledge engineer, the learning system reveals the current state of the system's knowledge, including the effects of the new information and how those effects diverge from what was intended by the knowledge engineer. Presenting the consequences of each knowledge-base modification to the knowledge engineer is essential for the incremental and inherently nonmonomtoic development of knowledge-based systems [Coh84, Mor89].

1.2.2 Enhancing learning opportunism

The study of knowledge integration as a learning task also explores ways to relax some of the applicability constraints imposed by traditional approaches to machine learning. The learning situation described in Figure 1.1 is more opportunistic than traditional tasks in machine learning in two ways:

1. The learner need not anticipate the content or form of new information.

2. The learner need not anticipate precisely the uses of acquired knowledge.

As the learning episode in Figure 1.1 begins, the student has no preconceptions about what material will be presented and how acquired knowledge will be used. The ability to learn without these preconceptions promotes opportunistic learning: the learner can accept and make use of whatever new information is encountered. However, learning without these preconceptions is more complex since learning methods now must include strategies for handling a wide variety of information and determining what lessons to learn from new information. These strategies introduce sources of *learning bias,* knowledge that guides learning. [2] This section explains the significance of these two aspects of knowledge integration.

Relaxing preconceptions of content: A learner often has prior expectations about the topic or content of new information. Differing learning situations afford the learner differing degrees of expectation:

1. At one extreme, a learner might pose a very focused question and expect to learn its answer. For example, the learner might ask: *What is Fred's home phone number?* Here the learner has very high expectations about the content of the new information.

2. At another extreme, the learner might not anticipate new information. For example, an acquaintance might unexpectedly announce: *Adult butterfly fish are capable of changing gender.* In this situation, the learner does not anticipate encountering new information and, consequently, cannot possess expectations about its content.

[2] Learning bias determines which new fragments of knowledge a learning system will acquire when there many alternative new fragments of knowledge that could be acquired.

In between these extremes is a range of learning situations in which the learner may have strong or weak expectations about the content of the new information.

Assumptions about the content of new information can simplify learning. Expectations relating the content of new information to a particular topic focuses the learner's attention on existing knowledge relevant to that topic. This is the principle underlying *advance organizers* in human education theory [Aus63, RS83, May80]. However, these same assumptions, when programmed into a machine learning system, restrict the applicability of the learning system: new information that violates the assumptions cannot benefit the learner.

Traditional approaches to machine learning adopt strong preconceptions about training content. Training is usually required to be either ground observations in the domain or example problems that the system must perform. These restrictions preclude learning from new information that includes general domain principles or explanations or new terms that extend the representation language. [3] Therefore, while it is advantageous for a learner to be able to exploit available and warranted preconceptions about the content of new information, it is preferable that general learning methods not commit to learning only from new information having a narrowly specified content.

Relaxing preconceptions of application: A learner often has some expectation about the applications of acquired knowledge. Again, differing learning situations support diverse degrees of expectation:

[3] Exceptions to this trend include ANT, which admits both instance-level observations and general rules while learning natural language grammars [LM90], PROTOS, which admits explanations of problem solutions that include general domain principles while learning diagnostic audiology [PBH90], and FOO, which operationalizes general strategic advice while learning strategies for playing the game of hearts [Mos83].

1. *Intentional learning* occurs when the learner approaches a learning task with specific expectations of at least one application of the knowledge to be acquired. Often learning occurs during problem-solving: the learner acquires knowledge or skill while trying to perform some specific task (e.g., solve a particular math problem, write a particular sentence, feed and water a particular plant, identify a cup, buy fish for an aquarium, call Fred at home). Because a clear and immediate application of the acquired knowledge exists, there is no doubt about what is to be learned. In such situations, new information is processed according to the specific requirements of the application: expectations about the use of the to-be-acquired knowledge guide intentional learning.

2. *Incidental learning* occurs when the learner has no specific predetermined expectations for the use of acquired knowledge. For example, unanticipated discourse can provide new answers to questions not previously considered, such as the new information about butterfly fish described above. An absence of known applications for acquired knowledge introduces additional complexity because the learning methods must determine *what* to learn from new information, not just *how* to learn. Strategies for determining what to learn are an important source of learning bias. In the absence of strong use expectations, learning is essentially a reactive, bottom-up process: constraints other than predetermined use expectations guide incidental learning.

Intentional and incidental learning are two extremes of a continuum. In between is a spectrum of learning situations ordered by the degree to which the learner has expectations for the use of acquired knowledge. Much of human learning occurs through the comprehension of mundane information when the

potential uses of acquired knowledge are unknown. In such situations, the focus of learning is comprehension, rather than application.

Often the learner has partial or generic expectations about the future uses of acquired knowledge. When students open up textbooks required by a course in which they are enrolled, they likely share the preconception that information contained in the textbooks will be used to perform some tasks required by the course (e.g., answer questions on examinations, perform lab experiments, etc.). However, each student typically does not know what tasks will be required (e.g., which questions will appear on an exam). Consequently, students cannot anticipate precisely what information presented in the textbooks will be essential, nor what general lessons to draw from the presented information, nor how it will have to be used during the examinations and lab exercises. Nor can students predict precisely how the acquired knowledge will be used for tasks beyond those imposed by the course. Human learning is thus opportunistic: knowledge is acquired without precise stipulations of its intended uses; such learning is ubiquitous [GH86]. Machine learning should be equally opportunistic.

Traditional approaches to machine learning commit to precise restrictions on the eventual use of acquired knowledge. For, example, most concept acquisition systems are designed to acquire knowledge dedicated to performing classification, applying the acquired definitions of the target concepts to unclassified instance descriptions. [4]

[4]One notable exception is AM [Len76], which learned by discovering new concepts in the domain of mathematics. In the absence of an assumed application task, AM confronted the issue of determining what to learn. It exploited an extensive set of heuristics that estimated the *interestingness* of a new domain concept, learning only new concepts deemed sufficiently interesting.

Restricting the use of acquired knowledge is not merely a commitment of the learning methods. This commitment is also made at the more fundamental level of the learning task: traditional machine learning tasks narrowly (although often implicitly) constrain the application of knowledge acquired by the learning methods that perform the tasks. [5] Learning methods that perform these tasks are generally not capable of acquiring other forms of knowledge than that prescribed by the tasks. Intuitively, traditional machine learning tasks are characterized as: given an application task and information about how to perform the task (e.g., examples of input and desired output), develop a procedure to perform the task efficiently and correctly. In contrast, a more opportunistic learning task can be characterized as: given new information, determine why the information is to be believed, what other beliefs might also be true, and what existing beliefs should no longer be held.

While it is advantageous that a learner be able to exploit available and warranted preconceptions about the applications of new information, methods of learning can be more opportunistic without restricting the use of acquired knowledge. Consequently, a general learning task must not commit to particular applications of acquired knowledge.

This dissertation raises the question: Why assume learning occurs only within a problem-solving context? Knowledge integration differs from traditional machine learning tasks by not to committing to specific application tasks. This difference introduces complexity because learning methods must include strategies to determine what to learn from new information, not just

[5] A *learning task* is a specification of a computational problem that is to be solved by a learning program; a *learning method* is a strategy for solving (aka performing) a particular learning task.

strategies for how to learn. However, this difference also significantly broadens the scope of the learning task, and, consequently, the applicability of the learning methods.

1.3 Dissertation organization

This dissertation investigates knowledge integration as a machine learning task. Chapter 2 presents a computational model, called REACT, for performing this task. REACT identifies several key capabilities required by any solution to the task of knowledge integration. The model also serves as conceptual framework for performing knowledge integration.

Chapters 3, 4 and 5 describe how the computational model proposed in Chapter 2 has been implemented in the computer program KI. Chapter 6 presents some empirical experiments with this program. Chapter 7 surveys other research that is relevant to knowledge integration. Finally, Chapter 8 includes a summary of the contributions and limitations of this research, and suggests an agenda for future work.

Included in the appendices are some of the less essential technical details of the implementation as well as several informal discussions on topics that are relevant (but not essential) to the thesis of this dissertation. Appendix A describes the learning environment in which knowledge integration is assumed to occur. It also presents an intuitive definition of learning, and a characterization of learning as a state-space search problem. Appendix B provides a specification of knowledge integration as an information-processing task and discusses learning goals that can guide learning in the absence of specific expectations for the use of acquired knowledge. Appendix C describes how the implementation performs the task of interpretation – that is, translating new

information, presented as semantic networks, into expressions in the internal representation language. Appendix D lists some heuristics for estimating the interestingess of a proposition that are used to guide the implemented learning program as it performs knowledge integration.

Chapter 2

REACT: a Computational Model for Knowledge Integration

Although the ubiquity and importance of knowledge integration has been recognized in research on human learning, the computational issues in performing knowledge integration have remained largely unexplored. This chapter discusses properties of methods for performing knowledge integration. The first section reviews findings in cognitive psychology that demonstrate the importance of both actively relating new and existing knowledge and focusing attention during learning. The second section proposes a computational model for performing knowledge integration that reflects these findings.

2.1 Knowledge Integration as Human Learning

Humans provide the best examples of intelligent learning systems. Because the essential computational properties of human learning may benefit any learning system, cognitive mechanisms or effects that are pervasive in theories of human learning suggest relevant computational issues for machine learning. Two tenets of cognitive theories of human learning are:

1. Learning is an active process that relates new and prior knowledge.

2. The resources for performing this activity are limited.

Relating new and prior knowledge is essential for learning; however, opportunities for performing this activity generally exceed the available cognitive resources. Learning is thus necessarily selective and exclusive: only a portion of the potential relations between new and prior knowledge will ever be established.

2.1.1 Learning actively relates new and prior knowledge

The view that learning involves relating new information to existing knowledge is central to contemporary theories of human learning and comprehension. It is fundamental to Piaget's general learning theory of assimilation and accommodation [Pia46] and to Kintsch's model of discourse comprehension [KvD78]. It is the foundation of the *learning strategy hypothesis*, a popular interpretation of numerous studies [May80]:

> [A]ctivities aimed at making the learner actively integrate new information into existing knowledge affect the encoding, storage, and eventual use of new material on performance tests.

A central component of many theories emerging from modern educational psychology is *elaboration*, the process of embellishing new information during comprehension. It arises from the interaction between new information and the learner's existing knowledge. There are potentially as many elaborations of new information as there are questions; both are unbounded. Figure 2.1 presents a partial list of common types of elaborations [Gag85, Wei78].

Extensive empirical investigations support the importance of elaboration for successful comprehension and learning [Gag78]. Studies by Haviland and Clark [HC74] and by Reder [Red79] indicate that readers elaborate to fill

1. confirm or explain
   (a) **statement:** *This plant looks as if it has died.*
   (b) **elaboration:** *Of course it died; it wasn't given any food; all living things die without food.*
2. recall examples
   (a) **statement:** *Eskimos believe polar bears were sent by the gods to keep human population low.*
   (b) **elaboration:** *I remember seeing a polar bear once, at the St. Louis Zoo.*
3. attribute detail
   (a) **statement:** *The Hispaniola is a fine vessel, a schooner with smart trim.*
   (b) **elaboration:** *The Hispaniola probably has three masts, since it is a schooner.*
4. make predictions
   (a) **statement:** *... and Scarlett O'Hara said: "Tomorrow is another day."*
   (b) **elaboration:** *I bet Scarlett never did remarry. I wouldn't!*
5. form analogies
   (a) **statement:** *Endosperm is the nutrient within a plant seed.*
   (b) **elaboration:** *An endosperm seems to be like the yolk of an egg.*
6. make comparisons
   (a) **statement:** *The basking shark grows up to forty feet in length.*
   (b) **elaboration:** *The basking shark is even bigger than the great white shark.*
7. draw conclusions
   (a) **statement:** *The prime rate increased by 1%.*
   (b) **elaboration:** *It's going to cost more to borrow money.*
8. motivate the lesson
   (a) **statement:** *Next, we consider the events leading up to the first world war.*
   (b) **elaboration:** *I'm supposed to understand what triggered World War I.*
9. evaluate
   (a) **statement:** *The ozone layer is deteriorating.*
   (b) **elaboration:** *This is a serious problem!*
10. identify principles
    (a) **statement:** *Consumers compete to purchase desired goods.*
    (b) **elaboration:** *As demand goes up, then price goes up too.*

Figure 2.1: Common Types of Elaboration

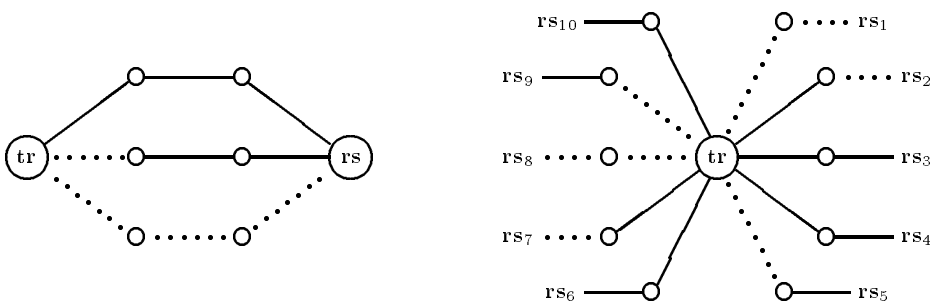
in missing details in text, and that elaboration is performed as information is encountered rather than during recall through reconstructive memory. The essential role of elaboration during comprehension is masked by its pervasiveness [GH86, page 474]:

> People are so practiced at integrating new information with old information in memory that they usually don't realize how meaningless individual sentences could be if the connections could not be made.

Studies by Mayer [May80] suggest two ways that elaboration enhances learning. First, elaboration increases the student's ability to recall information by establishing relations between the new information and existing knowledge. Second, elaboration expands the content of new information by adding new beliefs to the student's knowledge.

Advantages of elaboration: Elaboration facilitates subsequent recall by establishing relations between training and existing knowledge [And83]. Each new relation introduces a reminding path from existing knowledge to the new information. This increases both the number of paths between a given reminding source and the target response and the number of reminding sources that can stimulate recall of the target response. [1] Each additional path that connects training to prior knowledge enhances the probability of successful recall. For example, the elaboration of a query may overlap or recreate the elabora-

[1] In a recall test, the *reminding source* is provided to stimulate the subject's recall of the *target response*.



(a) Elaboration establishes new connections between a target response **tr** (e.g., the new information) and an existing reminding source **rs** (e.g., a fragment of prior knowledge). **(b)** Elaboration establishes connections between a target response and new reminding sources.

Figure 2.2: Improving recall with elaboration

tion of the training, resulting in a greater chance of target-response recall (what Anderson calls *semantic triangulation* [And83]).

Figure 2.2 presents a graph-theoretic account of the effects of elaboration. There are two useful interpretations of this graphical account:

1. In the first, nodes in the graph denote entities; arcs denote binary relations over the entities. Elaboration increases the number of relational paths in the graph, thereby making the graph more interconnected.

2. In the second, nodes denote knowledge segments (e.g., sets of beliefs); arcs denote applied operators (e.g., rules). Each operator derives one segment from another segment. Elaboration increases the number of derivational paths in the graph, thereby making the graph more interconnected.

Elaboration enhances subsequent problem-solving performance by expanding the content of new information. Elaborating new information results in addi-

tional beliefs. The additional beliefs may include new inference methods (e.g., rules) as well as new facts. Sometimes inferred information can be more useful than the explicit contents of new information [RS83]. Either new beliefs are added explicitly to the learner's knowledge and are directly accessible, or they are added implicitly and are quickly computable [Red79]. Empirical studies suggest that elaboration improves response times when subjects are asked to estimate the plausibility of statements [Red79, Red82b].

Disadvantages of elaboration: As might be expected, not all elaboration benefits learning. Elaboration may produce incorrect beliefs by propagating learners' misconceptions. However, the proliferation of false beliefs can be advantageous by increasing the likelihood of eventually uncovering and repairing the underlying misconceptions.

Secondly, important aspects of the training can be obscured by a mass of elaborated trivia. Too many related beliefs impedes recall of a particular fact in what is called the *fan effect* [And83]. Students sometimes lose the important points of a lesson in a mass of mundane conclusions constructed through elaboration. This is especially common when a teacher attempts to provide explicit elaborations of subject matter to students, rather than allowing students to form their own idiosyncratic elaborations through their own interests and expertise. Reder [Red82a] demonstrates that when textbooks include elaborations of main points, readers perform worse on recall tests than when they are exposed to only summary information. However, when students provide their own idiosyncratic elaborations, their recall performances improve, especially when the elaborations produced are sufficient to reconstruct the lesson's main conclusions.

A third (and related) potential hazard occurs when the inferences completed during elaboration are not appropriate for the particular goals of a learning episode, as determined by the criterial task. This is explained by the theory of *transfer-appropriate processing*, which stresses the importance of putting the processing of information during learning in correspondence with the processing required by the criterial task [MBF77]. Experiments demonstrate that the benefits of learning activities are strictly relative to the criterial task used to evaluate learning performance. Consequently, learning performance improves when elaboration reflects the processing required by the criterial task and suffers when elaboration is not relevant.

2.1.2 Limited resources force selective learning

For new information to be comprehended, it must include references to concepts familiar to the learner [HC74]. These references form the *given* component of new information, so called because they are used to ground the new information to existing segments of knowledge. This knowledge can then be embellished by the *new* component, the portion of new information that is not already known by the learner. The given provides indices into the learner's existing knowledge that locate where the new information is to be recorded. Each segment of existing knowledge that shares content with the given is in some way relevant to the new information and becomes a candidate for use during elaboration. Many alternative segments of knowledge may share content with the given of new information.

Because humans have limits on their resources for performing mental activities, cognitive processes – such as elaboration – compete for shares of the limited resource that is human attention. Elaboration is consequently

selective and exclusive: not every relevant segment of existing knowledge can be used during elaboration to embellish new information. Selecting among the alternative segments of relevant knowledge is the task of *recognition*.

Two different kinds of stimulus are required to select a segment of relevant knowledge: a stimulus that is specific to the content of that segment (e.g., the content that overlaps the given component of new information), and a stimulus that is not specific to the segment's content [Kah73]. This non-specific stimulus is often called *activation*. The distribution of activation manifests human selective attention, differentiating the cognitive activities that are performed (e.g., relating new information to segments of knowledge that are selected during recognition) from those that could be performed (e.g., relating new information to segments of knowledge that share content with the given component of new information).

As new information is encountered, some portion of the activation stimulus is distributed among the segments of existing knowledge that share content with the given. Those segments whose activation levels surpass a threshold become activated (e.g., are drawn into short-term memory, enter conscious thought), and only activated segments are eligible to participate in elaboration. Consequently, the importance of recognition for learning is fundamental: the existing knowledge that is activated during learning determines both the content and the quality of acquired knowledge [Sch76].

2.2 REACT

This section presents REACT, a computational model for performing knowledge integration. Such a model is useful because it defines a functional decomposition of the learning task and provides a conceptual framework for

describing methods of performing knowledge integration. The functional decomposition facilitates independent research on strategies for performing each sub-task. The conceptual framework is useful for comparing and relating different methods of performing knowledge integration.

The proposed model embodies three premises about learning:

1. Knowledge integration, and indeed all conceptual learning, ultimately requires affecting changes to a body of knowledge. Consequently, all methods that perform knowledge integration will have some capability to perform *adaptation*, that is, to modify existing knowledge.

2. Knowledge integration, and all but the most trivial forms of conceptual learning, requires determining how new and prior knowledge relate. Consequently, all methods that perform knowledge integration will have some capability to perform *comprehension*, that is, to make sense of the new information by relating it to prior knowledge.

3. Comprehension involves two essential capabilities, *recognition* and *elaboration*. Recognition selects existing knowledge that is relevant to new information. Elaboration relates the selected knowledge to the new information. Consequently, all methods that perform comprehension will have some capabilities to perform both recognition and elaboration.

The model identifies a functional decomposition expected of any system that performs the task. That is, any method that performs knowledge integration must be capable of performing at least three activities: recognition, elaboration, and adaptation. The model also embraces the hypothesis that any method capable of performing these three activities, in suitably compatible ways, will be

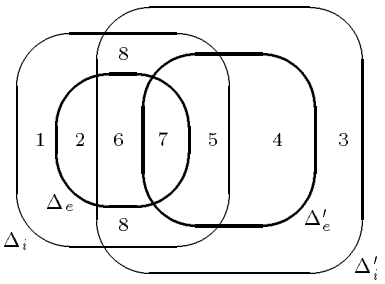
able to perform some form of knowledge integration. However, this functional decomposition need not be manifest as a physical or architectural decomposition within any learning system.

The proposed model makes explicit the intuitive relation between comprehension and learning. Comprehension makes sense of presented information by relating it to existing knowledge and reveals opportunities to add or revise beliefs, especially when the encountered information is new. Exploiting these opportunities constitutes learning. In Figure 1.1, the student's comprehension of new information establishes the physiological advantage that a cuticle provides to a plant leaf. Relating the new information to existing knowledge about transpiration reveals a specific learning opportunity: that of generalizing the new information for other transpiring plant organs.

2.2.1 Comprehension: relating new and prior knowledge

New information comprises a set of explicit beliefs that add to or retract from the explicit beliefs of a learner's theory. Comprehension reveals how new and prior knowledge interact; it explores the consequences of new information for prior knowledge and facilitates adapting new and prior knowledge when these consequences violate learning goals.

Figure 2.3 illustrates the types of belief transformations that occur during a knowledge-base modification. It partitions the beliefs of the theory – both before and after the modification – into eight possible dispositions and illustrates how changes to explicit knowledge can manifest changes in implicit knowledge. Comprehension explores the implicit changes caused by a knowledge-base modification as it relates new and prior knowledge. Figure 2.4 specifies necessary (not sufficient) conditions for perfoming the task of compre-



A theory (Δ) comprising explicit beliefs (Δ_e) and implicit beliefs (Δ_i) is modified, resulting in a new theory (Δ') comprising explicit beliefs (Δ'_e) and implicit beliefs (Δ'_i). Each numbered region illustrates a type of belief transformation.

Region 1: lost implicit beliefs
Region 2: lost explicit beliefs
Region 3: new implicit beliefs
Region 4: new explicit beliefs
Region 5: old implicit beliefs now explicit
Region 6: old explicit beliefs now implicit
Region 7: old explicit beliefs remain explicit
Region 8: old implicit beliefs remain implicit

Figure 2.3: Types of Belief Change

hension.

A total exploration of implicit beliefs is not possible for significant theories (e.g., knowledge bases that are large or that are represented in expressive languages, such as first-order logic). Therefore, comprehension selectively explores the consequences of new information under resource bounds. Methods

Given: (1) Δ: a set of beliefs (e.g., prior knowledge)
(2) Θ: a set of beliefs (e.g., new information)
(3) $\vdash$: an inference procedure
(4) b: resource bounds (e.g., a bound on execution time)

Find: Φ: beliefs derivable from applying $\vdash$ to elements of $\Delta \cup \Theta$
$\{p \mid (\Delta + \Theta \vdash p)\}$
within resource bounds b.

Figure 2.4: The task of comprehension

that perform this task must determine which beliefs from the entailment of new and prior knowledge are established. Recognition selects segments of explicit knowledge (i.e., subgraphs of the knowledge base) that include the given components of new information. Elaboration selects, by making explicit, segments of implicit knowledge (i.e., subgraphs of the inferential closure) derived from new information and prior knowledge selected during recognition.

The distinction between recognition and elaboration, between *selecting what to reason about* and *doing the reasoning*, has long been acknowledged in Artificial Intelligence. McCarthy, in his seminal paper proposing an advice-taking program, observes [McC58]:

> The intelligence, if any, of the advice taker will not be embodied in the immediate deductive routine. This intelligence will be embodied in the procedures that choose the lists of premises to which the immediate deductive routine is to be applied.

Performing recognition and elaboration may be piecemeal and interleaved, or they each may be performed as complete and sequential stages (with recognition occurring before elaboration).

Recognition – focusing attention during comprehension: Comprehension relates new information to existing knowledge; recognition selects which segments of existing knowledge the new information will be related to. In other words, recognition focuses the attention of the learner during comprehension. Figure 2.5 presents a task specification for recognition.

The problem of focusing attention can be stated clearly in terms of the generate and test paradigm. Given a current state and a set of operators that

Given: (1) Δ: an initial knowledge base
(2) Θ: a set of beliefs (e.g., new beliefs added to Δ)
(3) $\vdash$: an inference procedure

Find: Ψ: a subset of Δ s.t. $\vdash$ will be applied to elements of $\Psi \cup \Theta$

For the general task of recognition, no specification of a goal of the ensuing inference (i.e., applying $\vdash$ to $\Psi \cup \Theta$) may be assumed.

Figure 2.5: The task of recognition

map between states, determine which region (i.e., subset of accessible states) within the state space to generate. When the current state comprises beliefs denoting new information to be understood, and the operators relate these beliefs to a body of prior knowledge (e.g., by performing inferences), then the problem of focusing attention corresponds to the task of recognition during comprehension: the new information is comprehended in light of a particular subset of prior knowledge selected during recognition.

An important property of comprehension is how open and unconstrained it is. In general, a recipient of new information does not know in advance the contents of the new information nor its possible uses. Therefore, the learner cannot know in advance what existing knowledge will be relevant for making sense of the new information. For this reason, comprehension must include a reactive, data-driven process. This reactive aspect is an essential feature of performing recognition: the learner responds to the (unanticipated) content of new information.

Each subgraph of the knowledge base that includes some component of the new information (e.g., some component of the given) is relevant to the new information. The powerset of these subgraphs comprises the alternative segments of existing knowledge that can be related to the new information dur-

ing elaboration; they are the candidates for selection during recognition. This identifies two important issues confronting methods for performing recognition:

1. *Determining the appropriate grain size of the segment of existing knowledge selected during recognition.* The selected segment could be a single term, a single belief (e.g., a single fact or rule), or a large set of beliefs (e.g., the entire knowledge base). In general, the more exclusive recognition is, the more restricted subsequent elaboration will be.

2. *Determining focusing criteria for selecting from among a vast number of candidate knowledge segments.* The candidates all share some component with the given of new information; consequently, they already satisfy a relatively weak standard for establishing their relevance. Additional focusing criteria can establish a stronger *principle of relevance* that selected segments must satisfy. (Examples are discussed in Chapter 4.) Relevance-based focusing criteria relate the contents of candidate segments with the contents of new information. Alternatively, focusing criteria may exploit formal properties of the candidates that do not consider the content of the candidate knowledge segments or their relations to new information (e.g., an ordering of the rules in the knowledge base, an ordering of the conjuncts appearing in the antecedent of a rule, etc.).

The REACT model does not specify the appropriate grain size nor the appropriate focusing criteria to be adopted by methods that perform recognition. Nor does the model specify how the selected knowledge will be used during elaboration; any style of reasoning, such as forward chaining, backward chaining, general resolution, ground resolution, case-based reasoning, etc., can be used. But differing styles of recognition will be appropriate for differing styles

Given: (1) Ψ: a set of beliefs (e.g., existing knowledge)
(2) Θ: a set of beliefs (e.g., new information)
(3) $\vdash$: an inference procedure
(4) b: resource bounds (e.g., a bound on execution time)

Find: Φ: beliefs derivable from applying $\vdash$ to elements of $\Psi \cup \Theta$
$\{p \mid (\Psi + \Theta \vdash p)\}$
within resource bounds b.

Figure 2.6: The task of elaboration

of reasoning. For example, case-based reasoning requires selecting appropriate cases from among a library of alternative cases, while production systems require selecting appropriate rules to fire from among those that are triggered (e.g., conflict resolution). Consequently, the tasks of recognition and elaboration are not entirely independent; both are influenced by whatever mechanisms structure the system's knowledge.

Elaboration – embellishing new information during comprehension: Comprehension relates new information to prior knowledge; elaboration relates the new information and the prior knowledge selected during recognition by establishing beliefs that they entail; it is the second phase of comprehension. Figure 2.6. specifies necessary conditions for performing the task of elaboration.

The goals of learning (e.g., see Appendix B.2) require determining whether desired properties (e.g., consistency, completeness, etc.) hold on a set of beliefs. The beliefs that must satisfy these properties include both the learner's implicit and explicit beliefs. Learning therefore entails not only appraising the explicit content of the extended theory but its implicit content as well. Elaboration explores the impact of new information on the implicit content by making explicit a *partial entailment* of new information and prior

knowledge recalled during recognition. The partial entailment comprises a finite subset of the possibly infinite beliefs entailed by the new information and prior knowledge and is made explicit by applying the inference procedure to the new and sekected prior knowledge. Two important issues confronting methods of performing elaboration are:

1. *Operationalizing new and selected prior knowledge for the inference procedure.* The available inference procedure may not be directly applicable to the beliefs included in the new information and the prior knowledge selected during recognition. For example, these beliefs may include only rules, while the inference procedure may permit applying rules only to ground propositions and not to other rules (i.e., general resolution is not permitted). Consequently, new information and selected prior knowledge must be operationalized so that the inference procedure can apply to them.

2. *Maintaining the dependencies between supporting knowledge and the facts that are established as inferences are completed.* Methods for detecting and exploiting learning opportunities (examples are discussed in the next chapter) require identifying the facts and rules that support a derived belief. Therefore, the inference graphs that are completed during elaboration must be maintained for subsequent analysis during adaptation.

An important subset of beliefs established during elaboration are those supported by new information. A set of beliefs Θ (within knowledge base Δ) *supports* another set of beliefs Φ precisely when Θ participates in some derivation of Φ:

$$\exists\ (\Psi)\ (\Psi\ \subseteq\ \Delta)\ \&\ \neg(\Psi\ \vdash\ \Phi)\ \&\ (\Psi + \Theta\ \vdash\ \Phi)$$

The *consequences* of the new information is the set of beliefs supported by the new information.

Appraising the consequences of new information: One common and important benefit of comprehension is that it reveals the consequences of adding new information to prior knowledge. The analysis of interactions between new information and existing knowledge is fundamental to nontrivial modes of knowledge integration. Even determining that new information is in fact new (i.e., constitutes new beliefs) requires establishing that it includes beliefs not accessible as consequences of existing beliefs.

Intuitively, determining how new information interacts with prior knowledge involves:

1. identifying new beliefs supported by the new information (regions 3 and 4 of Figure 2.3) [2]

2. identifying prior knowledge supported by the new information (regions 5, 6, 7, and 8 of Figure 2.3)

3. identifying beliefs contained in the new information supported by prior knowledge (regions 5 and 7 of Figure 2.3)

4. identifying conflicts between the new information and prior knowledge (regions 1 and 2 of Figure 2.3)

Figure 2.7 presents a task specification for identifying the consequences of adding information to existing knowledge. This task identifies beliefs supported

[2]Unless otherwise noted, assume the new information includes asserting regions (of Figure 2.3) 4, 5, and a subregion of 7, and unasserting regions 2 and 6.

Given:
1) Δ: an initial knowledge base
2) Θ: new beliefs added to Δ
3) $\vdash$: an inference procedure
4) b: resource bounds (e.g., a bound on execution time)

Find: $\Phi = \Phi_+ \cup \Phi_-$: the consequences of adding Θ to Δ
Φ_+: beliefs supported by the new information, defined as
$\{p_+ \mid (\exists \Psi_+)(\Psi_+ \subseteq \Delta)\ \&\ \neg(\Psi_+ \vdash p_+)\ \&\ (\Psi_+ + \Theta \vdash p_+)\}$
Φ_-: beliefs disputed by the new information, defined as
$\{p_- \mid (\exists \Psi_-)(\Psi_- \subseteq \Delta)\ \&\ (\Psi_- \vdash p_-)\ \&\ \neg(\Psi_- + \Theta \vdash p_-)\}$
within resource bounds b.

Figure 2.7: Assessing the consequences of new information

by the new information and constitutes a sufficient condition for performing comprehension. Some interesting special cases are:

1. **Identifying new beliefs supported by the new information:**

   (a) $\Psi_+ = \Delta$: belief p_+ was not supported by prior knowledge but is supported with the addition of the new information (knowledge-level expansion).

      i. $p_+ \in \Theta$: belief p_+ is in region 4 of Figure 2.3.

      ii. $p_+ \notin \Theta$: belief p_+ is in region 3 of Figure 2.3.

   (b) $\Psi_+ = \{\}$: belief p_+ is derivable from just the new information (and is not tautological).

2. **Identifying corroborations between new and prior knowledge:**

   (a) $p_+ \in \Delta$: the new information supports explicit prior beliefs (affirmation).

      i. $p_+ \in \Theta$: the new information includes explicit prior beliefs; belief p_+ is in region 7 of Figure 2.3.

ii. $p_+ \notin \Theta$: the new information supports explicit prior beliefs; belief p_+ is in region 6 or 7 of Figure 2.3.

(b) $(\Delta \vdash p_+)$ & $(p_+ \notin \Delta)$: the new information supports implicit prior beliefs.

i. $p_+ \in \Theta$: prior knowledge supports beliefs contained in the new information; belief p_+ is in region 5 of Figure 2.3 (validation).

ii. $p_+ \notin \Theta$: the new information supports prior implicit beliefs; belief p_+ is in region 8 of Figure 2.3 (local expansion).

3. **Identifying conflicts between new and prior knowledge:**

(a) $\Psi_- = \Delta$: the new information refutes beliefs supported by prior knowledge (knowledge-level contraction).

i. $p_- \in \Delta$: the new information refutes explicit prior beliefs; belief p_- is in region 2 of Figure 2.3.

ii. $p_- \notin \Delta$: the new information refutes implicit prior beliefs; belief p_- is in region 1 of Figure 2.3.

(b) $\Psi_+ \vdash \neg p_+$: the new information provides new support for the negation of a previously supported belief (local revision). [3]

i. $\Psi_+ = \Delta$: the new information negates a previously supported belief (knowledge-level revision).

ii. $\Psi_+ = \{\}$: the new information is inconsistent.

(c) $p_- \in \Delta$: the new information refutes support for prior explicit beliefs.

[3] Analogously, $\Psi_- + \Theta \vdash \neg p_-$: the new information negates a previously supported belief (local revision).

(d) $\Psi_- \vdash \neg p_-$: the new information resolves a local contradiction.

(e) $\Psi_+ + \Theta \vdash \neg p_+$: the new information supports a local contradiction.

(f) $(p_+ \in \Theta)$ & $(\neg p_+ \in \Delta)$: the new information explicitly contradicts prior knowledge.

Each of these types of interaction between new and existing knowledge affords learning opportunities; detecting and exploiting learning opportunities occurs during adaptation.

2.2.2 Adaptation – detecting and exploiting learning opportunities:

Although many conditions qualify as learning opportunities and can trigger adaptation, only three are discussed below. Examples of each of these learning opportunities are presented in Chapter 5.

1. One type of learning opportunity occurs when comprehension identifies new and interesting beliefs that are supported by new information, as illustrated in Figure 1.1. In response to the new information describing a leaf cuticle, the learner finds support for the belief that the other transpiring organs also have a cuticle and suggests this to the teacher. Previously, support for this belief was evident.

2. A second type of learning opportunity occurs when comprehension identifies conflicts between new and prior knowledge, also illustrated in Figure 1.1. The learner concludes that the leaf cuticle inhibits photosynthesis; this conflicts with the expectation that leaves perform photosynthesis. In order to resolve the inconsistency, the learner suggests adopting additional beliefs: either portals exist in the leaf covering or the cuticle only partially covers the leaf.

3. A third, important type of learning opportunity occurs when comprehension establishes how new information can explain existing beliefs that were previously assumed without explanation. For example, when told that chloroplasts contain chlorophyll, the learner is able to explain why leaves are green and can perform photosynthesis [Mur90]. Previously, these beliefs were known, but the system was unable to explain them. The advantages of possessing explanations of beliefs is well recognized [Swa83, SWMB85, SS77], yet learning behaviors that acquire explanations of existing beliefs have not been widely investigated in machine learning research.

The REACT model does not commit to a particular set of conditions that trigger adaptation. These are specified as the *admissibility criteria,* conditions (e.g., consistency requirements) that must be satisfied by the resulting knowledge base, and they may reflect any learning goal, such as those discussed in Appendix B.2. Two significant issues confronting methods that perform adaptation are detecting learning opportunities (e.g., violations of the admissibility criteria) among the results of elaboration and exploiting learning opportunities by determining which knowledge-base modifications to make in order to satisfy the admissibility criteria. A task specification for adaptation – modifying knowledge to comply with the admissibility criteria – is presented in Figure 2.8.

An important point of comparison among machine learning systems is the catalyst, or trigger, conditions for changing the knowledge base. Almost all learning systems rely on observed performance failures; learning corrects the observed failure. One important class of performance failure occurs when the given classification of a training instance is not consistent with the target

Given: (1) Δ a set of beliefs (e.g., existing knowledge)
(2) Ψ: a subset of Δ
(3) Θ: a set of beliefs (e.g., new information)
(4) $\vdash$: an inference procedure
(5) Φ: a set of beliefs from inferential closure by applying $\vdash$ to $\Psi \cup \Theta$
(6) Γ: a predicate on belief sets (e.g., consistency requirements)

Find: (1) Υ: subsets of Φ that fail Γ
(2) Ψ' and Θ': transformations of Ψ and Θ that would retract the elements of Υ from Φ
(3) Δ': a transformation of Δ that reflects Ψ' and Θ' (e.g., $\Delta' = ((\Delta - \Psi) - \Theta) + \Psi' + \Theta'$)

Figure 2.8: The task of adaptation

concept definition. Learning revises the target concept definition to resolve the inconsistency. A second important class of performance failures occurs when a system manages to compute the correct response to a request, but the computation was costly. Learning revises the knowledge to ensure the response can be inexpensively computed for subsequent requests. However, performance failure is not the only catalyst for learning. AM, for example, uses heuristics to determine whether a new mathematical concept is *interesting* enough to add to its current knowledge [Len76] and Cobweb uses heuristics that assess the predictive power of new potential concepts to determine which concepts should be learned [Fis87]. Additional examples will be presented in Chapters 3 and 5.

2.2.3 Discussion

The REACT model decomposes the very general task of knowledge integration into three fairly focused constituent tasks. Comprehension occurs when beliefs are established from the entailment of new information and selected segments of prior knowledge: recognition selects segments of prior knowledge, and elaboration establishes the beliefs. Learning occurs when new beliefs are added or existing beliefs are modified during adaptation.

The model does not commit to a type of knowledge or style of reasoning (such as case-based, production rules, logic, frame-based); it requires only that the constituent activities are suitably compatible (e.g., the method of recognition is capable of selecting a portion of knowledge to be used during elaboration, and the adaptation methods are sensitive to the results of elaboration).

Adaptation methods do not completely define all learning opportunities; rather, learning opportunities transcend the closed system. By generating predictions of the consequences of new information, the elaboration method reveals to the teacher the state of the learner's knowledge, including any misconceptions that occur. The teacher can respond with unsolicited additional information to extend or revise the learner's knowledge. Consequently, recognition and elaboration should not be finely tuned to a specific adaptation regime. They should use the same reasoning mechanisms and (as much as is possible) strategies the system would exploit during problem solving.

Learning Bias: One of the contributions of REACT is that it identifies tacit sources of learning bias. Learning bias is knowledge that guides learning (e.g., the strategic knowledge that determines which particular knowledge-base modifications will be affected when many alternatives are possible). The model identifies three activities as being individually necessary and jointly sufficient for performing knowledge integration. Methods that perform each activity introduce learning bias:

1. Adaptation detects and exploits learning opportunities. The catalyst conditions that trigger each adaptation method determine when knowledge is modified during learning; they manifest one important form of

learning bias. Furthermore, each method's strategy for modifying the knowledge base in response to the triggering condition (i.e., to satisfy the admissibility criteria) manifests a second important form of learning bias.

2. Elaboration explores the consequences of new information as it establishes implicit beliefs supported by the new information and selected segments of prior knowledge. The particular beliefs established are the grist for adaptation; they determine what learning opportunities will be encountered. Consequently, the method of elaboration manifests a form of learning bias to the extent that it determines which beliefs are established.

3. Recognition determines which segments of prior knowledge are to be related to the new information. It has a dramatic effect on what beliefs can be established during elaboration and so manifests an important source of learning bias.

Sources of learning bias traditionally recognized in Machine Learning literature (e.g., see [Die86]) include only biases for implementing adaptation strategies; REACT also identifies methods of both recognition and elaboration as important sources of learning bias. Use expectations are probably the single most powerful and pervasive source of learning bias in traditional machine learning systems. However, they have not been explicitly recognized as an important source of learning bias in the literature.

Use Expectations: The REACT model helps clarify how use expectations have been exploited by traditional machine learning systems to guide learning: the integration of new information need only consider how changes affect lines

of reasoning required for problem solving. This directive guides the design of methods for each of the three activities identified by the model:

1. Recognition: select only knowledge segments required to perform the anticipated application task.

2. Elaboration: pursue only lines of reasoning required by the application task.

3. Adaptation: affect only knowledge-base changes required by the application task.

Rather than exploring the consequences of new information for existing knowledge, a learning system dedicated to acquiring single-task knowledge simply ensures the new knowledge does not invalidate performance on a set of test cases [Wil88].

When the anticipated application task is cast as a search problem, techniques such as backward chaining or goal reduction guide the selection of operators so that only states known to be relevant to the goal condition are generated. These techniques address the problem of determining the appropiate grain size and focusing criteria confronting methods for performing recognition: the grain size is simply a single operator, and the principle of relevance is that of *subgoaling*. In subgoaling an operator is relevant to the goal condition when it establishes either some requisite (e.g., some conjunct) of the goal condition or some condition required by a relevant operator. [4]

[4]Additional focusing criteria is used to select a single operator from among those candidates that satisfy the criteria of subgoaling. Typically, this involves an ordering on operators in the knowledge base and an ordering of the preconditions required for each operator.

This makes clear one of the essential roles of use expectations in guiding learning: strong use expectations define an operational goal condition; the goal condition defines a principle of relevance (i.e., subgoaling); the principle of relevance solves the recognition problem and guides elaboration. For example, supervised concept learning systems receive new information comprising a set of pairs $(x_i\ f(x_i))$ (i.e., an example from the domain of the target function and the corresponding object from the function's range). This learning situation affords a simple and natural principle of relevance: existing rule $R(x_1..x_n)$ [5] is relevant if and only if either it establishes $f(x_i)$ (i.e., $f(x_i)$ is a consequence of $R(x_1..x_n)$) or it establishes some antecedent to a relevant rule. Such learning systems rely on an external mechanism to specify the goal condition; they consequently avoid, rather than solve, the problem of focusing attention during comprehension. Unfortunately, such learning systems remain incapable of acquiring knowledge to support any performance task other than responding to the goal query. Postponing commitment to use expectations by the methods that perform each learning subtask reduces the system's use-based brittleness (see Section A.1.5). However, this also requires determining an appropiate grain size and focusing criteria to use during recognition.

2.3 Summary

The importance of actively relating new information to prior knowledge is widely recognized by theories of cognitive and educational psychology: it is necessary for making sense of new information; it improves subsequent recall of new information; and it improves subsequent problem solving by ex-

[5] Notation explanation: $R(x_1..x_n)$ denotes a rule involving n terms.

panding the content of new information. However, care must be taken to encourage relating new information to prior knowledge in useful ways (i.e., those that promote learning goals rather than distracting the learner with a mass of irrelevant details). Furthermore, the cognitive resources available to relate new and prior knowledge is limited. Consequently, it is a selective and exclussive process: new information is related to only a chosen portion of the relevant prior knowledge.

REACT is proposed as a computational model of knowledge integration. It identifies a functional decomposition of knowledge integration into three activities that are individually necessary and collectively sufficient to perform knowledge integration, and each is an important source of learning bias. Recognition selects segments of prior knowledge to relate to new information. Elaboration embellishes new information and explores its consequences while establishing beliefs supported by new information and prior knowledge selected during recognition. Adaptation exploits learning opportunities detected among the beliefs established during elaboration. The next three chapters describe a machine learning program that implements the proposed model.

Chapter 3

KI: A Tool for Knowledge Integration

Implementing a computational model for a new task as a computer program has several advantages. It demonstrates the computational feasibility of the model, it facilitates empirical evaluations of the utility of performing the new task, and it defines strategies for solving the significant computational problems involved in performing the task. This and the next two chapters describe KI, an implementation of the REACT computational model for performing knowledge integration, and illustrate how KI performs the learning example in Figure 1.1. An empirical evaluation of KI follows in Chapter 6. KI builds on the approaches of many other researchers in machine learning and related disciplines; these will be described in Chapter 7.

3.1 Overview

KI is an interactive tool for knowledge integration. It was developed to help knowledge engineers extend the Botany Knowledge Base, a large-scale knowledge base representing plant anatomy, physiology, and development [PLM+88]. When a user provides new information, KI uses the existing knowledge base to identify possible gaps or conflicts and to identify beliefs supported by the new information. KI helps to verify that the *actual* effect of the new information accords with the knowledge engineer's *intended* effect. By posing questions and further knowledge-editing suggestions back to the knowledge

engineer, KI solicits additional information. Thus, KI provides a highly interactive knowledge-editing interface between the knowledge engineer and the knowledge base to guide knowledge-base development.

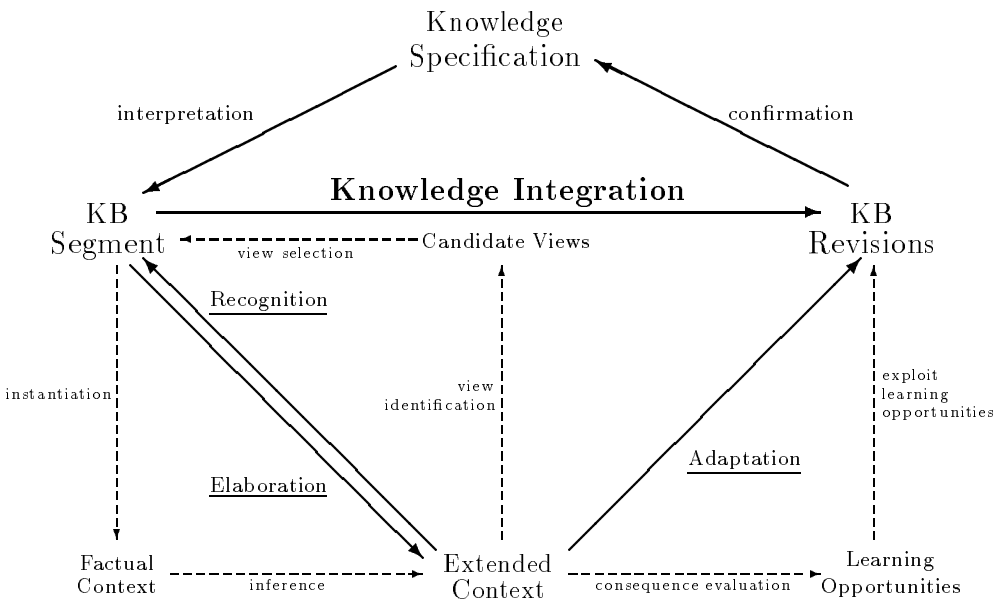
KI does more than identify "surface" gaps or conflicts (such as explicit constraint violations); it also determines indirect interactions between new information and existing knowledge. This involves a focused, best-first search to explore the consequences of new information and to detect learning opportunities.

The Botany Knowledge Base is not dedicated to a single, narrow application task; rather, it contains foundational, textbook knowledge intended to support a wide range of tasks in the domain of botany. Consequently, KI must adopt generic learning goals, such as promoting consistency, completeness, economy, and conviction (see Appendix B.2) as well as some of the general learning goals appropriate for this domain (e.g., relating plant anatomy to physiology).

KI implements REACT, the computational model for knowledge integration presented in Chapter 2, and so comprises three prominent activities:

1. Recognition: identifying existing knowledge relevant to new information.
2. Elaboration: applying relevant domain rules to determine the consequences of new information.
3. Adaptation: modifying new or prior knowledge to satisfy learning goals.

Furthermore, KI performs *interpretation*, which translates information from the input language (i.e., a machine readable form of the specification language)



Nodes denote data; arcs denote processes that transform the data. Knowledge integration decomposes into the activities of recognition, elaboration, and adaptation. Recognition decomposes into view identification and selection; elaboration decomposes into instantiating quantified formulae and completing inferences; and adaptation decomposes into a suit of methods that detect and exploit learning opportunities. Collectively, the arcs implement KI, except for *confirmation*, which is performed by the user.

Figure 3.1: The KI Architecture

into the internal representation language. Figure 3.1 presents the process architecture of KI.

During recognition, KI identifies beliefs existing in the knowledge base that are relevant to new information. KI uses *views* to determine which beliefs and concepts, beyond those explicitly referenced by the new information, are relevant [MP89]. Each view contains a set of beliefs that interact in some significant way and is indexed by concepts mentioned in the beliefs it contains. When new information is presented, KI identifies the views indexed by concepts

referenced by the new information and heuristically selects one. The beliefs contained in the selected view are deemed relevant to the new information, and KI focuses its search for learning opportunities on the interaction of the new information with the beliefs contained in the selected view.

During elaboration, KI investigates the consequences of new information for relevant beliefs in the knowledge base. This involves completing inferences and instantiating quantified formulae included in both the new information and those beliefs recalled during recognition. Elaboration "expands the content" of the new information by making explicit a partial entailment of the new information and relevant prior knowledge. Concepts referenced by beliefs contained within the selected view index other views relevant to this partial entailment; each of these views contains beliefs that could be used to extend the entailment. KI enters a cycle of recognition (i.e., selecting views) and elaboration (i.e., completing inferences to extend the partial entailment) to determine the consequences of the new information for relevant prior knowledge as it searches for learning opportunities.

During adaptation, KI detects and exploits learning opportunities suggested by both conflicts and novel explanations. Conflicts are revealed when inferences completed during elaboration establish inconsistent conclusions. KI responds by analyzing the explanations of inconsistent beliefs to identify knowledge-base modifications that resolve the conflict. Identifying and correcting conflicts during knowledge integration promotes consistency. Novel explanations are detected when the new information enables inferences. KI evaluates these explanations to identify new beliefs that, for example, generalize the new information or augment the representations of existing concepts. Extending the knowledge base with new beliefs promotes completeness (when

the beliefs are not entailed by existing knowledge), economy (when the beliefs are implicit consequences of existing knowledge), and conviction (when the beliefs entail existing knowledge). By detecting implicit learning opportunities and suggesting further knowledge-base modifications, KI assists the user in extending the knowledge base with new information.

The next section describes the knowledge-based system in which KI has been implemented. The following three sections describe the implementation of interpretation, elaboration and some aspects of adaptation while illustrating how KI begins the learning scenario presented in Figure 1.1.

3.2 The Botany Knowledge Base

The knowledge-representation language is an extended form of first-order predicate calculus [LG90]; it is typed, nonmonotonic, and, in a few special cases, second-order. The aspects of this language that are significant to KI are briefly discussed in this section.

3.2.1 Representing concepts

Four major categories partition the constants in this language:

1. individuals: particular and singular entities or events; e.g., *WinstonChurchill*, *AustinTexas*, *LindberghLandingInParis*, ...
2. attribute values: e.g., *Green*, *VeryHigh*, *Disabled*, *2*, ...
3. predicates: constants denoting relations; e.g., *color*, *density*, *cardinality*, ...
4. collections: sets of similar constants; e.g., *Politician*, *StateCapital*, *HistoricEvent*, *Color*, *BinaryPredicate*, ...

Many collection terms denote infinite sets; [1] e.g., *Leaf*, *Cell*, *Photosynthesis*. Consequently, the vast majority of elements for such sets are left implicit (i.e., there is no constant in the knowledge base corresponding to these individuals). Such elements are called *implicit concepts*.

The arguments of predicates are typed (i.e., constrained to be elements of a designated collection). For example, the binary predicate *physicalPart* denotes the physical decomposition of a tangible thing into its constituent tangible parts; both of its arguments are constrained to be elements of the collection *TangibleObject*. Furthermore, argument constraints can be collection-specific; e.g., when the first argument of *physicalPart* is an element of the collection *Plant*, then the second argument must be an element of the collection *BotanicalOrganismComponent*.

3.2.2 Ontology

The ontology of this knowledge base includes many standard representational distinctions. For example, the predicate that relates an individual to the collections of which it is an element is *isa*; e.g., $isa(AustinTexas\ StateCapital)$ denotes that Austin is a state capital. The ontology also makes a few novel distinctions, some of which will be mentioned in the forthcoming discussion of the example. These are briefly introduced in this section.

The ontology allows representing whether or not an event can occur and refering explicitly to the event in either case: $status(x\ Disabled)$ denotes that event x does not occur (i.e., that it has no temporal extent, no duration).

It is useful to be able to state that an individual has some property

[1] Or, more precisely, they denote sets that are convenient to consider infinite.

without stating the particular value of that property; the predicate *likelyFor* denotes this. For example,

$$[\forall\ (x)\ \neg translucency(x\ Transparent)\ \Rightarrow\ likelyFor(color\ x)]$$

denotes the belief that tangible objects that are not transparent probably have one or more colors. [2]

It is useful to represent explicitly some aspects of the way we think about the domain of botany, whether or not those aspects actually exist in the domain. Causation is an example: it is useful to be able to state that the occurrence of one event causes another event to occur. Whether or not causation actually exists in physical domains, it does often exist in the human study of those domains. The notion of a domain goal is another example. We often structure our understanding of biology by focusing on how the properties of an organism contribute to its survivability; that is, we consider surviving to be a "goal" of the organism (or species). Initially, the only goal asserted in the knowledge base is *hasPhysiologicalGoal(LivingObject health Facilitated)*; it is the goal of every living thing to facilitate its own good health. Furthermore, we often try to make sense of the plant's anatomy and physiology in terms of how goals are achieved. The ternary predicate $hasPhysiologicalFunction$ identifies those behaviors of an object that contribute to establishing one of its goals. For example, $hasPhysiologicalFunction(Leaf\ performs\ Photosynthesis)$ denotes that performing photosynthesis is a "function" of the leaf; it contributes to the goal of facilitating the leaf's (and the plant's) health.

[2] Applicable argument typing constraints are implicitly conjoined to the antecedents of rules; the first argument of *translucency* is constrained to be a tangible object.

3.2.3 Inference

Rules are typically first-order axioms with quantified variables. They are triggered only by ground propositions; i.e., general resolution is not supported. They can be directed to chain forwards or backwards.

Types of rules: KI distinguishes between two important types of rules:

1. skolemizing rules reference skolem functions
   (e.g., $\forall\,(x)\ isa(x\ Person)\ \Rightarrow\ \exists\,(y)\ isa(y\ FemalePerson)\ \&\ mother(x\ y)$)

2. non-skolemizing rules reference no functions and cannot introduce terms
   (e.g., $\forall\,(x)\ isa(x\ Leaf)\ \Rightarrow\ color(x\ Green)$)

Skolemizing rules are significant because they introduce terms (i.e., they make implicit concepts explicit). [3] Distinguishing between these two classes of rules facilitates KI's method for controlling inference during elaboration. (This method will be presented in the next chapter.)

Nonmonotonicity: Default reasoning is important to knowledge-based systems as a method for coping with the qualification problem [McC77]: default beliefs can be based on assumptions that are usually true but are not easily verified. Default reasoning is important to learning systems as a guide for coping with conceptual conflict: default beliefs are based on assumptions that are not always warranted. When inconsistent beliefs are established with the support

[3] There are no function symbols in the language; the only function terms come from turning existentially quantified variables into skolem functions. Skolem functions are currently the only means by which the system automatically creates new constants (i.e., makes implicit concepts explicit).

of default beliefs, the assumptions underlying those beliefs suggest conditions that, if asserted, would retract the beliefs and resolve the conflict.

Rules in the knowledge base are associated with one of two degrees of certainty: some are default (i.e., nonmonotonic); others are absolute (i.e., monotonic). Furthermore, the representation language includes the operator *unless* to provide nonmonotonic inference based on the closed world assumption:

$$x \Leftarrow y \ \& \ unless(z)$$

denotes that when y can be established and $\neg z$ can be established under the closed world assumption (i.e., with negation as failure) then conclude that x is established. This scheme makes explicit the assumptions underlying default beliefs (e.g., $\neg z$ is an assumption underlying the inferred belief x). Consequently, when a conflict results from nonmonotonic inference, the underlying assumptions can be identified by searching the inference graphs.

Rule macros: For rules that share a very common syntactic form, it is convenient to define predicates that behave as *rule macros* and expand into rule forms. [4] Some examples are:

1. ako: $\forall\ (xy)\ [ako(x\ y)] \Leftrightarrow [\forall\ (z)\ isa(z\ x) \Rightarrow isa(z\ y)]$; e.g., $ako(OakTree\ Tree)$
2. inherits: $\forall\ (xyp_1p_2...p_n)\ [inherits(x\ (p_1\ p_2\ ...\ p_n)\ y)]$
   $\Leftrightarrow [\forall\ (z_1z_2...z_n)\ isa(z_1\ x)\ \&\ p_1(z_1\ z_2)\ \&\ p_2(z_2\ z_3)\ \&\ ...\ p_{n-1}(z_{n-1}\ z_n) \Rightarrow p_n(z_n\ y)]$
   (note: each p_i denotes a binary predicate); e.g., $inherits(Leaf\ (element\ color)\ Green)$.
3. relationType: $\forall\ (xyz)\ [relationType(x\ y\ z)]$
   $\Leftrightarrow [\forall\ (x_1)\ isa(x_1\ x) \Rightarrow \exists\ (z_1)\ isa(z_1\ z)\ \&\ y(x_1\ z_1)]$;
   e.g., $relationType(Person\ mother\ FemalePerson)$
4. akoAttribute: $\forall\ (xy)\ [akoAttribute(x\ y)] \Leftrightarrow [\forall\ (pz)\ p(z\ x) \Rightarrow p(z\ y)]$
   (note: p denotes a binary predicate); e.g., $akoAttribute(Indigo\ Blue)$

[4] Predicates denoting relations in the domain are binary. However, predicates that act as rule macros can have higher arities.

5. akoSlot: $\forall\ (p_1 p_2)\ [akoSlot(p_1\ p_2)] \Leftrightarrow [\forall\ (xy)\ p_1(x\ y) \Rightarrow p_2(x\ y)]$
(note: p_1 and p_2 denote binary predicates); e.g., $akoSlot(mother\ parent)$
6. inverseSlot: $\forall\ (p_1 p_2)\ [inverseSlot(p_1\ p_2)] \Leftrightarrow [\forall\ (xy)\ p_1(x\ y) \Rightarrow p_2(y\ x)]$
(note: p_1 and p_2 denote binary predicates); e.g., $inverseSlot(part\ partOf)$
7. argumentOneType: $\forall\ (px)\ [argumentOneType(p\ x)] \Leftrightarrow [\forall\ (x_1 z)\ p(x_1\ z) \Rightarrow isa(x_1\ x)]$
(note: p denotes a binary predicate; there is a similar predicate for argument two); e.g., $argumentOneType(physicalPart\ TangibleObject)$
8. classArgTwoType: $\forall\ (pxy)\ [classArgTwoType(x\ p\ y)]$
$\Leftrightarrow [\forall\ (x_1 y_1)\ isa(x_1\ x)\ \&\ p(x_1\ y_1) \Rightarrow isa(y_1\ y)]$ (note: p denotes a binary predicate); e.g., $classArgTwoType(Plant\ physicalPart\ BotanicalOrganismComponent)$
9. likelyForType: $\forall\ (px)\ [likelyForType(p\ x)] \Leftrightarrow [\forall\ (x_1)\ isa(x_1\ x) \Rightarrow \exists\ (y)\ p(x_1\ y)]$
(note: p denotes a binary predicate)

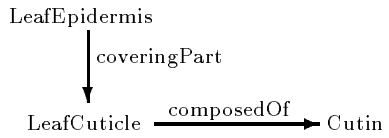
Rule macros are important to knowledge-based systems because they permit defining efficient special-case inference implementations, meta-reasoning methods (e.g., to detect conflicts and subsumption), and editing methods (e.g., to specify and present knowledge) for these very common inference patterns [LG90, Der90]. They are important to learning systems because they suggest one criterion for when knowledge compilation should occur: compile an inference path into a new shallow rule if the resulting rule can be expressed as a rule macro.

The remainder of this and the following two chapters describe in detail how KI performs the learning example presented in Figure 1.1.

3.3 Interpreting new information

Figure 3.2 presents the new information for the example as it is specified by knowledge engineers, provided as input to KI, and interpreted by KI. New information is provided to KI as a semantic network (in the form of nested lists) since knowledge engineers developing the knowledge base adopted semantic networks as their knowledge specification language [PLM+88]. The nodes and arcs of the networks correspond to knowledge-base constants.

(a) The new information stated in the specification language



(b) The new information stated in the input language

(LeafEpidermis (coveringPart (LeafCuticle (composedOf (Cutin)))))

(c) The interpretation of the new information

Rule A : *Each leaf epidermis has a leaf cuticle as a covering part.*
[∀ (x) isa(x LeafEpidermis) ⇒ ∃ (y) isa(y LeafCuticle) & coveringPart(x y)]

Rule B : *Leaf epidermises have only leaf cuticles as covering parts.*
[∀ (xy) isa(x LeafEpidermis) & coveringPart(x y) ⇒ isa(y LeafCuticle)]

Rule C : *Each leaf cuticle is a covering part of a leaf epidermis.*
[∀ (x) isa(x LeafCuticle) ⇒ ∃ (y) isa(y LeafEpidermis) & coveringPart(y x)]

Rule D : *Leaf cuticles are covering parts of only leaf epidermises.*
[∀ (xy) isa(x LeafCuticle) & coveringPart(y x) ⇒ isa(y LeafEpidermis)]

Rule E : *Each leaf cuticle is composed of cutin.*
[∀ (x) isa(x LeafCuticle) ⇒ composedOf(x Cutin)]

Rule F : *Leaf cuticles are components of botanical organisms.*
[∀ (x) isa(x LeafCuticle) ⇒ isa(x BotanicalOrganismComponent)]

Fact G : *The class of leaf cuticles is a type of tangible object.*
isa(LeafCuticle TangibleObjectTye)

The new information **(a)** conceived as a semantic network encoded graphically (i.e., in the specification language), **(b)** presented as a semantic network encoded as nested lists (i.e., in the input language), and **(c)** interpreted as first-order axioms (i.e., in the representation language).

Figure 3.2: Interpreting the new information

Interpretation translates new information expressed in the specification language into the internal language of knowledge representation. A detailed presentation of KI's methods for interpreting semantic networks is included in Appendix C.

One of the important subtasks of interpretation is handling new symbols (i.e., a node or arc not corresponding to any existing knowledge-base constant). This involves both detecting new symbols denoting domain concepts

not previously represented in the knowledge base and determining preliminary properties (e.g., taxonomic properties) of the new concepts.

When new symbols are detected in the new information KI (optionally) requests confirmation that the user intends to introduce a new constant to the knowledge base. This prevents the generation of spurious new symbols in the KB resulting from, for example, typing mistakes or false assumptions about the precise formal symbols used in the knowledge base to denote a particular domain concepts (e.g., the formal symbol in the knowledge base denoting the collection of plant flowers might be *Flower, Flowers, PlantFlower, BotanicalFlower, Fleur, Gensym022, ...*).

Interpretation also determines some preliminary properties of new concepts. This includes preliminary assertions about the argument types appropriate for new predicates and the existing collections that have new individuals as elements and new collections as subsets. In the example, new information introduces the constant *LeafCuticle*. Argument typing constraints applicable to the predicates that reference *LeafCuticle* requires that it subset *BotanicalOrganismComponent*.

Interpretation in KI relies on heuristic and defeasible strategies: the semantic networks that reference a new concept may not provide sufficient information to determine precisely where in the subsumption hierarchy that new concept belongs (e.g., it is often ambiguous whether a new term denotes an individual or a collection of individuals).

Interpretation translates the new information from the input language into axioms stated in the representation language of the knowledge base. However, the consequences of resulting axioms remain implicit: elaboration investigates how these new axioms interact with prior knowledge.

3.4 Elaboration: determining the consequences of new information

During elaboration, KI determines how new information interacts with existing knowledge. This involves maintaining a *learning context* comprising only beliefs deemed relevant to the new information. Initially, only the new information is included in the learning context. Elaboration then extends the learning context with a partial entailment of new and relevant prior knowledge.

3.4.1 Initializing the learning context

KI initializes the learning context with the new information, which often includes quantified formulae. Since inference in the knowledge base is constrained to reason only with ground propositions, rules appearing in the new information will not directly unify with existing knowledge-base rules. Consequently, KI must operationalize new rules by creating a set of ground propositions that satisfy the rules' antecedents (Figure 3.3). Each variable appearing in the new information is bound to a constant denoting a hypothetical instance of the collection over which that variable may range.

There are three reasons for restricting the learning context to ground propositions:

1. The inference engine of the knowledge base permits only ground propositions to trigger rules: quantified formulae can only unify with ground formulae, not with other quantified formulae. Since inference performed during learning should be representative of the inference capabilities that will support problem solving, elaboration is subjected to this same restriction.

(a) The new information

[∀ (x) isa(x LeafEpidermis) ⇒ ∃ (y) isa(y LeafCuticle) & coveringPart(x y) & composedOf(y Cutin)]

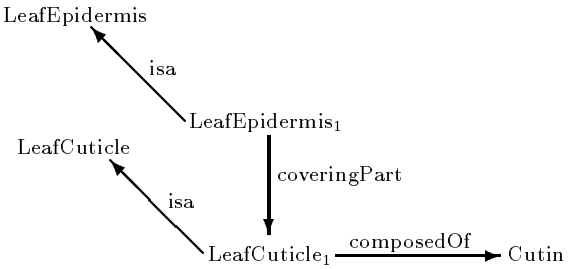
(b) The hypothetical concepts

{LeafEpidermis$_1$
LeafCuticle$_1$}

(c) The initial learning context

{isa(LeafEpidermis$_1$ LeafEpidermis)
isa(LeafCuticle$_1$ LeafCuticle)
coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)
composedOf(LeafCuticle$_1$ Cutin)}

(d) The learning context as a semantic network



(a) New information provided to KI specifying that leaf epidermises are covered by leaf cuticles composed of cutin. **(b)** KI instantiates constants denoting hypothetical instances of the collections over which variables appearing in the new information may range. **(c)** KI initializes the learning context with a set of ground propositions that satisfy the new information. **(d)** The contents of the learning context depicted graphically as a semantic network. Numerical subscripts denote class membership (e.g., $isa(LeafEpidermis_1\ LeafEpidermis)$); henceforth these *isa* links will not be shown.

Figure 3.3: Initializing the Learning Context

2. Reasoning with specific instances is more natural for humans than reasoning with general concepts and rules (e.g., syllogisms) [JLB84]. Our understanding of a description often feels incomplete until we can imagine a concrete example of what is described [GH86]. By performing inference with ground propositions, KI produces a concrete (i.e., instantiated) representation of the new information and its consequences. Traces of the system's inference engine using this representation are more meaningful to the user/teacher than, for example, traces involving general resolution. Consequently, the representation aids the user/teacher both in compre-

hending the state of the system/learner's knowledge and in detecting and correcting faulty inference during learning.

3. Inference can be controlled by restricting the set of constants available for binding to variables. Also, since constants directly denote domain concepts and variables do not, selecting which constants to make available for binding during inference can be guided by domain knowledge. This strategy for controlling inference by restricting the constants available for binding to variables is fundamental to KI and will be fully explained in the next chapter.

3.4.2 Extending the learning context via inference

After the learning context has been initialized with the new information and new rules have been operationalized as sets of ground propositions, elaboration makes explicit a partial entailment of new and prior knowledge by permitting the non-skolemizing rules in the knowledge base to exhaustively forward-chain. Thus inference extends the learning context with consequences of the new information. Figures 3.4 and 3.5 illustrate how inference in the learning context reveals some of the implicit consequences of the new information. Figure 3.4 shows some of the rules that are triggered as the ground propositions that initialize the learning context are asserted. Figure 3.5 shows the learning context after being extended by inference.

The task of completing inference in the learning context is described formally in Figure 3.6. As each ground proposition p is added to the context, every non-skolemizing rule triggered by any set of ground propositions in the knowledge base that necessarily includes p is fired. To avoid completing inferences that are completely independent of the learning context (and irrelevant

Rule 1 : *Epidermis is a type of container.*
[∀ (x) isa(x Epidermis) ⇒ isa(x Container)]

Rule 2 : *The coverings of containers are themselves containers.*
[∀ (xy) isa(x Container) & coveringPart(x y) ⇒ isa(x Container)]

Rule 3 : *Containers are solid.*
[∀ (x) isa(x Container) ⇒ isa(x Solid)]

Rule 4 : *Solid objects are opaque.*
[∀ (x) isa(x Solid) ⇒ transparency(x Opaque)]

Rule 5 : *Homogeneous composition suggests class membership.*
[∀ (xy) composedOf(x y) & unless(∃ (z) z≠y & composedOf(x z)) ⇒ isa(x y)]

Rule 5a : *Class membership in a composition type suggests composition.*
[∀ (xy) isa(x y) & isa(y CompositionType) ⇒ composedOf(x y)]

Rule 6 : *Cutin is impermeable to gases.*
[∀ (x) isa(x Cutin) ⇒ impermeableToType(x Gas)]

Rule 7 : *Cutin is impermeable to liquids.*
[∀ (x) isa(x Cutin) ⇒ impermeableToType(x Liquid)]

Rule 8 : *Covering parts cover.*
[∀ (xy) coveringPart(x y) & unless(partiallyCovers(y x)) ⇒ covers(y x)]

Rule 9 : *Impermeable coverings suggest impermeability.*
[∀ (xyz) covers(x y) & impermeableToType(x z)
& unless(∃ (w) portal(y w) & ¬covers(x w)) ⇒ impermeableToType(y z)]

Some of the non-skolemizing rules triggered by ground beliefs asserted as the learning context is initialized.

Figure 3.4: Rules triggered in the learning context

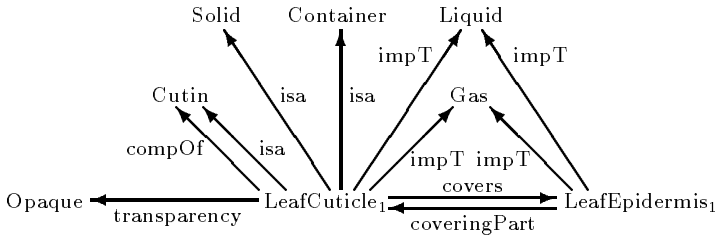
to the new information) the set of propositions that trigger a rule must include at least one proposition contained in the learning context. All inferences completed during elaboration are consequences of (i.e., supported by), and therefore relevant to, the contents of the learning context, which is initialized with the new information. Thus, during elaboration, a partial entailment of the new information is completed, and only inferences relevant to the new information are performed.

It is significant that only non-skolemizing rules are allowed to chain exhaustively. Without this restriction, infinite computations might be attempted. A standard example illustrates this:

(a) The extended learning context

{isa(LeafEpidermis$_1$ LeafEpidermis)	*isa(LeafCuticle$_1$ Cutin)*
isa(LeafCuticle$_1$ LeafCuticle)	*isa(LeafCuticle$_1$ Container)*
coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)	*isa(LeafCuticle$_1$ Solid)*
composedOf(LeafCuticle$_1$ Cutin)	*transparency(LeafCuticle$_1$ Opaque)*
	covers(LeafCuticle$_1$ LeafEpidermis$_1$)
	impermeableToType(LeafCuticle$_1$ Gas)
	impermeableToType(LeafCuticle$_1$ Liquid)
	impermeableToType(LeafEpidermis$_1$ Gas)
	impermeableToType(LeafEpidermis$_1$ Liquid)}

(b) The learning context as a semantic network



Abbreviations: impT : impermeableToType
compOf : composedOf

(a) The learning context extended with inferred consequences of the new information. Inferred facts are presented in italics, and inferred facts that are not consequences of the new information (e.g., isa(LeafEpidermis$_1$ Container) are omitted. **(b)** The consequences depicted graphically as a semantic network.

Figure 3.5: The learning context extended through inference

1. $\forall\,(x)\ isa(x\ Person)\ \Rightarrow\ \exists\,(y)\ isa(y\ FemalePerson)\ \&\ mother(x\ y)$
2. $\forall\,(x)\ isa(x\ FemalePerson)\ \Rightarrow isa(x\ Person)$

chains endlessly once triggered as an infinite number of implicit female ancestors are created and made explicit. Limiting exhaustive chaining during elaboration to non-skolemizing rules precludes such infinite computations.

Given:
1) Δ_n: non-skolemizing beliefs of the knowledge base
2) Φ: the learning context (i.e., a set of facts)

Find: Φ': an extended learning context that includes the partial entailment of $(\Delta_n \cup \Phi)$, defined as $\{p \mid \exists(\Psi)(\Psi \subseteq \Delta_n) \,\&\, \neg(\Psi \vdash p) \,\&\, (\Psi + \Phi \vdash p)\}$

Figure 3.6: Inference in the learning context

3.4.3 Distinguishing consequences of new information

While many inferences are completed during elaboration, not all are consequences of the new information. For example, one of the propositions used to initialize the learning context, $isa(LeafEpidermis_1\ LeafEpidermis)$, triggers many rules (e.g., inheritance, taxonomic subsumption) that add new facts to the learning context (e.g., $isa(LeafEpidermis_1\ TangibleObject)$). These new facts may turn out to be relevant to the new information, but they are not consequences of it (i.e., that a leaf epidermis is tangible is not a consequence of its having a cuticle). There are two advantages in distinguishing the consequences of the new information from the other inferred facts:

1. Guiding subsequent elaboration: consequences of new information are more relevant to the new information and tend to be more interesting and useful to the user. Distinguishing the consequences from the non-consequences enables KI to guide subsequent elaboration in directions that deepen the inference paths of consequences. This bias promotes the relevance and utility of subsequent elaboration.

2. Guiding interactions with the teacher/user: consequences of new information are presented separately to the user. As Chapter 1 discusses, observing the consequences enables the user to better perceive the actual

(vs. intended) effects of adding the new information to the knowledge base. Furthermore, elaboration can populate the learning context with a plethora of inferred facts; reporting all of them to the user would be overwhelming. Distinguishing the consequences from the nonconsequences enables KI to focus the user's attention on the more relevant and useful results of elaboration.

Therefore, during elaboration, KI distinguishes between those facts that are and those that are not consequences of the new information.

By construction, the initialization of the learning context will necessarily include some consequences: if the new information includes ground propositions, then those propositions are consequences; if the new information includes rules, then the ground propositions that instantiate the right-hand sides of the rules are consequences. In the example, the initialization of the learning context includes four propositions (Figure 3.3c), three of which are consequences:

{isa(LeafCuticle$_1$ LeafCuticle)
coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)
composedOf(LeafCuticle$_1$ Cutin)}

The inferred facts that are consequences of new information in the example appear in italics in Figure 3.5a.

3.4.4 Maintaining justifications of inferred facts

As facts are inferred, the underlying knowledge base updates a truth maintenance system (TMS) that records all inferential dependencies involved in establishing each fact. However, during elaboration, KI also maintains a

separate record of inferential dependencies. The former records will be referred to as the *TMS level*, the latter as the *explanation level*.

The two levels support different capabilities and have different requirements. The TMS level is charged with managing inferential dependencies used to implement nonmonotonic inference: when an established fact violates an assumption, the TMS uses the recorded inferential dependencies to determine which facts must be retracted because they rely on the violated assumption. For this purpose, the TMS level must include every inferential dependency; that is, it must include every inference path that establishes an inferred fact. However, search at the TMS level is quite focused: only the inferential dependencies involving particular facts (e.g., ones that are being retracted) need be inspected. In general, the TMS level is not searched extensively.

The explanation level is charged with managing inferential dependencies used to detect and exploit learning opportunities during adaptation. (Some of these will be discussed in the following section.) For this purpose, the explanation level is more extensively searched but need include only those inferential dependencies that may participate in learning opportunities. Consequently, many inference paths at the TMS level can be omitted from the explanation level.

Intuitively, distinct inference paths at the explanation level should correspond to distinct reasons, hypotheses, or phenomena in the domain. If several TMS-level inference paths establishing a common fact are considered (e.g., by a human) to provide essentially the same rationale for why that fact is believed, then these inference paths form an equivalence class, and only one would be included in the explanation level. Therefore, the explanation level comprises a subset of the TMS level; it is an abstraction of the TMS level that

omits details not required for detecting and exploiting learning opportunities.

There are two heuristics implemented in KI for separating the explanation level from the TMS level:

1. ignore differences among inference paths due to *inverseSlot* inferences
2. ignore differences among inference paths due to *akoSlot* inferences

The first heuristic is warranted because slot [5] inverses are artifacts of the representation language and are not conceptually significant in the domains represented. For example, $color(Leaf_1\ Green)$ and $colorOf(Green\ Leaf_1)$ are different formal denotations of the same domain belief. Consequently, inference paths that differ only because of slot inverse inferences do not correspond to distinct reasons or hypotheses.

The second heuristic is warranted because inferences based on *akoSlot* are obvious and uninformative. Many slots are artifacts of the representation language and fail to denote any significant distinction in the domain. For example, $akoSlot(superset\ ako)$ is asserted since *ako* is simply the transitive closure of the *superset* relation; two inference paths that differ only because one references $superset(Leaf\ BotanicalOrgan)$ and the other references $ako(Leaf\ BotanicalOrgan)$ make no distinction that is useful during adaptation. It is neither interesting nor conceptually significant in the domain to distinguish among the different paths through the slot generalization hierarchy when analyzing or reporting the inference paths that establish a fact. Furthermore, both *inverseSlot* inferences and *akoSlot* inferences are considered definitional and monotonic; they do

[5] A *slot* is a binary predicate.

not introduce unwarranted support for erroneous or inconsistent facts. Consequently, KI need not suspect them when resolving inconsistencies.

Figure 3.7 illustrates the distinction between the TMS level and the explanation level with an example. A TMS-level inference graph comprising four explanations of the fact that the leaf cuticle is impermeable to gas is compressed to an explanation-level inference graph comprising a single explanation. [6] By identifying equivalence classes of inference paths at the TMS-level and collapsing each class into a single explanation at the explanation level, KI drastically reduces the number of explanations it must search for learning opportunities during adaptation.

3.5 Adaptation: detecting and exploiting learning opportunities

During adaptation, KI appraises those inferences completed during elaboration and assists the user in modifying the knowledge base to accommodate the consequences of the new information. This assistance often takes the form of suggestions for further knowledge-base editing, such as retracting or modifying existing beliefs, adding new beliefs, and soliciting additional knowledge from the user. Alternatively, it can involve autonomous modifications of the knowledge base, each accompanied by a notification that both explicitly asks whether the modification is appropriate and implicitly offers the option to undo the modification.

Performing adaptation requires detecting and exploiting learning opportunities that arise during elaboration and recognition. Each of the various

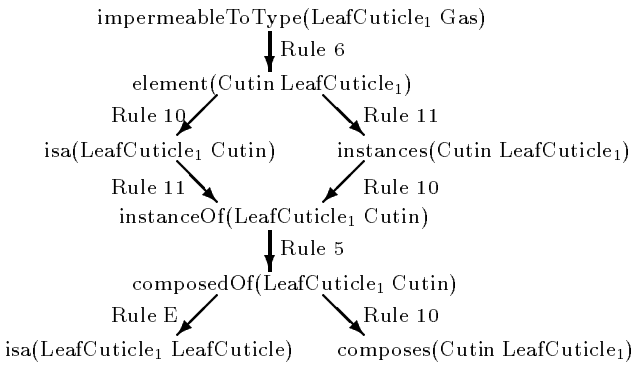
[6] Figure 3.7 does not include all the inference paths that establish this fact: there actually are ten distinct TMS-level inference paths and four explanation-level inference paths.

(a) Slot inverse and generalization rules

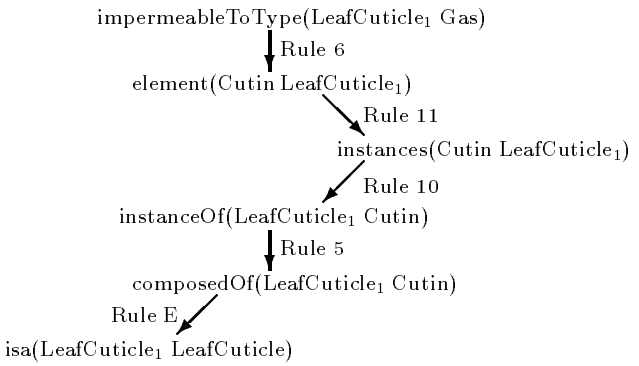
Rule 10 : *inverseSlot rule.*
$[\forall\ (xyp_1p_2)\ p_1(x\ y)\ \&\ inverseSlot(p_1\ p_2)\ \Rightarrow\ p_2(y\ x)]$

Rule 11 : *akoSlot rule.*
$[\forall\ (xyp_1p_2)\ p_1(x\ y)\ \&\ ako(p_1\ p_2)\ \Rightarrow\ p_2(x\ y)]$

(b) A TMS-level inference graph



(c) An explanation-level inference graph



(a) Rules that support *inverseSlot* and *akoSlot* inferences. **(b)** An inference graph at the TMS level comprising four distinct inference paths that establish the fact that the hypothetical leaf cuticle is impermeable to gas. **(c)** An inference graph at the explanation level comprising one distinct inference path that establishes the same fact. By collapsing multiple TMS-level paths into fewer explanation-level paths, KI drastically reduces the number of explanations it must search for learning opportunities during adaptation.

Figure 3.7: The TMS level vs. the explanation level

learning opportunities acquires different types of knowledge and provides different types of knowledge-base improvements; each reflects a very different learning heuristic. Consequently, adaptation in KI exemplifies multi-strategy learning. [Mic94].

The adaptation methods for each learning opportunity comprise methods to detect the learning opportunity as well as methods to exploit it. While the latter are fairly independent, the former are each triggered by the status and contents of the learning context, so the implementation is conceptually similar to a blackboard architecture [EL75]. In the example, elaboration of the instantiated training reveals several learning opportunities. The following three sections describe those opportunities that involve resolving inconsistencies, compiling explanations into new rules, and abductively refining new rules.

3.5.1 Resolving inconsistencies

Large knowledge bases are prone to internal inconsistencies; resolving them promotes the general learning goal of consistency. One of the most important design features of KI is that it "exercises" the knowledge base; elaboration uses explicit knowledge and the inference engine to explore regions of implicit knowledge. As implicit beliefs become explicit, tacit inconsistencies in the knowledge base are revealed, and each offers the learning opportunity of resolving it.

In the example, elaboration establishes that the leaf cuticle covers the leaf epidermis (Figure 3.5). However, this belief conflicts with the argument typing constraints defined for *covers*; specifically, the second argument is required to be an element of *SheetOfStuff*, tangible objects having sheet-like dimensions (i.e., flat, significant in precisely two dimensions) that are also ho-

mogeneous (i.e., they can be thought of as unstructured, having no discernible physical parts without changing grain size).

KI relies on the underlying inference engine to detect internal inconsistencies. [7] The system detects about a dozen different types of inconsistency, each involving a constraint and a fact. When an inconsistency is detected KI identifies knowledge-base modifications to resolve the conflict (Figure 3.8) using three basic strategies:

1. Analyze the justification of the fact to determine what knowledge-base modifications would cause the fact to be retracted.

2. Analyze the scope and justification of the constraint to determine what knowledge-base modifications would cause the constraint to be retracted or made inapplicable to the fact.

3. Analyze how the fact fails the constraint to determine what additional beliefs would enable the fact to satisfy the constraint.

Each of these strategies can identify plausible knowledge-base modifications to resolve the conflict, and KI pursues all three whenever an inconsistency is encountered. The cumulative results of these pursuits are included in a memo to the user suggesting how to resolve the inconsistency (Figure 3.8e). Each of these strategies is illustrated in turn using the example.

Determining how to refute a fact: The inference paths that established the fact are analyzed for essential support. A set of beliefs provide *essential support*

[7] KI actually intercepts attempts by the system to suspend execution of the inference engine and invoke the debugger.

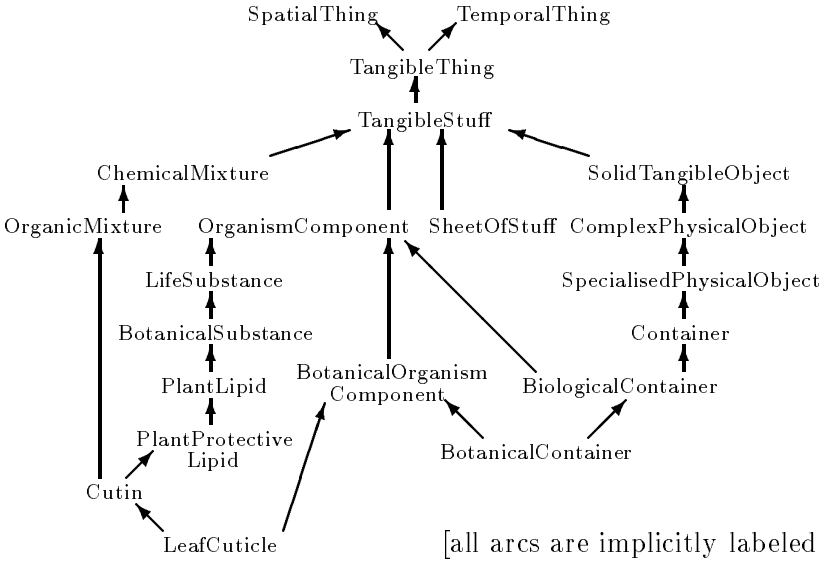
(a) An inconsistency: a fact and a violated constraint

covers(LeafCuticle$_1$ LeafEpidermis$_1$) argumentOneType(covers SheetOfStuff)

(b) An explanation and assumptions establishing the fact

$covers(LeafCuticle_1\ LeafEpidermis_1)$
$\Leftarrow_8\ coveringPart(LeafEpidermis_1\ LeafCuticle_1)$
$\Leftarrow_A\ isa(LeafEpidermis_1\ LeafEpidermis)$
$unless(partiallyCovers(LeafCuticle_1\ LeafEpidermis_1)$

(c) A portion of the subsumption hierarchy



The minimal generalization of LeafCuticle and SheetOfStuff is TangibleStuff.

(d) The condition imposed by the constraint

[covers(LeafCuticle$_1$ LeafEpidermis$_1$) $\Rightarrow$ isa(LeafCuticle$_1$ SheetOfStuff)]

(e) Suggested knowledge-base revisions

[**:assert** relationType(LeafCuticle LeafEpidermis partiallyCovers)]
[**:retract** argumentOneType(covers SheetOfStuff)]
[**:assert** argumentOneType(covers TangibleStuff)]
[**:assert** ako(LeafCuticle SheetOfStuff)]

(a) An inconsistency: a fact and an applicable constraint not satisfied by the fact. **(b)** An explanation establishing the fact reveals an underlying assumptions. (Rule 8 includes an assumption.) Only one explanation exists for the fact, so the assumption provides essential support of the fact. Notation: $p \Leftarrow_n q$ denotes that p follows from rule n triggered by q, and vertical alignment of antecedents denotes conjunction. **(c)** A portion of the knowledge-base subsumption hierarchy. **(d)** The condition imposed by the violated constraint. **(e)** The knowledge-base revisions suggested to the user: asserting (an abstraction of) a refutation of the assumption made by the explanation of the fact; retracting the constraint; asserting a minimally weakened constraint that admits the fact; asserting that (an abstraction of) the condition imposed by the constraint is satisfied.

Figure 3.8: Resolving an inconsistency

if they participate in every explanation of the fact. The retraction of essential support for an inferred fact refutes every inference path that establishes the fact, causing the fact to also be retracted.

The explanations of a fact can include any number of alternative essential supports, so a preference criteria is adopted to select among the alternatives. Each candidate essential support suggests beliefs that can be retracted to resolve the inconsistency. The preference among alternative essential supports is therefore based inversely on the conviction of the beliefs contained in the supports:

1. refuting explicit assumptions (e.g., as identified by *unless* clauses) is preferred to refuting nonmonotonic facts, and

2. refuting nonmonotonic facts is preferred to refuting monotonic facts

Consequently, KI first searches the explanations for assumptions that constitute essential support. If this search fails, KI next searches the explanations for essential nonmonotonic facts (i.e., those established by nonmonotonic rules). If this search fails as well, KI then searches the explanations for any kind of essential facts (e.g., ones established by either monotonic rules or directly by a knowledge engineer). This last search is guaranteed to succeed: the fact participating in the inconsistency itself constitutes essential support either established by some rule or directly asserted by some knowledge engineer. Guiding the search for candidate facts to refute using a preference based inversely on some notion of the strength or utility of the refutation candidates is common to other approaches to resolving inconsistencies in knowledge bases (e.g., [GP94, DW93]).

While searching for essential support, KI maintains a "single fix" preference: some support (e.g., assumption or inferred belief) will participate in every explanation and so constitute essential support. KI thus avoids having to search for preferred combinations of inessential support from various explanations in order to construct a set of beliefs that collectively provide essential support. [8] Alternative approaches to resolving inconsistencies in knowledge bases which do not permit refuting a fact (e.g., a positive training instance) generally cannot adopt a single fix assumption and rely on a greedy algorithm to heuristically search the combinations of inessential support for refutation candidates that collectively provide essential support (e.g., [OM90]).

The knowledge-base modifications of refuting each essential support retrieved by this search are included in a memo to the user that suggests how to resolve the inconsistency. In the example, the explanations of $covers(LeafCuticle_1$ $LeafEpidermis_1)$ include the essential assumption $unless(partiallyCovers(LeafCuticle_1$ $LeafEpidermis_1))$ (Figure 3.8b). Simply asserting $partiallyCovers(LeafCuticle_1$ $LeafEpidermis_1)$ would resolve this particular inconsistency, but its effects would be local to the learning context and would not resolve the tacit conflict among the rules of the knowledge base. In proposing knowledge-base modifications to the user, KI must abstract from particular terms used in its local analysis within the learning context to terms that are referenced by the enduring beliefs in the knowledge base. The abstraction is formed by replacing, those hypothetical instances referenced by the proposition with the collections they instantiate (e.g., $partiallyCovers(LeafCuticle\ LeafEpidermis)$), and then interpreting the resulting figurative proposition. [9] The suggestion

[8] The inessential support of fact p includes every fact that is referenced by some, but not all, explanations that establish p.

[9] A proposition is *figurative* when one or more of its arguments are a specialization, rather

[:assert $relationType(LeafCuticle\ partiallyCovers\ LeafEpidermis)$]

if executed, resolves the tacit conflict in the knowledge base, and KI includes this suggestion in the memo to the user.

Determining how to refute a constraint: The justifications of the constraint, if there are any, are analyzed for assumptions. If assumptions are found, then knowledge-base modifications to refute them are included in the memo to the user that suggests how to resolve the inconsistency. If there are no assumptions underlying the constraint, the knowledge-base modification of retracting the constraint itself is included in the memo. KI further identifies alternative *minimal modifications* of the constraint that admit the fact. The knowledge-base modifications to assert these revised constraints are also included in the memo.

The type of minimal modification appropriate for admitting the fact depends on the type of constraint. For an argument typing constraint, the appropriate minimal modification is a *minimal generalization* of the argument type (e.g., *SheetOfStuff*) and of the collections instantiated by the argument (e.g., *LeafCuticle*). A minimal generalization of two collections is an existing, reified collection that is a superset of each of the two collections but is not itself a proper superset of any other superset of the two collections. In the example, the minimal generalization of *LeafCuticle* and *SheetOfStuff* is *TangibleStuff* (Figure 3.8c). Therefore, the suggestion [:assert $argumentOneType(coversObject\ TangibleStuff)$] is included in the memo.

---

than an element, of the applicable argument types. Interpretation and figurative references are discussed in Appendix C.

Determining how to appease a violated constraint: Constraints impose conditions that must be true of the beliefs to which they apply. The third strategy KI pursues for resolving inconsistencies involves identifying what conditions are being imposed by the constraint, and then determining what additional beliefs might satisfy those conditions.

Different constraints impose different types of conditions on beliefs. For example, an argument-typing constraint requires that an argument of a predicate be an element of a specified collection. In the example, the fact fails the constraint because the fact $isa(LeafCuticle_1\ SheetOfStuff)$ has not been established (Figure 3.8d). Simply asserting this fact would resolve the inconsistency in the learning context, but, as before, the suggestion memos must include knowledge-base modifications that resolve the tacit conflict among rules in the knowledge base. KI abstracts this fact so that it no longer references terms specific to the learning context, and the suggestion

[:assert $ako(LeafCuticle\ SheetOfStuff)$]

is included in the memo.

This last suggestion is valid in the domain and will eventually be accepted by the user. Thus, while resolving an inconsistency, KI has uncovered a gap in the knowledge base and adbuctively filled it with new taxonomic knowledge that extends the new information.

3.5.2 Inference-path compilation

An important and ubiquitous learning opportunity occurs when a useful but deep inference path is compiled into a shallow and efficient rule [MKKC86, DM86, Die86]; the new rule's antecedent identifies the weakest pre-

conditions [Dij75] for completing the inference path. Acquiring such compilations can promote the general learning goal of economy as useful implicit beliefs are made more accessible.

At this point in the example, the three types of knowledge KI acquires by exploiting this learning opportunity are new inheritance (e.g., *inherits*) rules, new taxonomic (e.g., *ako*) rules, and new skolemizing (e.g., *relationType*) rules.

Acquiring inheritance rules: In the cuticle scenario, elaboration reveals that the hypothetical leaf epidermis is impermeable to gases. This fact is established when KI determines that the epidermis is covered by a leaf cuticle, which is composed of cutin, a wax-like substance that is impermeable to gases (Figure 3.9). By analyzing the explanation of why the leaf epidermis is impermeable, KI determines that, under certain assumptions, all leaf epidermises are impermeable to gases. Consequently, KI asserts the inheritance specification that all leaf epidermises are assumed to be impermeable to gases. KI further associates this rule with its underlying assumptions (e.g., that the leaf cuticle is homogeneous, composed only of cutin; that it completely covers the epidermis). Making these underlying assumptions explicit permits identifying them as assumptions that might be violated in situations where the rule is found to be invalid (as discussed in the previous section.)

Acquiring taxonomic rules: The taxonomic subsumption hierarchy of the knowledge base is fundamental to the inference engine: establishing class membership of constants is pervasive as inheritance is propagated and predicate argument typing constraints are enforced. An important learning opportunity involves establishing new taxonomic relations.

(a) An explanation of the leaf epidermis' impermeability to gas

$$
\begin{array}{l}
impermeableToType(LeafEpidermis_1\ Gas) \\
\quad \Leftarrow_9\ covers(LeafCuticle_1\ LeafEpidermis_1) \\
\qquad \Leftarrow_8\ coveringPart(LeafEpidermis_1\ LeafCuticle_1) \\
\qquad\quad \Leftarrow_A\ isa(LeafEpidermis_1\ LeafEpidermis) \\
\qquad impermeableToType(LeafCuticle_1\ Gas) \\
\qquad \Leftarrow_6\ isa(LeafCuticle_1\ Cutin) \\
\qquad\quad \Leftarrow_5\ composedOf(LeafCuticle_1\ Cutin) \\
\qquad\qquad \Leftarrow_A\ isa(LeafEpidermis_1\ LeafEpidermis)
\end{array}
$$

(b) A proof establishing impermeability to gas

$$
\begin{array}{l}
impermeableToType(x\ Gas) \\
\quad \Leftarrow_9\ covers(skolem_i(x)\ x) \\
\qquad \Leftarrow_8\ coveringPart(x\ skolem_i(x)) \\
\qquad\quad \Leftarrow_A\ isa(x\ LeafEpidermis) \\
\qquad impermeableToType(skolem_i(x)\ Gas) \\
\qquad \Leftarrow_6\ isa(skolem_i(x)\ Cutin) \\
\qquad\quad \Leftarrow_5\ composedOf(skolem_i(x)\ Cutin) \\
\qquad\qquad \Leftarrow_A\ isa(x\ LeafEpidermis)
\end{array}
$$

(c) A shallow rule compiled from the proof

[∀ (x) isa(x LeafEpidermis) ⇒ impermeableToType(x Gas)]
≡
inherits(LeafEpidermis (impermeableToType) Gas)

(d) A deep rule compiled from the proof

[∀ (x) isa(x LeafEpidermis)
& unless(∃ (y) coveringPart(x y)
& [(impermeableToType(y Gas)
& [partiallyCovers(y x) **OR** (∃ (z) portal(x z) & ¬covers(y z))])
OR (∃ (z) composedOf(y Cutin) & composedOf(y z) & z≠Cutin)]
⇒ impermeableToType(x Gas)]

(a) An explanation for the hypothetical leaf epidermis being impermeable to gases. Rules 1 though 9 are presented in Figure 3.4; Rule A is from the interpretation of the new information, presented in Figure 3.2. **(b)** A general proof acquired by generalizing the ground explanation. **(c)** A new, shallow, inheritance rule acquired by compiling the general proof excluding the assumptions (e.g., the *unless* conditions in the antecedents of rules used in the explanation). **(d)** A rule acquired by compiling the general proof including the assumptions; this rule is correct to the extent that the rules used in the explanation are correct.

Figure 3.9: Compiling a new inheritance rule

(a) An explanation establishing the leaf cuticle is a container

$$
\begin{array}{l}
isa(LeafCuticle_1\ Container) \\
\quad \Leftarrow_2\ isa(LeafEpidermis_1\ Container) \\
\qquad \Leftarrow_1\ isa(LeafEpidermis_1\ LeafEpidermis) \\
\qquad\quad \Leftarrow_D\ isa(LeafCuticle_1\ LeafCuticle) \\
\qquad\qquad coveringPart(LeafEpidermis_1\ LeafCuticle_1) \\
\qquad\qquad\quad \Leftarrow_C\ isa(LeafCuticle_1\ LeafCuticle) \\
\qquad coveringPart(LeafEpidermis_1\ LeafCuticle_1) \\
\qquad \Leftarrow_C\ isa(LeafCuticle_1\ LeafCuticle)
\end{array}
$$

(b) A proof establishing every leaf cuticle is a container

$$
\begin{array}{l}
isa(x\ Container) \\
\quad \Leftarrow_2\ isa(skolem_i(x)\ Container) \\
\qquad \Leftarrow_1\ isa(skolem_i(x)\ LeafEpidermis) \\
\qquad\quad \Leftarrow_D\ isa(x\ LeafCuticle) \\
\qquad\qquad coveringPart(skolem_i(x)\ x) \\
\qquad\qquad\quad \Leftarrow_C\ isa(x\ LeafCuticle) \\
\qquad coveringPart(skolem_i(x)\ x) \\
\qquad \Leftarrow_C\ isa(x\ LeafCuticle)
\end{array}
$$

(c) A new taxonomic rule

$$
\begin{array}{c}
[\forall\ (x)\ \text{isa(x LeafCuticle)} \Rightarrow \text{isa(x Container)}] \\
\equiv \\
\text{ako(LeafCuticle Container)}
\end{array}
$$

(a) A ground explanation establishing that the hypothetical leaf cuticle is a container. **(b)** A general proof establishing that every leaf cuticle is also a container. **(c)** A new, shallow, taxonomic rule acquired by compiling the general proof.

Figure 3.10: Compiling a new taxonomic rule

In the cuticle example, elaboration reveals that the hypothetical leaf cuticle is a container. The explanation of this conclusion can be generalized and compiled into a new taxonomic rule stating that every leaf cuticle is also a container (Figure 3.10).

Acquiring new skolemizing rules: Skolemizing rules are also fundamental to the inference engine: they establish which implicit components can be included in a domain configuration (i.e., a representation of a set of domain objects arranged in order to denote some domain situation). Of particular importance are *relationType* rules, which define the types of other components

(a) An explanation establishing the cuticle covers the epidermis

$$covers(LeafCuticle_1\ LeafEpidermis_1)$$
$$\Leftarrow_8\ coveringPart(LeafEpidermis_1\ LeafCuticle_1)$$
$$\Leftarrow_C\ isa(LeafEpidermis_1\ LeafEpidermis)$$

(b) A proof establishing every cuticle covers an epidermis

$$covers(x\ skolem_i(x))$$
$$\Leftarrow_8\ coveringPart(skolem_i(x)\ x)$$
$$\Leftarrow_C\ isa(x\ LeafCuticle)$$

$$isa(skolem_i(x)\ LeafEpidermis)$$
$$\Leftarrow_C\ isa(x\ LeafCuticle)$$

(c) A new *relationType* rule

[∀ (x) isa(x LeafCuticle) ⇒ ∃ (y) isa(y LeafEpidermis) & covers(x y)]
≡
relationType(LeafCuticle covers LeafEpidermis)

(a) A ground explanation establishing that the hypothetical leaf cuticle covers the leaf epidermis. **(b)** A general proof acquired by generalizing the ground explanation. **(c)** A new, shallow, *relationType* rule formed by compiling the proof. However, the same rule is acquired by simply generalizing the hypothetical instances to be arbitrary elements of the classes of which they are hypotheticals. because this rule is subsequently invalidated by an applicable argument typing constraint, it is suggested to the user and not autonomously asserted.

Figure 3.11: Compiling a new skolemizing rule

that any instance of a collection can relate to, including its partonomic, ancestral, and behavioral relations. Therefore, an important learning opportunity involves acquiring new *relationType* rules.

In the cuticle example, elaboration reveals that the leaf cuticle covers the leaf epidermis. The explanation of this conclusion can be generalized and compiled into a new skolemizing rule stating that every leaf cuticle covers some leaf epidermis (Figure 3.11). However, rather than performing this weakest-precondition analysis on the explanation of this fact, KI creates a new rule by simply abstracting the fact. As before, this involves replacing the hypothtical terms with the collections they instantiate and then interpreting the resulting figurative proposition. This results in a new relationType rule (Figure 3.11c).

KI then determines whether this new rule is consistent with relevant existing rules (e.g., the applicable argument typing constraints).

This strategy (in contrast to the standard weakest-precondition analysis [MKKC86, DM86]) is adopted for relationType rules for two reasons:

1. KI expects to be reasoning with hypothetical instances of collections, and each instance typically denotes a very representative (and often arbitrary) element of the collection. Consequently, any binary relation that holds for two hypotheticals is also quite likely to hold for pairs of elements from the two classes. It is a reasonable conjecture that any particular element from one class will, by defualt, be related to some element from the other class. [10]

2. Easily accessed and applicable rules in the form of the predicate argument typing constraints are always available. These rules impose necessary conditions which the candidate rule must satisfy and help to identify unwarranted rules.

Thus, to acquire new inheritance and taxonomic rules, KI imposes a conservative test that requires candidate rules to be validated (e.g., through a weakest-precondition analysis of an inference path), but for new relationType rules, KI admits candidate rules which satisfy the applicable argument typing constraints.

[10] If KI were provided with new information comprising ground propositions on particular individuals, rather than rules, the expectation that reasoning would involve representative hypothetical instances of classes, rather than idiosyncratic instances, would be violated, and this strategy would be inappropriate.

As Figure 3.8 illustrates, the candidate rule is not consistent with an existing constraint and so is not asserted. However, despite this conflict, there exists a precedent (e.g., the fact $covers(LeafCuticle_1\ LeafEpidermis_1)$) for the rule, so KI simply creates a memo suggesting that the user consider adding the candidate rule to the knowledge base.

3.5.3 Abduction: acquiring rules to explain new beliefs

Explaining why a new belief is true establishes a better (e.g., more complete) comprehension of the new belief. Completing such explanations may require assuming additional beliefs, which extends the new information. Acquiring these additional beliefs promotes both the general learning goals of conviction (because the learner gains the capability of justifying the new belief) and completion.

When new taxonomic beliefs are established, KI attempts to identify refinements of existing taxonomic beliefs that subsume the new ones. For example, during interpretation, KI establishes the taxonomic belief $ako(LeafCuticle\ BotanicalOrganismComponent)$. Subsequently, when $ako(LeafCuticle\ Container)$ is established (Section 3.5.2), KI attempts to identify other collections that subsume the previous two and to propose them as possible generalizations of *LeafCuticle*. When two collections intersect, many existing, reified collections may be included in the intersection (i.e., collections that are proper specializations of both of the two intersecting collections). In observing the minimal change principle [Har86], KI identifies the *maximal specializations* of the current generalizations. A maximal specialization of a set of collections is an existing, reified collection that is a common subset – a subset of each element of the set of collections – but is not itself a proper subset of any other common subset.

In the example, the only maximal specialization of *Container* and *BotanicalOrganismComponent* is *BotanicalContainer* (Figure 3.8c), so KI proposes the new belief $ako(LeafCuticle\ BotanicalContainer)$. [11]

The resulting new taxonomic belief is automatically asserted by KI only if the affected collection (e.g., *LeafCuticle*) is a new concept and if there is only one maximal specialization; otherwise, the new belief is simply suggested to the user. Furthermore, if the new taxonomic belief is asserted, KI creates a memo suggesting that the user assert the rule that any element of the newly-subsumed generalizations (e.g., *Container* and *BotanicalOrganismComponent* is also an element of the maximal specialization. In the example, KI suggests asserting:

$$[\forall\ (x)\ isa(x\ Container)\ \&\ isa(x\ BotanicalOrganismComponent) \Rightarrow isa(x\ BotanicalContainer)]$$

3.6 Discussion

Through instantiating the new information, permitting non-skolemizing rules to exhaustively forward propagate, and analyzing the resulting learning opportunities, KI has begun to determine how the new information and existing knowledge interact. This analysis has proved useful, resulting in several worthwhile extensions to the knowledge base. However, the analysis thus far has been relatively shallow: it has been restricted to inference paths that reference only explicit concepts in the knowledge base and the hypothetical individuals that instantiate collections mentioned in the new information (e.g., $LeafEpidermis_1$

[11] This example illustrates the analysis performed when multiple *ako* beliefs are established for a class. A similar analysis is performed when multiple *isa* beliefs are established for a class.

and $LeafCuticle_1$). To further identify implicit consequences of the new information, this analysis must be taken deeper. Recognition extends the learning context to include instances of collections that are relevant to – but not explicitly mentioned in – the new information. This extension permits much deeper inference paths that establish interactions between new and prior knowledge.

Determining what prior knowledge to use while extending the learning context is a critical step in performing knowledge integration: it will circumscribe which inferences can be performed during the subsequent elaboration and, consequently, which learning opportunities will surface and be exploited during adaptation. However, the space of possible extensions to the learning context is vast, and identifying which prior knowledge is most useful to consider during knowledge integration is problematic. KI's method for performing recognition – for heuristically selecting prior knowledge to add to the learning context – is discussed in the next chapter.

Chapter 4

Recognition: Focusing Attention with Views

Hoare's Law of Large Problems:

Inside every large problem is a small problem struggling to get out.

Corollary:

Inside every large body of available knowledge is a small body of relevant knowledge struggling to get out.

4.1 The problem of focusing attention

In general, a recipient of new information does not know the contents of the new information in advance and therefore cannot know what existing knowledge will be relevant to its comprehension. Without the use of knowledge to make sense of new information, comprehension is limited to the most trivial level (e.g., one comparable to an emacs buffer; what might be called zeroth-order comprehension). Prior knowledge is required to investigate the consequences of the new information; e.g., by completing inferences. The difficult problem, however, is not completing inferences; rather it is deciding which inferences to complete among the potentially infinite number that could be completed. Therefore, inference during comprehension must be bounded, and an appropriate (finite) subset of inferences to complete must be selected. Any general computational model of comprehension must include some method of

focusing attention, of determining which inferences to complete.

By operationalizing new information and permitting non-skolemizing rules in the knowledge base to exhaustively forward-chain, elaboration begins to identify the consequences of the new information for prior knowledge. The partial entailment that is completed can be considered first-order comprehension: some implicit consequences of the new information have been made explicit, but only for a narrow range of concepts (e.g., those referenced by the new information and those already explicit in the knowledge base). Thus, this partial entailment is quite limited: it excludes consideration of skolemizing rules and the implicit concepts they reference. Deeper comprehension requires determining additional interactions between the new information and prior implicit knowledge; it requries extending the partial entailment; it requires further elaboration in which selected implicit concepts are made explicit and added to the learning context.

During recognition KI identifies existing knowledge that is relevant to the new information to further elaboration. This requires selecting fragments of prior knowledge, not already included in the learning context, to use while extending the partial entailment of new and prior knowledge. This is quite a difficult problem because there are so many alternative extensions: given n beliefs in the knowledge base, there are 2^n alternative extensions. This problem poses the single most difficult computational hurdle encountered while implementing KI. Its solution involves a basic approach that controls inference by restricting what concepts will be reasoned about (Section 4.2); the strategy for implementing this basic approach involves both a model of relevance (Section 4.3) and a context-based method for structuring knowledge (Section 4.4).

4.2 The approach: determining *what* to reason about

A pearl of conventional wisdom about reasoning with first-order theories states that completing inferences and instantiating quantified formulae are separable tasks [McA80]. Because reasoning with ground formulae is typically simple and fast, the problem of controlling first-order inference can be reduced to the problem of controlling instantiation. [1] Controlling reasoning by controlling the instantiation of quantified formulae is the foundation of KI's approach to guiding elaboration.

During elaboration, KI restricts reasoning to rules triggered by facts contained in the learning context; therefore, every fact established through inference is a consequence of other facts included in the learning context. At any point during knowledge integration, some subset of facts contained in the learning context are *primitive*, not derived as a consequence of other facts in the learning context. These primitive facts determine what inferred facts are established; they control elaboration. During recognition, KI determines what quantified formulae to instantiate as extensions of this set of primitive facts. [2]

The set of primitive facts in the learning context is initialized with facts that instantiate the new information (Figure 3.3). In other words, during the first cycle of comprehension, recognition simply "selects" the new infor-

[1] In fact, some researchers hold the extreme position that controlling instantiation is the "only difficult issue remaining in automated deduction" [McA80, page 1].

[2] The task of determining a set primitive of facts from which to reason extensively also arises in (e.g., qualitative) model-based reasoning. Typically, a particular model-based reasoning task will not require reasoning with the entire model, and, for tractability concerns, only a portion of the model (one that is sufficient for the task) will be used [FF91]. Determining this portion of the model (i.e., what components to include) is the model-selection problem. Recognition is quite similar to model-selection: it determines both which components (e.g., hypothetical instances) should be considered during elaboration and in which configuration these components should be arranged; the components, defined in a particular configuration, are the primitive facts used to initialize and extend the learning context.

mation. During each subsequent cycle of comprehension, recognition selects another set of quantified formulae to instantiate as an extension of the set of primitive facts. Each extension of the primitive facts enables a new region of implicit knowledge to be made explicit; explicating that region occurs during elaboration as the non-skolemizing rules exhaustively forward chain. Since each inferred fact is necessarily a consequence of the extension, the extension – the set of new primitive facts selected during recognition – controls the inferences that will be completed during elaboration.

The goal of comprehension is to assess how new information interacts with prior knowledge. Therefore, each extension of primitive facts should be *relevant* to the new information, that is, it should trigger inferences that establish consequences of both the new information and the extension.

4.3 A model of relevance

To focus attention during inference, KI adopts an inference-based model of relevance: two beliefs are *relevant* to the extent that they participate in common inference paths. Thus, a fact is relevant both to its consequences and to every fact of which it is a consequence. Under this interpretation, relevance is a reflexive, symmetric, and transitive relation. The symmetrical aspect of relevance is not common in other formulations of relevance (e.g., [Han92, SG87]).

Defining relevance in terms of inference would seem to preclude using relevance to control inference: if inference paths must be completed to establish relevance, how can relevance be used to determine which inference paths to complete? The answer to this paradox is that relevance itself is not directly used to control inference; rather, properties that suggest (i.e., correlate with)

relevance are used.

One such property is *connectedness:* two beliefs are mutually relevant with respect to a set of beliefs only if some subset of the belief set forms a connected graph that connects the them. Using property as a necessary condition for an inference-based model of relevance is warranted because rules are the syntactic conduits of inference and because the sets of facts referenced by rule instances [3] virtually always form connected graphs. When all instances of a rule are necessarily connected, the rule itself is said to be connected. [4] Furthermore, if every rule participating in an inference path is connected, then the set of facts referenced by the inference path froms a connected graph, and the inference path itself is said to be connected. If the knowledge base contains only rules that are connected, then every inference graph will be connected.

The connectedness requirement is exploited during recognition to drastically restrict the candidate extensions of the learning context: a set of new primitive facts is relevant to the existing contents of the learning context only when the union of the new primitive facts and some non-empty subset of the learning context forms a connected graph. Requiring the set of new primitive facts to be connected and to contain some belief that is already linked to (i.e., shares a term with) some belief in the learning context ensures that the new primitive facts is connected with some subset of the learning. This policy guarantees that if the learning context is connected it will remain so after

[3] A *rule instance* is a rule with all variables bound to constants such that the rule's antecedent is satisfied.

[4] Note that being connected is not a formal and universal requirement of rules. A particular representation language may permit independent, free floating clauses to appear either in a rule's antecedent (e.g., $[\forall\ (x)\ isa(x\ LivingObject)\ \&\ \exists\ (y)\ color(y\ Green)\ \Rightarrow\ \exists\ (z)\ parents(x\ z)]$) or in its consequence (e.g., $[\forall\ (xy)\ isa(x\ LivingObject)\ \&\ parents(x\ y)\ \Rightarrow\ \exists\ (z)\ color(z\ Green)]$). While such rules may be legal statements in the language, clauses that violate the connectedness requirement (e.g., $\exists\ (y)\ color(y\ Green)$) cause them to lack coherence (if not correctness).

recognition extends it with new primitive facts. Importantly, this restriction promotes selecting new primitive facts that can extend the inference paths already completed in the learning context, since inference paths can be extended only by facts that are connected to facts currently referenced by the inference graph. Thus, the restriction promotes selecting extensions that are relevant to the learning context. [5]

There are various ways to implement this restriction. One very simple method is to permit as new primitive facts only those that are directly linked to some fact already in the learning context. Including every fact that is directly linked to some fact in the learning context achieves a sort of "spreading activation" behavior [And83]. The resulting search for the consequences of new information is very complete. After cycle n (i.e., after n iterations of the recognition-elaboration cycle) every fact accessible from the new information by an access path of length n will be included in the learning context, and every consequence that requires an inference graph of order n will be established, where the *order* of an inference graph is the cardinality of the set of terms referenced by graph's leaf nodes. However, this strategy is too permissive: at cycle n, recognition will add to the learning context on the order of m^n new facts, where m is the average number of beliefs with which a fact shares some term. Furthermore, this strategy tends to minimally extend the length of access paths among concepts in the learning context. Therefore, each

[5] Inference path extensions, facilitated by new primitve facts, can deepen the inference paths in both directions. For example, by combining with existing facts in the learning context to trigger rules that establish facts new to the context they can extend inference paths with new roots (i.e., adding onto the "tops" of inference paths). Similarly, by combining with existing facts to trigger rules that establish beliefs already in the learning context (e.g., the beliefs that instantiate new information) they can identify new explanations of those established beliefs and extend inference paths with new leaves (i.e., adding onto the "bottoms" of inference paths).

cycle of recognition minimally increases the order of inference graphs that can be completed during elaboration. Assuming the depth of an inference graph correlates positively with its order, this strategy promotes the completion of relatively shallow inference graphs over relatively deep ones, where the *depth* of an inference graph is the greatest number of inference steps that connect the root belief to a leaf node. Thus, connectedness is a necessary condition for relevance, but alone it is insufficiently constraining, so additional criteria must be imposed.

To avoid exponential growth in the learning context and to promote the completion of deep inference paths, KI uses a context-based approach to select sets of new primitive facts with which to extend the learning context and thereby solve the problem of focusing attention. [6]

4.4 Views: contexts of mutually relevant beliefs

> *[A]ny problem that a person can solve at all is worked out at each moment in a small context ... [T]he key operations in problem-solving are connected with finding or constructing these working environments.* [Minsky, 1981]

Views are contexts; that is, they are sets of beliefs. Unlike other types of contexts, such as those that comprise beliefs sharing some epistemological basis (i.e., sets of implicit assumptions) [Guh91], views comprise beliefs that are mutually relevant, that is, they interact in some significant way and should be considered together. Views are connected.

[6] This model of relevance provides a computational account for why new information must reference known concepts (i.e., known constants) to be comprehended: if it does not, the new information is not connected with, and not relevant to, any belief in the knowledge base.

Figure 4.1 presents a view containing beliefs that describe the hypothetical leaf epidermis as a container. It includes facts denoting that the epidermis acts as a conduit in the leaf's acquisition of light and carbon dioxide and in the leaf's release of water vapor during transpiration.

The knowledge base includes a system of views. This system embodies a type of meta-knowledge: it structures the knowledge base and segregates beliefs into coherent (i.e., mutually relevant) delineated contexts. The view in Figure 4.1 is coherent because every fact contained therein is relevant to describing the leaf epidermis as a container.

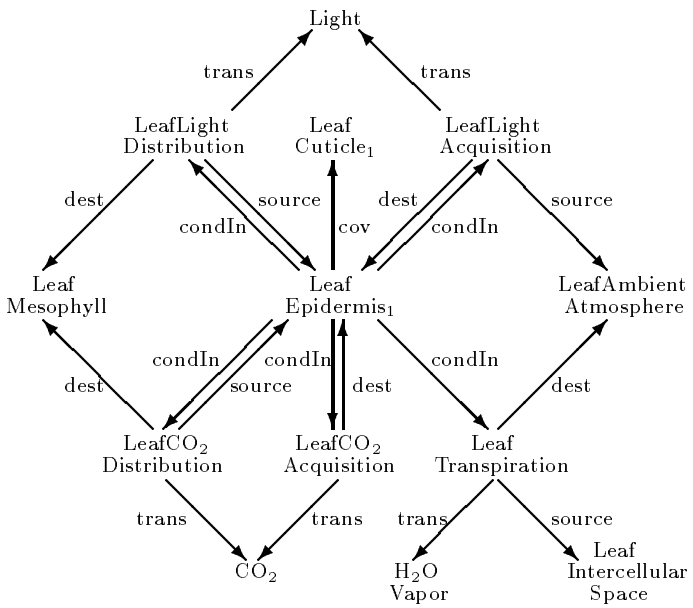
Intuitively, views are similar to gestalts. The components of a gestalt are so inextricably tied together that an agent cannot easily perceive or conceive of some of the components in isolation from the rest. Similarly, the contents of views are beliefs that are indexed by the view and so are not retrieved independently but rather as a collective whole. That is, when structured with views, knowledge is retrieved from the knowledge base as coherent constructs comprising sets of beliefs rather than as individual beliefs.

Views index the beliefs they contain. During recognition, KI determines an extension to the primitive facts contained in the learning context. Rather than selecting each new primitive fact independently, KI selects a single view. To ensure the selected view is relevant (i.e., connected to the learning context), the view must contain at least one belief that is directly linked to some belief in the learning context. Furthermore, to ensure the selected view extends the primitive facts of the learning context, it must include at least one belief not included in the learning context. The large problem of focusing attention during recognition then decomposes into the smaller problems of creating and selecting views.

(a) The view: *LeafEpidermis$_1$ QuaContainer*

{coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)
conduitIn(LeafEpidermis$_1$ LeafLightAcquisition)
conduitIn(LeafEpidermis$_1$ LeafLightDistribution)
conduitIn(LeafEpidermis$_1$ LeafCO$_2$Acquisition)
conduitIn(LeafEpidermis$_1$ LeafTranspiration)
conduitIn(LeafEpidermis$_1$ LeafCO$_2$Distribution)
source(LeafCO$_2$Distribution LeafEpidermis)
destination(LeafCO$_2$Distribution LeafMesophyll)
transportee(LeafCO$_2$Distribution CO$_2$)
source(LeafTranspiration LeafIntercellularSpace)
source(LeafLightAcquisition LeafAmbientAtmosphere)
destination(LeafLightAcquisition LeafEpidermis)
transportee(LeafLightAcquisition Light)
source(LeafLightDistribution LeafEpidermis)
destination(LeafLightDistribution LeafMesophyll)
transportee(LeafLightDistribution Light)
source(LeafCO$_2$Acquisition LeafAmbientAtmosphere)
destination(LeafCO$_2$Acquisition LeafEpidermis)
transportee(LeafCO$_2$Acquisition CO$_2$)
transportee(LeafTranspiration H$_2$OVapor)
destination(LeafTranspiration LeafAmbientAtmosphere)}

(b) The view presented as a semantic network



Abbreviations:

trans : transportee
dest : destination
condIn : conduitIn
cov : coveringPart

(a) The view *LeafEpidermis$_1$ QuaContainer* represents the hypothetical leaf epidermis in its role as a container. The view comprises twenty-one beliefs, including one fact (e.g., *coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)*) and twenty figurative beliefs. **(b)** The view represented as a semantic network. For clarity, the belief *source(LeafCO$_2$Acquisition LeafAmbientAtmosphere)* is omitted from the diagram.

Figure 4.1: An example view

A large knowledge base contains a plethora of possible contexts. Considering only contexts comprising skolemizing rules, 2^n contexts can be defined for a knowledge base containing n skolemizing rules. Since it is untenable to consider all possible contexts, the algorithms developed for creating and manipulating contexts must be biased to consider only a select subset. KI exploits two sources of bias: an *ontological bias* restricts the set of primitive contexts that can be defined (Section 4.4.1); an *indexical bias* restricts the set of contexts that can be accessed in any given situation (Section 4.4.2).

4.4.1 Creating views with view types

The most imposing question concerning the use of views is: How are these convenient contexts of mutually-relevant beliefs created? One approach assumes each view is constructed prior to the reasoning tasks that require their use (e.g., all views are created during some monolithic sweep through the knowledge base). However, this is untenable because the system's knowledge includes implicit concepts that cannot be explicitly referenced and so cannot be explicitly included in any handcrafted view. For example, the view presented in Figure 4.1 cannot be defined before that hypothetical instance was created. Also, there are an infinite number of plants and leaves and leaf epidermises, etc., tacitly represented; making them (and the views that reference them) explicit is not feasible. Furthermore, assuming the knowledge base is incomplete, there are domain concepts (e.g., the leaf cuticle) that cannot be referenced within views before their introduction. Therefore, the knowledge base cannot undergo some single, definitive structuring process; view construction must be dynamic.

The key observation underlying the approach adopted in KI is that many views share a common structure; that is, while the terms referenced

within two similar views might be completely different, the relations between those terms are the same in both views. For example, Figure 4.1 illustrates the view representing the hypothetical leaf epidermis in its role as a container. Not surprisingly, there are analogous views for other leaf epidermises represented as containers. The terms of these views are different (e.g., each leaf epidermis has a distinct leaf cuticle); however, the relations that hold over the terms referenced in each view will be the same. In other words, when represented as semantic networks, these views tend to be isomorphic; the nodes differ, but the arcs are identical. Similarly, analogous views represent other types of epidermis (e.g., the stem epidermis, the root epidermis) considered as containers. The semantic network representation of these views will likely not be isomorphic because direct analogs do not exist in every view for each term in any view. For example, there will likely not be an analog of the leaf acquisition of light in a view representing the root epidermis as a container. However, the set of relations, their identity and pattern, will remain similar; they will include propositions representing the contents of the container as well as the events that move things into and out of the container. Furthermore, there are analogous views representing the seed coat, the cell wall, and the pollen sac, each having different terms but a common pattern of relations. In general, anything that functions as a container can be represented as a container, and a common pattern of relations will be appropriate for every representation. Therefore, a class of views can be defined and the relations common to every instance of this class can be identified and associated with the class. Knowledge-base constants denoting classes of views are called *view types*.

Figure 4.2 presents an example of a view type. View type *QuaContainer* identifies the knowledge-base paths emanating from a concept that iden-

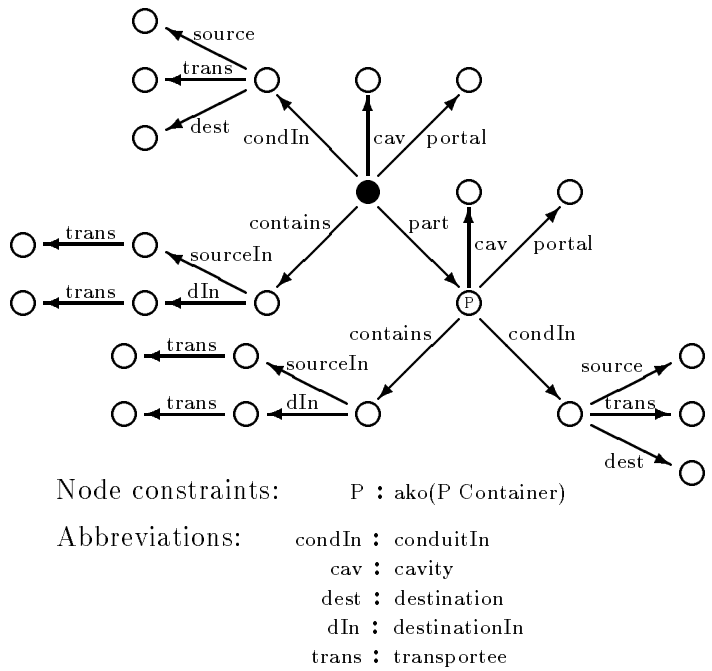
tify properties relevant to representing that concept as a container. These properties include the contents of the container, the processes that transport items into and out of the container, and the parts that are themselves containers.

A view type is a semantic-network schema, represented as a set of paths emanating from a root node. Rather than having knowledge-base constants as nodes, it has variables, each of which can bind to one or more constants. A view is defined by instantiating a view type. This involves binding the variables to (zero or more) constants and, perhaps, specializing the relations. For example, the view type in Figure 4.2 is instantiated to produce the view in Figure 4.1. The view includes five constants (e.g., $LeafCO_2Distribution$, $LeafCO_2Acquisition$, $LeafTranspiration$, ...) that bind to a single variable in the view type, and the relation *part* in the view type is replaced by its specialization $coveringPart$ in the view.

Views are created by applying a view type to a domain concept: the concept is bound to the root node and each access path is instantiated. Only the individual view types are manually constructed. Each view type can then be used to generate (a potentially infinite number of) distinct views as they are needed. Currently, there are twenty-two handcrafted view types defined in the knowledge base.

Instantiating an access path within a view for a given concept first involves binding the root node of the access path to the concept, then determining the bindings for the next node along the path. The arc that connects the root node to the next node is a binary predicate. Candidate bindings for the node include the second arguments of facts in the knowledge base whose predicates are a specialization of the arc and whose first arguments are the concept bound to

The view-type *QuaContainer*



The view type *QuaContainer* represented as a semantic-network schema. It contains access paths relevant to considering a thing as a container; the shaded node is the position of the root concept. Nodes are variables that can bind to knowledge-base constants: unlabeled nodes are unconstrained; labeled nodes are constrained (e.g., any constant binding to node *P* must be a subset or element of the collection *Container*).

Figure 4.2: An example view type

root node. For example, while instantiating *QuaContainer* for $LeafEpidermis_1$, $LeafCuticle_1$ is a candidate binding for the node connected to the root node by the arc labeled *part* since $coveringPart(LeafEpidermis_1\ LeafCuticle_1)$ has been established and $akoSlot(coveringPart\ part)$ is true. Candidate bindings for the node also include the third arguments of *relationType* assertions whose second arguments are specializations of the arc and whose first arguments are collections that contain (i.e., as an element or as a subset) the concept bound

to the root node. [7] For example, *LeafLightAcquisition* is a candidate binding for the node connected to the root node by the arc labeled *conduitIn* since *relationType*(*LeafEpidermis conduitIn LeafLightAcquisition*). Finally, if the node is labeled and has constraints, then candidate bindings that do not satisfy the node's constraints are discarded. Since prior elaboration established *isa*($LeafCuticle_1$ *Container*) (Figure 3.5), $LeafCuticle_1$ satisfies the constraint on node *P* (Figure 4.2) and is accepted as a binding. The remaining candidate bindings are returned as bindings. [8] To instantiate the rest of the path, each node binding is treated as a distinct, new root node, and the remaining access path is instantiated for it using precisely the same procedure recursively. The number of recursions equals the depth of the access path.

In the process of instantiating an access path, if there are no candidate bindings for a node and if the predicate denoted by the arc connecting the node to its predecessor is expected to have a value, then a learning opportunity is detected. While creating the view presented in Figure 4.1, for example, no candidate bindings are found for the nodes connected to the root node by the arcs labeled *cavity*, *portal* and *contains*. The predicates *portal* and *contains* are expected to have values for leaf epidermises because, respectively, most plant components have some sort of portal, and epidermises contain the internal parts of the morphological part they cover (i.e., the knowledge base includes the assertions *likelyForType*(*portal BontanicalComponent*) and *likelyForType*(*contains Epidermis*)). However, the knowledge base includes no expectation that the leaf epidermise

[7] These second type of candidate bindings result in figurative references. Only the most specific set of those figurative references for which no instance-level candidate bindings exist is included as candidate bindings.

[8] As illustrated in Figure 4.1, a view can include multiple bindings for any non-root node in a view-type. Each distinct binding of the root node spawns a distinct view.

has a value for the predicate *cavity*. [9] Consequently, when creating the view $LeafEpidermis_1 QuaContainer$, KI detects two gaps in the knowledge base and creates a memo to the user suggesting the missing information be provided. The memo includes the request: *"For LeafEpidermis, please specify the relation types for predicates: contains, portal."* KI thus exploits expectations that particular predicates should have values for elements of particular collections (e.g., *likelyFor* assertions) to detect gaps in the knowledge base and request additional knowledge to fill those gaps.

An important property of view types (and their constituent access paths) is that they are trees and thus are connected, rooted, acyclic graphs. The views created by instantiating view types are also graphs. Their nodes are those constants in the knowledge base that can bind to the variables referenced in the access paths, and their arcs are the predicates (or their specializations) that appear in the access paths. Views need not be acyclic since a concept that binds to one access-path node may also bind to another subsequent node along the same access path. However, views are necessarily connected. Therefore, the primary effect of KI's ontological bias for defining views is that only connected contexts can be created with view types. This bias promotes defining relevant contexts since views are necessarily connected. Furthermore, the bias dramatically restricts the space of contexts that can be created as views: rather than considering the space of all the knowledge-base subgraphs, only the space of connected subgraphs is considered.

The view presented in Figure 4.1 contains one belief included in the

[9] In fact, the knowledge base should include this expectation as well but is missing the assertion $likelyFor(cavity\ LeafEpidermis)$; it is, of course, also missing the assertion $relationType(LeafEpidermis\ cavity\ LeafIntercellularSpace)$.

Interpreting figurative references in *$LeafEpidermis_1 QuaContainer$*

[∃ (a b c d e f g h i j k)
isa(a LeafMesophyll) & isa(b CO_2) & isa(c H_2OVapor) & isa(d Light) & isa(e LeafAmbientAtmosphere)
& isa(f LeafIntercellularSpace) & isa(g LeafLightAcquisition) & isa(h LeafLightDistribution)
& isa(i LeafCO_2Acquisition) & isa(j LeafCO_2Distribution) & isa(k LeafTranspiration)
& conduitIn(LeafEpidermis$_1$ g) & transportee(g d) & source(g e) & destination(g LeafEpidermis$_1$)
& conduitIn(LeafEpidermis$_1$ h) & transportee(h d) & source(h LeafEpidermis$_1$) & destination(h a)
& conduitIn(LeafEpidermis$_1$ i) & transportee(i b) & source(i e) & destination(i LeafEpidermis$_1$)
& conduitIn(LeafEpidermis$_1$ j) & transportee(j b) & source(j LeafEpidermis$_1$) & destination(j a)
& conduitIn(LeafEpidermis$_1$ k) & transportee(k c) & source(k f) & destination(k e)]

The interpretation of the figurative references in view *$LeafEpidermis_1 QuaContainer$* is an existentially quantified formula that can be used to extend the learning context.

Figure 4.3: Interpreting a view

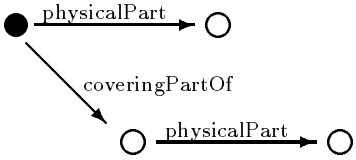
learning context at the end of the first cycle of elaboration:

$$coveringPart(LeafEpidermis_1\ LeafCuticle_1)$$

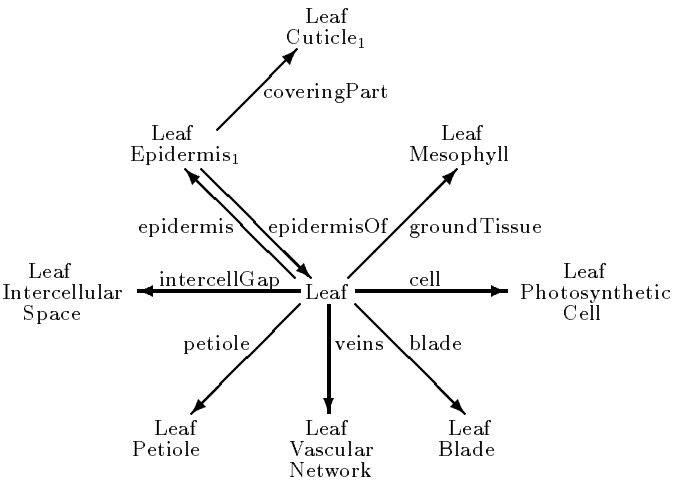
Therefore, this view is connected to the learning context. However, most of the beliefs included in this view are figurative; that is, they make figurative references. As with new information, figurative beliefs contained in views are interpreted as quantified formulae. Figure 4.3 presents the interpretation of the figurative beliefs included in this view. The existentially quantified variables denoted by figurative beliefs included in a view define new concepts (e.g., hypothetical individuals) that can be considered during elaboration. Instantiating these existentially quantified formulae defines new primitive facts that can be added to the learning context. For example, extending the learning context with this view adds thirty-one new primitive facts and eleven new terms to the learning context.

It is often useful to represent a given concept in different roles: a leaf can be considered in its role as a producer (e.g., of sugar via photosynthesis) or as a consumer (e.g., of carbon dioxide and water). Consequently, there

(a) The view type: *QuaCoveringPart*



(b) The view *LeafEpidermis*$_1$ *QuaCoveringPart*



(a) The view type *QuaCoveringPart* contains access paths relevant to considering something as a covering part of an object. **(b)** The view *LeafEpidermis*$_1$ *QuaCoveringPart* represents the hypothetical leaf epidermis in its role as the covering part of some leaf.

Figure 4.4: A second view of LeafEpidermis$_1$

can be several, perhaps many, view types applicable to each concept in the knowledge base. Each view type defines a distinct view of that concept. For example, another view type applicable to *LeafEpidermis*$_1$ is *QuaCoveringPart*. Figure 4.4 presents this view type and the view that results when it is applied to *LeafEpidermis*$_1$.

Not all view types can be applied to all concepts in the knowledge base; it doesn't make (literal) sense, for example, to apply the view type *QuaContainer* to an event (e.g., *Photosynthesis*). Therefore, each view type has

preconditions that act as sufficiency constraints: the view type can be applied only to concepts that satisfy its preconditions. Since each view identifies beliefs relevant to considering some concept in a particular situation, the preconditions of a view type simply require that a proposed root concept be in the situation designated by the view type. For example, *QuaContainer* can be applied to any concept that is a subordinate (i.e., an element or a subset) of *Container*; *QuaCoveringPart* can be applied to any plant component known to function as a covering (e.g., any subordinate of *Epidermis*, *OvuleIntegument*, or *SeedCoat*); and *QuaDehydratingLivingThing* can be applied to any living thing that is dehydrating. By associating preconditions with view types, the appropriateness of a candidate view can be determined before it is created, and inappropriate views need never occur.

To extend the learning context, KI identifies relevant views by determining the view types applicable to concepts already contained in the learning context. At the end of the first cycle of elaboration, the learning context contains the two hypothetical individuals $LeafEpidermis_1$ and $LeafCuticle_1$. KI identifies the view types applicable to these concepts.

There are four view types that apply to $LeafEpidermis_1$:

1. *QuaContainer* – applying it to $LeafEpidermis_1$ identifies beliefs representing the leaf epidermis considered as a container (Figure 4.1).

2. *QuaCoveringPart* – applying it to $LeafEpidermis_1$ identifies beliefs representing the leaf epidermis considered as the covering part of the leaf (Figure 4.4).

3. *QuaPhysicalComponent* – applying it to the $LeafEpidermis_1$ identifies beliefs representing the leaf epidermis considered as a physical component of the

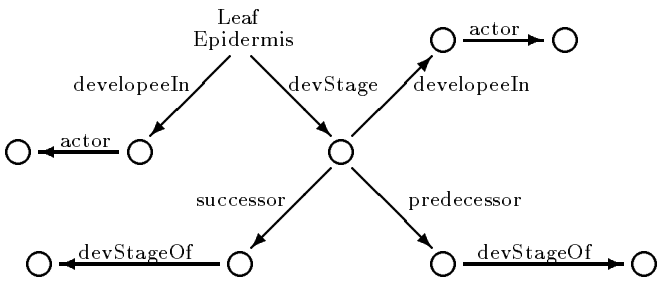
leaf. This set of beliefs is identical to $LeafEpidermis_1 QuaCoveringPart$.

4. *QuaBiologicalDevelopingThing* – applying it to $LeafEpidermis_1$ attempts to identify beliefs representing the leaf epidermis considered as a living thing with a developmental progression from creation through maturity to death.

This last view, however, cannot be created because the beliefs denoting the developmental progression of a leaf epidermis are missing from the knowledge base. Any concept that satisfies the precondition of $QuaBiologicalDevelopingThing$ (e.g., any subordinate of *BiologicalLivingThing*) is expected to root a set of beliefs that denote its developmental progression. When these beliefs are found to be missing for $LeafEpidermis$, KI detects a gap in the knowledge base and creates a memo suggesting that the user add beliefs to the knowledge base that describe how the leaf epidermis develops. By referring to the access paths associated with the view type, this request for additional information can be quite focused (Figure 4.5).

There are three view types that apply to $LeafCuticle_1$. Prior elaboration established that the leaf cuticle is an instance of both container and cutin (Figure 3.5). Furthermore, cutin is a type of both *BotanicalSubstance* – the collection of non-living compounds, solutions or mixtures that are part of a plant – and *OrganismComponent* – the collection of physical parts of organisms. By virtue of being a container, the view type *QuaContainer* is applicable to the leaf cuticle; by virtue of being a botanical substance, the view type *QuaBiologicalProduct* is applicable to it; and by virtue of being an organism component, the view type *QuaPhysicalComponent* is applicable as well. However, since $LeafCuticle$ is a new concept, there is insufficient knowledge about it to create views capable

Suggestion: *Please describe how the LeafEpidermis develops.*



Abbreviations: devStage : developmentalStage
devStageOf : developmentalStageOf

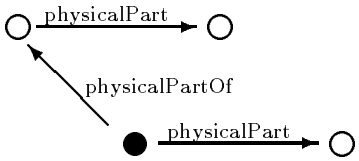
The view type *QuaDevelopingThing* contains access paths relevant to considering the developmental progression of a living thing. When applying this to *LeafEpidermis*, no bindings could be found for any of the non-root node variables. Since this view type is applicable to every living thing (i.e., every living thing should exhibit a developmental progression), the absence of these beliefs reveals knowledge base gaps. A memo is created suggesting the user describe the developmental progression of the leaf epidermis by completing the access paths in *QuaDevelopingThing* rooted at *LeafEpidermis* (i.e., by specifying what concepts bind to the variable nodes in these access paths).

Figure 4.5: Soliciting knowledge of how the leaf epidermis develops.

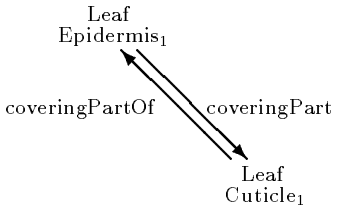
of extending the learning context. The view $LeafCuticle_1QuaPhysicalComponent$ simply contains the fact that the leaf cuticle covers the leaf epidermis (Figure 4.6). Furthermore, this view is already activated (i.e., every belief is fully instantiated and already contained in the learning context). There are no beliefs contained in the other two views because there is no explicit knowledge of how the leaf cuticle functions as a container or how it is produced. Consequently, KI creates memos suggesting that the user provide the missing knowledge to complete these views (e.g., Figure 4.7).

Typically, however, there will be many concepts represented in the learning context, each having several different applicable views types that define candidate views for extending the learning context. Therefore, a method is

(a) The view type: *QuaPhysicalComponent*



(b) The view *LeafCuticle*$_1$ *QuaPhysicalComponent*



(a) The view type *QuaPhysicalComponent* contains access paths relevant to considering something as a physical component of an object. **(b)** The view *LeafCuticle*$_1$ *QuaPhysicalComponent* represents the hypothetical leaf cuticle as the covering part of the leaf epidermis. The leaf cuticle is the only known part of the leaf epidermis (due to knowledge-base gaps), so there are no node bindings to concepts not already included in the learning context. Consequently, this view is a proper subset of the learning context and does not extend it.

Figure 4.6: A view of LeafCuticle$_1$

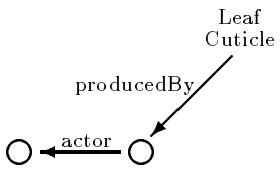
needed for selecting one view from among the numerous alternatives.

4.4.2 Selecting views

KI's indexical bias restricts the set of alternative views that can by accessed during learning. Selecting a view to compose with the existing learning context is performed in a generate and test manner: alternative candidate views are first identified, then a single candidate view is selected.

Identifying candidate views: Every concept in the learning context is an eligible root concept; every applicable unactivated view of each eligible root

Suggestion: *Please describe how the LeafCuticle is produced.*



The view type *QuaBotanicalProduct* contains access paths relevant to considering how a botanical substance is produced. When applying this to *LeafCuticle*, no bindings could be found for any of the non-root node variables. Since this view type is applicable to every botanical substance (i.e., every substance in a plant is produced by some event), the absence of these beliefs reveals knowledge base gaps. A memo is created suggesting that the user describe how the leaf cuticle is produced by completing the access paths in *QuaBiologicalProduct* rooted at *LeafCuticle*.

Figure 4.7: Soliciting knowledge of how the leaf cuticle is produced.

concept is an *eligible view*. However, creating views indiscriminately is computationally prohibitive for an interactive system. Therefore, a set of views, called *candidate views*, is heuristically selected from those eligible, and only candidate views are created. Selecting candidate views involves three steps:

1. The first step selects views whose view types are most specific. Because some views types are more specialized and focused than others, view types – and the views created by applying them to concepts – can be related hierarchically. For example, *QuaCoveringPart* is a specialization of *QuaPhysicalPart*; it refines (i.e., further restricts) the set of beliefs that will be included in the view and is expected to provide greater focus; it promotes greater mutual relevance among the included beliefs. Each view type that is a generalization of another view type applicable to the same concept is removed from consideration. In the example, the eligible view $LeafEpidermis_1 QuaPhysicalComponent$ is removed.

Qualitative Value	Numeric Value
VeryHigh	1.0
High	0.75
Medium	0.5
Low	0.25
VeryLow	0.01

Figure 4.8: The qualitative and numeric values of interestingness

2. The second step ranks the remaining eligible views by a heuristic estimate of interestingness, a product of the interestingness of the root concept and the default interestingness of the view type.

   The default interestingness of each view type is a qualitative attribute provided manually as view types are defined. The qualitative values for specifying interestingness are converted into real numbers between 0 and 1 (Figure 4.8). In the example, five different view types structure the six eligible views remaining under consideration. Figure 4.9 presents the default interestingness associated with each view type defined in the knowledge base.

   Computing the interestingness of a concept (e.g., the root concept of an eligible view) involves summing the interestingness of every fact asserted in the learning context whose first argument is the concept. A small set of heuristics is used to appraise qualitatively how interesting a proposition is within the learning context. The qualitative value is then converted into a real number (Figure 4.8). The heuristics for appraising how the interestingness of a proposition are explained in Appendix D and summarized in Figure 4.10. Note that some are context specific (e.g., heuristics *a, b, e, f, g,* and *h*), while others are not.

View Type	Qualitative Value	Numeric Value
QuaAcquirer	VeryHigh	1.0
QuaConsumer	VeryHigh	1.0
QuaDehydratingThing	VeryHigh	1.0
QuaProducer	VeryHigh	1.0
QuaStarvingThing	VeryHigh	1.0
QuaAssimilateProcessor	High	.75
QuaBotanicalGreenThing	High	.75
QuaPlantEnergySource	High	.75
QuaResourceAssimilator	High	.75
QuaResourceAttainment	High	.75
QuaResourceUtilization	High	.75
QuaBotanicalOrgan	Medium	.5
QuaDevelopingSystem	Medium	.5
QuaDevelopingThing	Medium	.5
QuaAttachedPart	Low	.25
QuaBiologicalProduct	Low	.25
QuaContainedObject	Low	.25
QuaContainer	Low	.25
QuaCoveringPart	Low	.25
QuaPhysicalComponent	Low	.25
QuaPortal	Low	.25
QuaResourceAssimilate	Low	.25

Figure 4.9: The default interestingness of the view types

The six eligible views under consideration have two different root concepts (e.g., $LeafEpidermis_1$ and $LeafCuticle_1$); the appraisal of interestingness for these two concepts is presented in Figure 4.11.

3. The third and final step orders the set of eligible views by their interestingness estimates and selects candidate views in descending order of their interestingness estimate until the cardinality of the set of candidate views reaches a given threshold. [10] Figure 4.12 presents the eligible views ordered by their interestingness estimates.

[10] This threshold is a parameter to KI; its value was 10 for all the described examples.

Given a proposition p, of the form $s(x\ y)$
Compute $interest(p)$ as follows:

[a] if x or y is directly referenced by the training, then return VeryHigh
[b] else if p explains some fact that instantiates the training, then return VeryHigh
[c] else if p denotes a domain goal (e.g., a physiological goal) of x, then return VeryHigh
[d] else if p is anomalous (e.g., it conflicts with some constraint) then return VeryHigh
[e] else if p is a consequence of the training, then return High
[f] else if x or y instantiate a class referenced by the training, then return High
[g] else if p refutes an existing assumption then return High
[h] else if p is not asserted in the current context, then return Medium
[i] else if s is a modulatory predicate (e.g., *enables*, *restricts*), then return Medium
[j] else if y is an attribute value, then return Medium
[k] else if s is a partonomic predicate, then return Low
[l] else if p denotes the participation of some entity in an event then return Low
[m] else return VeryLow

The letters along the left margin label the individual interestingness heuristics and will be used to identify how the interestingness of particular propositions was assessed.

Figure 4.10: Estimating the interestingness of a proposition

Significantly, the resulting set of candidate views is determined without actually creating any views. The typically large set of eligible views is pared down to the relatively small set of candidate views using only the interestingness of the eligible root concepts, the a priori interestingness of the view types, and the taxonomic relations among view types. The (sometimes computationally expensive) task of creating views need only be performed for a small set containing the most promising candidates.

Each candidate view is then generated: the view type is applied to the root concept to determine the set of beliefs from the knowledge base that represents the root concept in the situation denoted by the view type. As previously discussed, one of the views of $LeafEpidermis_1$ and two of the three views of $LeafCuticle_1$ are empty due to insufficient domain knowledge in the knowledge base; the third view of $LeafCuticle_1$ is already activated (i.e., ev-

(a) computing the interestingness of LeafCuticle$_1$

Proposition	Rule	Qualitative	Numeric
composedOf(LeafCuticle$_1$ Cutin)	[a]	VeryHigh	1.0
isa(LeafCuticle$_1$ Cutin)	[a]	VeryHigh	1.0
covers(LeafCuticle$_1$ LeafEpidermis$_1$)	[e]	High	0.75
impermeableToType(LeafCuticle$_1$ Gas)	[e]	High	0.75
impermeableToType(LeafCuticle$_1$ Liquid)	[e]	High	0.75
isa(LeafCuticle$_1$ Container)	[e]	High	0.75
isa(LeafCuticle$_1$ Solid)	[e]	High	0.75
transparency(LeafCuticle$_1$ Opaque)	[e]	High	0.75
coveringPartOf(LeafCuticle$_1$ LeafEpidermis$_1$)	[f]	High	0.75
total for LeafCuticle$_1$			7.25

(b) computing the interestingness of LeafEpidermis$_1$

Proposition	Rule	Qualitative	Numeric
hasCover(LeafEpidermis$_1$ LeafCuticle$_1$)	[d]	VeryHigh	1.0
impermeableToType(LeafEpidermis$_1$ Gas)	[e]	High	0.75
impermeableToType(LeafEpidermis$_1$ Liquid)	[e]	High	0.75
basicUnit(LeafEpidermis$_1$ BotanicalCell)	[f]	High	0.75
coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)	[f]	High	0.75
isa(LeafEpidermis$_1$ Solid)	[f]	High	0.75
transparency(LeafEpidermis$_1$ Opaque)	[f]	High	0.75
total for LeafEpidermis$_1$			5.5

Figure 4.11: Estimating the interestingness of the root concepts

Root Concept	View Type	Estimated Interestingness
LeafEpidermis$_1$	QuaDevelopingThing	2.7500
LeafCuticle$_1$	QuaBiologicalProduct	1.8125
LeafCuticle$_1$	QuaContainer	1.8125
LeafCuticle$_1$	QuaPhysicalComponent	1.8125
LeafEpidermis$_1$	QuaCoveringPart	1.3750
LeafEpidermis$_1$	QuaContainer	1.3750

Figure 4.12: Eligible views ranked by interestingness estimates

ery belief is fully instantiated and already contained in the learning context). Consequently, these four views are removed from further consideration.

Having been created, the contents of the two remaining candidate views are appraised for their *activation level* (see Section 2.1.2). Each candidate view is ranked using measures of coreference, which is a heuristic estimate of relevance, and interestingness.

Appraising view relevance: The relevance between two contexts cannot be directly measured without exploring the inferences enabled by the beliefs contained in those contexts. Therefore, appraising the actual relevance between candidate views and the learning context is not possible since each candidate view is not yet activated. However, the relevance between two contexts can be estimated by the extent to which their beliefs reference the same concepts. Note that when the concepts referenced by two contexts are disjoint, their union is not connected. Consequently, no inferences will be triggered by merging the two contexts, and no inference graphs will contain beliefs from both contexts. Thus, the two contexts are irrelevant to each other. On the other hand, each concept referenced by both contexts can bridge an access path connecting beliefs in one context to beliefs in the other, and each such path is also a potential access path among beliefs appearing in an inference graph. When high overlap occurs among the concepts referenced by two contexts, there is greater potential for inferential synergy between the two (i.e., for inference graphs to contain beliefs from each context); hence the two contexts are more likely to be relevant.

This heuristic estimate of relevance is evaluated for one context (i.e., set of propositions) with respect to another. It measures the *coreference* of the two contexts; that is, the degree to which the two contexts reference common

$$\text{coreference}(C_1\ C_2) = \text{containment}(C_1\ C_2) * \text{coverage}(C_1\ C_2)$$
$$\text{containment}(C_1\ C_2) = |\ \text{concepts}(C_1) \cap \text{concepts}(C_2)\ | \div |\ \text{concepts}(C_1)\ |$$
$$\text{coverage}(C_1\ C_2) = |\ \text{concepts}(C_1) \cap \text{concepts}(C_2)\ | \div |\ \text{concepts}(C_2)\ |$$

C_1 and C_2 denote two arbitrary contexts (e.g., C_1 a candidate view and C_2 the learning context). The function $concepts(C_i)$ returns the set of relational terms contained in C_i.

Figure 4.13: Estimating the mutual relevance of two contexts

concepts.

Coreference is computed for each candidate view with respect to the learning context. Specifically, coreference is measured as the product of two ratios (Figure 4.13). The first measures a view's *containment*: the portion of those concepts contained in the view that are also contained in the learning context. The second measures a view's *coverage*: the portion of those concepts contained in the learning context that are also contained in the view.

Figure 4.14 presents the relevance estimates for each of the two candidate views from the example. Both views completely cover the learning context. [11] Since $LeafEpidermis_1QuaCoveringPart$ is smaller (e.g., contains fewer terms), it has a higher containment value and so is estimated to be more relevant to the learning context than $LeafEpidermis_1QuaContainer$. [12]

In early versions of KI, only the coverage ratio was used to estimate a

[11] Note: for estimating relevance, only the hypotheticals of the learning context are considered.

[12] When a view's containment (with respect to the learning context) is 1, then every concept referenced by the view is already in the learning context, and the view cannot introduce new relational terms (e.g., hypothetical instances) to the learning context. However, not every proposition contained in the view is necessarily already asserted in the learning context. In this situation, activating the view (e.g., establishing the as yet unasserted propositions) tends to be inexpensive and so is performed, and the view is labeled as activated and removed from further consideration.

(a) Estimating the relevance of LeafEpidermis$_1$QuaContainer

| concepts(LeafEpidermis$_1$QuaContainer) | = 13
| concepts(LearningContext) | = 2
| concepts(LeafEpidermis$_1$QuaContainer) ∩ concepts(LearningContext) | = 2
coverage = 2 ÷ 2 = 1.0
containment = 2 ÷ 13 = .1538
relevance(LeafEpidermis$_1$QuaContainer LearningContext) = 1.0 * .1538 = .1538

(b) Estimating the relevance of LeafEpidermis$_1$QuaCoveringPart

| concepts(LeafEpidermis$_1$QuaCoveringPart) | = 9
| concepts(LearningContext) | = 2
| concepts(LeafEpidermis$_1$QuaCoveringPart) ∩ concepts(LearningContext) | = 2
coverage = 2 ÷ 2 = 1.0
containment = 2 ÷ 9 = .2222
relevance(LeafEpidermis$_1$QuaCoveringPart LearningContext) = 1.0 * .2222 = .2222

(a) The view *LeafEpidermis$_1$QuaContainer* contains thirteen relational concepts (Figure 4.1); the learning context contains two (Figure 3.3), both of which are also contained in the view. **(b)** The view *LeafEpidermis$_1$QuaCoveringPart* contains nine relational concepts (Figure 4.4) including both of those in the learning context.

Figure 4.14: Estimating the relevance of the candidate views

view's relevance [Mur88, MP89]. However, this has proved inadequate because it promotes strictly larger, more encompassing views. For example, a huge view containing 100 concepts, including all, say, 15 concepts in the learning context, would be judged more relevant than a view containing 15 concepts including only 14 of those in the learning context. Always selecting very large views tend to introduce an over abundance of new relational terms (e.g., hypotheticals). This sacrifices focus and degrades performance since the selected view introduces too many new terms to adequately constrain inference during elaboration. Similarly, containment, if used alone, promotes selecting views that introduce very few new terms. Again, performance is degraded since the overhead of identifying, creating, selecting, and activating views must be repeated too frequently relative to the amount of inference that each view's activation enables; thus inference is overly constrained. Including both the coverage and containment ratios creates a competition of opposing pressures: one for selecting large, encompassing views to promote bountiful inference; one for selecting small, restrictive views to restrain and focus inference. Together these ratios simultaneously facilitate both focusing and stimulating inference and appear to provide a better assessment of relevance than either does alone.

Appraising view interestingness: Interestingness is computed for the set of candidate views in three steps (Figure 4.15): [13]

1. The *absolute interest* for each view is computed by summing the interestingness of each belief contained in the view.

[13]Intuitively, both relevance and interestingness are needed because relevance grounds inference in the new information while interestingness discriminates among beliefs according to their importance within both the general domain and the specific learning situation (i.e., not all beliefs are created equally).

1. absoluteInterest(V) = $\sum$ interestingness(p_i)
   for each belief $p_i \in$ concepts(V)
2. availableInterest(C) = $\sum$ interestingness(p_i)
   for each belief p_i in the union of concepts(V_i) for each candidate view V_i
3. relativeInterest(V) = absoluteInterest(V) $\div$ availableInterest(C)

   V denotes an arbitrary candidate view; C denotes the learning context.

Figure 4.15: Computing the relative interestingness of views

2. The *available interest* for the set of candidate views is computed by summing the interestingness of each belief contained in any candidate view.

3. The *relative interestingness* of each view is computed as the ratio of the available interestingness attributed to the beliefs it contains.

Figure 4.16 presents the computation of relative interest for each of the two candidate views from the example. Virtually all the beliefs contained in the candidate views contain figurative references. Consequently, some of the interestingness heuristics of Figure 4.10 (e.g., those that consider how a belief is supported or what other beliefs it supports) cannot apply. However, other interestingness heuristics (e.g., heuristics *a, f, g, j, k,* and *s* of Figure 4.10) do apply to figurative beliefs. The view $LeafEpidermis_1QuaContainer$ is dominated by beliefs describing processes that move things into and out of the leaf epidermis; many of these beliefs directly reference the leaf epidermis. The view $LeafEpidermis_1QuaCoveringPart$ is dominated by beliefs describing the parts of the leaf; very few of these beliefs reference the epidermis. Consequently, in the context of appraising new information about the leaf epidermis, $LeafEpidermis_1QuaContainer$ is deemed substantially more interesting than $LeafEpidermis_1QuaCoveringPart$.

(a) Absolute interest for the candidate views

Belief	Rule	Interestingness	V_1	V_2
coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)	[f]	0.75	+	+
conduitIn(LeafEpidermis$_1$ LeafLightAcquisition)	[f]	0.75	+	-
conduitIn(LeafEpidermis$_1$ LeafLightDistribution)	[f]	0.75	+	-
conduitIn(LeafEpidermis$_1$ LeafCO_2Acquisition)	[f]	0.75	+	-
conduitIn(LeafEpidermis$_1$ LeafCO_2Distribution)	[f]	0.75	+	-
conduitIn(LeafEpidermis$_1$ LeafTranspiration)	[f]	0.75	+	-
destination(LeafLightAcquisition LeafEpidermis$_1$)	[f]	0.75	+	-
source(LeafLightDistribution LeafEpidermis$_1$)	[f]	0.75	+	-
destination(LeafCO_2Acquisition LeafEpidermis$_1$)	[f]	0.75	+	-
source(LeafCO_2Distribution LeafEpidermis$_1$)	[f]	0.75	+	-
epidermisOf(LeafEpidermis$_1$ Leaf)	[f]	0.75	-	+
epidermis(Leaf LeafEpidermis$_1$)	[f]	0.75	-	+
groundTissue(Leaf LeafMesophyll)	[k]	0.25	-	+
cell(Leaf LeafPhotosyntheticCell)	[k]	0.25	-	+
blade(Leaf LeafBlade)	[k]	0.25	-	+
veins(Leaf LeafVascularNetwork)	[k]	0.25	-	+
petiole(Leaf LeafPetiole)	[k]	0.25	-	+
intercellGap(Leaf LeafIntercellularSpace)	[k]	0.25	-	+
source(LeafLightAcquisition LeafAmbientAtmosphere)	[l]	0.25	+	-
transportee(LeafLightAcquisition Light)	[l]	0.25	+	-
destination(LeafLightDistribution LeafMesophyll)	[l]	0.25	+	-
transportee(LeafLightDistribution Light)	[l]	0.25	+	-
source(LeafCO_2Acquisition LeafAmbientAtmosphere)	[l]	0.25	+	-
transportee(LeafCO_2Acquisition CO_2)	[l]	0.25	+	-
destination(LeafCO_2Distribution LeafMesophyll)	[l]	0.25	+	-
transportee(LeafCO_2Distribution CO_2)	[l]	0.25	+	-
source(LeafTranspiration LeafIntercellularSpace)	[l]	0.25	+	-
destination(LeafTranspiration LeafAmbientAtmosphere)	[l]	0.25	+	-
transportee(LeafTranspiration H_2OVapor)	[l]	0.25	+	-
total		13.25	10.25	3.75

(b) Relative interest for the candidate views

relativeInterest(LeafEpidermis$_1$QuaContainer) = $10.25 \div 13.25 = .7736$

relativeInterest(LeafEpidermis$_1$QuaCoveringPart) = $3.75 \div 13.25 = .283$

(a) The absolute interest for the two candidate views: column *Rule* identifies the heuristic that assesses the belief's interestingness (Figure 4.10); column V_1 identifies the beliefs in *LeafEpidermis*$_1$*QuaContainer*; column V_2 identifies the beliefs in *LeafEpidermis*$_1$*QuaCoveringPart*. **(b)** The relative interest of the two candidate views: *LeafEpidermis*$_1$*QuaContainer* has significantly higher relative interestingness.

Figure 4.16: Computing the interestingness of candidate views

activationScore(LeafEpidermis$_1$QuaContainer) = .1538 * .7736 = .1190
activationScore(LeafEpidermis$_1$QuaCoveringPart) = .2222 * .283 = .0629

activationLevel(LeafEpidermis$_1$QuaContainer) = .1190 ÷ .1819 = .6542
activationLevel(LeafEpidermis$_1$QuaCoveringPart) = .0629 ÷ .1819 = .3458

Figure 4.17: Computing activation levels for the candidate views

Combining relevance and interestingness: The final step in selecting a candidate view is to combine the assessments of relevance and relative interestingness into a single activation score for each candidate view. This *activation score* is computed as the product of its relevance and relative interestingness. The *activation level* of each candidate view is the ratio of its score to the sum of the scores of all the candidate views. The candidate view having the highest score (and hence level) is selected for activation. The beliefs contained in the selected view are instantiated and added to the learning context. Thus the learning context is extended with facts about those concepts in the knowledge base heuristically considered most interesting and relevant to the new information. Figure 4.17 presents the activation computations for each of the two candidate views from the example: the view $LeafEpidermis_1QuaContainer$ has a greater score and is therefore selected for activation.

4.5 Discussion

4.5.1 Why views work

Like endorsements [Coh85], views identify paths of relations that have holistic properties: the property preserved by an endorsement is warranted inference; the property preserved by a view is mutual coherence (e.g., relevance). The ontological bias of views ensures that the beliefs contained in views are

connected. This appears to be a necessary condition for preserving relevance but alone is not sufficient (e.g., since any set of arbitrary beliefs can be extended to be connected by adding the facts $isa(concept_i\ Thing)$ for every $concept_i$ referenced by one of the beliefs). Consequently, views are more than arbitrary collections of connected beliefs: they manifest domain knowledge encoded in the view types about what facts comprise useful contexts in the domain. View types are more than arbitrary patterns of access paths: they manifest domain knowledge about which access paths are useful in a domain and which paths are most useful when considered together.

Using views to extend the learning context with segments of prior knowledge reflects the intuition that humans do not retrieve individual facts from memory as they reason in a domain. Like gestalts or schemas [Sch82], views include beliefs that belong together and exclude beliefs that do not belong. The decisions of what domain situations are useful to consider and what individual facts to include and exclude from these situations are unavoidable for teachers, authors of textbooks, illustrators, etc. As humans gain experience in a domain, they naturally (although perhaps tacitly) learn what situations are useful to consider; they learn how to reason (e.g., what to reason about) as well as what is true within the domain.

Composing views during multiple iterations of the comprehension cycle frees KI from the assumption that some single view will always exist that is sufficient to comprehend the new information (i.e., a perfect view that includes all background knowledge relevant to the new information). The view mechanism is complete with respect to connected sets of domain propositions: a view type can be defined that selects arbitrary connected subgraphs of the knowledge base. Furthermore, with view composition, the learning context can

be extended to include any connected set of propositions in the knowledge base.

4.5.2 Views qua learning bias

The view mechanism in KI solves the problem of focusing attention by determining *what* to reason about. At any point during comprehension, there exists some subset of facts in the learning context that are primitive – not derived as a consequence of other facts in the learning context. Because these primitive facts determine precisely what inferred facts will be established; they control elaboration. Initially, primitive facts comprise only those facts that instantiate the new information. During each subsequent cycle of comprehension, recognition selects a view that determines an extension of the primitive facts. Each extension makes a particular region of implicit knowledge accessible; explicating that region occurs as non-skolemizing rules exhaustively forward chain during elaboration. Each inferred fact is necessarily a consequence of the extension, so the contents of the extension – the set of new primitive facts added to the learning context each cycle – determines precisely those inferences that are completed during that cycle. Thus, the primitive facts of the learning context, which are determined exclusively by the new information and the selected views, completely control the inferences completed during comprehension.

Learning opportunities arise from inferences completed during comprehension. By determining what inferences are completed, the views indirectly determine which learning opportunities become available and, consequently, are detected and exploited. Thus, by controlling inference, selected views are a very significant source of learning bias.

4.5.3 Views qua schemas

Views constitute yet another proposal for implementing schemas [Sch82]; however, they have some distinct advantages over earlier proposals, such as frames. [14]

One relative advantage of views over frames is that frames collect all propositions that directly reference a particular concept (as the first argument) and bundle those propositions together in the frame representing that concept. Thus, the frame of a concept includes all, and only, propositions that directly reference that concept. This has two deficiencies:

1. Frames presume a single description of each concept. Ironically, this violates a venerable adage of knowledge-base design: *"Since one does not usually know in advance what aspect of an object or action is important, it follows that most of the time, a given object will give rise to several different coarse internal descriptions"* [Mar77]. Views facilitate maintaining multiple descriptions of a concept without losing coherence. Each distinct view provides a single, coherent description; the view selection mechanism facilitates activating the most appropriate description for any particular situation.

2. Frames include only propositions that directly reference the represented concepts. Consequently, frames tend to provide partial, incomplete descriptions of concepts. For example, given the facts $producerIn(DairyCow\ CowMilking)$ and $product(CowMilking\ CowMilk)$, those products produced

[14] Here, *frames* denotes the data structure common to knowledge-based systems [BBB+83] rather than the system of structuring knowledge proposed by Minsky [Min81].

from dairy cows cannot be determined by inspecting only the frame representing dairy cows. It is sometimes reasonable to compose a path of two or more predicates into a single predicate (e.g., to compose the path [*producerIn product*] into the single predicate *produces*), but in general it is unreasonable to assume that every useful path through the knowledge base (e.g., every access path appearing in any view type) has been compressed into a single predicate. Views permit paths of propositions that are relevant to, but do not necessarily directly reference, the root concept.

Thus, frames do not permit multiple descriptions of a single concept and do not permit a description of a concept to include indirect propositions. Structuring knowledge with views permits multiple descriptions of a concept, each containing only propositions that are relevant to that description, including relevant propositions that do not necessarily reference the concept directly.

The methods of view creation and instantiation implemented in KI accord with two features in proposed psychological assessments of schemas that have eluded many computational implementations [RSMH86]:

1. *The interdependency of variables:* The binding of one variable should be able to affect the bindings of other variables. This is a natural consequence of instantiating access paths in views; the bindings of one node determine the bindings of the successor nodes along the path.

2. *Constraining variables:* Schema variable constraints should serve two purposes: they restrict the eligible bindings of the variable; and they provide a default value if no bindings are identified. This feature is a natural consequence of the use of figurative references and hypothetical instances. A view type applied to a class contains figurative beliefs. Similarly, each

node variable of an instance-level view assumes a figurative binding when no instance-level binding is available (e.g., Figure 4.1). Each figurative reference identifies a collection of eligible bindings for that node. When an instance-level view is activated, a hypothetical instance is created for any figurative reference it contains. Thus, figurative references both constrain bindings at the instance level and suggest default bindings, in the form of hypothetical instances, when no other bindings exist.

Perhaps the most significant advantage of views as a proposal for implementing schemas is that they are a dynamic, generative method for structuring knowledge. Rather than existing as rigid, static entities, views are dynamically created from view types, as they are needed. This offers several advantages:

1. Structuring knowledge dynamically permits the vast store of an agent's knowledge to reside in an unstructured form. This knowledge remains tacit and unstructured until it is accessed; only as knowledge is accessed need it become structured. Consequently, schemata are transient structures, created dynamically, as needed, rather than stored as static structures in memory. This accords with current trends in theories of human knowledge [Sch82, Sch87, BM77].

2. Structuring knowledge dynamically also permits a natural and seamless evolution in the contents of views as knowledge changes. For example, the view of a leaf epidermis in its role as a container can not include beliefs that reference the leaf cuticle before the knowledge base is extended with new information about the leaf cuticle. However, as that new information is received, and knowledge about the anatomy of the leaf epidermis is extended, all the appropriate views include the new knowledge as they

are subsequently created. Because views are transient and created dynamically, they naturally adapt to evolving knowledge.

3. Developing strategies for structuring knowledge that are independent of the contents of knowledge also accords with a venerable adage in knowledge engineering that separates concerns for the *epistemological adequacy* of the knowledge from concerns for its *heuristic adequacy* [MH69]. The contents of the knowledge (i.e., its epistemological adequacy) can be developed without committing either to particular applications or to ways of accessing the knowledge in order to perform those applications (i.e., its heuristic adequacy). In particular, knowledge of the domain can be formalized and added to the knowledge base without first committing to a specific array of views, view types, or view selection heuristics.

Furthermore, the view mechanism is not a purely formal method, free of domain-specific knowledge. View types represent a kind of domain meta-knowledge that includes knowledge of what contexts or situations in a domain are useful. This meta-knowledge is heuristic; it is exploited in order to guide the use of more traditional domain knowledge that denotes what concepts exist in a domain and what properties are true of those concepts.

Chapter 5

Identifying Deep Consequences of New Information

The first cycle of comprehension considers only the new information and non-skolemizing rules to reveal relatively shallow consequences of the new information. Subsequent cycles consider other segments of relevant prior knowledge to reveal deeper implicit consequences. This chapter illustrates how multiple iterations of the comprehension cycle enable KI to identify and exploit learning opportunities provided by identifying these deep consequences.

Rather than attempting a single, monolithic assessment of what prior knowledge is relevant when presented with new information, KI enters a two-phase cycle of selecting a relevant view to consider and determining the interaction between the new information and the selected prior knowledge. View selection guides elaboration which then facilitates additional view selection. During each cycle, KI determines how the new information interacts with the beliefs contained in the selected view by extending the partial entailment of new and prior knowledge. Then KI determines what prior knowledge not yet considered is relevant to the partial entailment while selecting another view to consider. Each iteration of the cycle begins by selecting a relevant portion of prior knowledge; that selection process is guided by all the inferences established during prior iterations. The cycle continues either until the user intervenes (e.g., to respond to some consequence or suggestion identified by KI)

or until the computational resources expended exceed a threshold. [1]

The first two sections of this chapter describe two iterations of the comprehension cycle while KI performs the example of Figure 1.1. The following three sections describe how KI exploits learning opportunities revealed during these cycles.

5.1 Consequences for the leaf epidermis as a container

Chapter 4 discusses how the view $LeafEpidermis_1QuaContainer$ is selected to extend the learning context. This view contains propositions that describe the leaf epidermis as a container, including the processes that move things (e.g., carbon dioxide, oxygen, water vapor) into and out of the epidermis. Elaboration activates this view to identify consequences of the new information for the beliefs it contains.

Activating a view involves operationalizing the quantified formulae it contains and then adding the resulting facts to the learning context. [2] As each fact is added, non-skolemizing rules are permitted to chain exhaustively; Figure 5.1 presents some of the triggered rules. Activating the view adds twenty-one new facts to the learning context. This, in turn, stimulates the completion of 170 inferences, including seventy consequences of both the new information and the contents of the activated view. Figure 5.2 presents the learning context extended to include this view as well as some of the resulting consequences.

Many of the inferences completed during this cycle of elaboration

[1] Unless otherwise noted, each example described was permitted to continue for three cycles, instantiating the training followed by two extensions of prior skolemizing knowledge.

[2] Quantified formulae in a view are operationalized in precisely the same way as are quantified formulae appearing in the new information (Section 3.4.1).

Rule 12 : *Transport assumes permeability.*
[∀ xyz conduit(x y) & transportee(y z) & unless(impermeableTo(y z)) ⇒ permeableTo(y z)]

Rule 13 : *Permeability to light is translucence.*
[∀ xy permeableTo(x y) & isa(y Light) ⇒ transparency(x Translucent)]

Rule 14 : *Covering parts preserve translucence.*
[∀ xy coveringPart(x y) & transparency(x Translucent) ⇒ transparency(y Translucent)]

Rule 15 : *Compositions preserve translucence.*
[∀ xy transparency(x Translucent) & composedOf(x y)
& unless(intensionalAttribute(y transparency Opaque))
⇒ intensionalAttribute(y transparency Translucent)]

Rule 16 : *Impermeable to type suggests impermeable to.*
[∀ xyz impermeableToType(x y) & isa(z y)
& unless(permeableTo(x z)) ⇒ impermeableTo(x z)]

Rule 17 : *Impermeable conduit restricts emission.*
[∀ xyz isa(x Emission) & conduit(x y) & transportee(x z) & impermeableTo(y z)
⇒ restrictsEmission(y x)]

Rule 18 : *Impermeable conduit restricts intake.*
[∀ xyz isa(x Intake) & conduit(x y) & transportee(x z) & impermeableTo(y z)
⇒ restrictsIntake(y x)]

Rule 19 : *Object restricts emission of.*
[∀ xy restrictsEmission(x y) & transportee(y z) ⇒ restrictsEmissionOf(x z)]

Rule 20 : *Object restricts intake of.*
[∀ xy restrictsIntake(x y) & transportee(y z) ⇒ restrictsIntakeOf(x z)]

Rule 21 : *Impermeable conduit cover part restricts emission.*
[∀ wxyz isa(w Emission) & conduit(w x) & coveringPart(x y) & transportee(w z)
& impermeableTo(y z) ⇒ restrictsEmission(y w)]

Rule 22 : *Impermeable conduit cover part restricts intake.*
[∀ wxyz isa(w Intake) & conduit(w x) & coveringPart(x y) & transportee(w z)
& impermeableTo(y z) ⇒ restrictsIntake(y w)]

Rule 23 : *Transport requires permeability.*
[∀ xyz conduit(x y) & transportee(x z) & impermeableTo(y z) ⇒ status(x Disabled)]

Rule 24 : *Transport assumes conduit contacts source.*
[∀ xyz conduit(x y) & source(x z) & unless(y=z) ⇒ contacts(y z)]

Rule 25 : *Transport assumes conduit contacts destination.*
[∀ xyz conduit(x y) & destination(x z) & unless(y=z) ⇒ contacts(y z)]

Some of the non-skolemizing rules triggered by ground beliefs asserted in the learning context as the view $LeafEpidermis_1\ QuaContainer$ is activated during the second cycle of comprehension.

Figure 5.1: Rules triggered during cycle 2

(a) The learning context after the first cycle

{isa(LeafEpidermis$_1$ LeafEpidermis)	*isa(LeafCuticle$_1$ Cutin)*
isa(LeafCuticle$_1$ LeafCuticle)	*isa(LeafCuticle$_1$ Container)*
coveringPart(LeafEpidermis$_1$ LeafCuticle$_1$)	*isa(LeafCuticle$_1$ Solid)*
composedOf(LeafCuticle$_1$ Cutin)	*transparency(LeafCuticle$_1$ Opaque)*
	covers(LeafCuticle$_1$ LeafEpidermis$_1$)
	impermeableToType(LeafCuticle$_1$ Gas)
	impermeableToType(LeafCuticle$_1$ Liquid)
	impermeableToType(LeafEpidermis$_1$ Gas)
	impermeableToType(LeafEpidermis$_1$ Liquid)}

(b) Additions to the learning context during the second cycle

{isa(LeafLightDistribution$_1$ LeafLightDistribution)	*<transparency(LeafCuticle$_1$ Opaque)>*
isa(LeafLightAcquisition$_1$ LeafLightAcquisition)	*transparency(LeafCuticle$_1$ Translucent)*
isa(LeafAmbientAtmosphere$_1$ LeafAmbientAtmosphere)	*impermeableTo(LeafCuticle$_1$ CO_{21})*
isa(LeafCO_2Acquisition$_1$ LeafCO_2Acquisition)	*impermeableTo(LeafEpidermis$_1$ CO_{21})*
isa(LeafCO_2Distribution$_1$ LeafCO_2Distribution)	*impermeableTo(LeafCuticle$_1$ WaterVapor$_1$)*
isa(LeafTranspiration$_1$ LeafTranspiration)	*impermeableTo(LeafEpidermis$_1$ WaterVapor$_1$)*
isa(LeafMesophyll$_1$ LeafMesophyll)	*impermeableTo(LeafCuticle$_1$ LeafAmbientAtmosphere$_1$)*
isa(Light$_1$ Light)	*impermeableTo(LeafEpidermis$_1$ LeafAmbientAtmosphere$_1$)*
isa(CO_{21} CO_2)	*restrictsIntake(LeafCuticle$_1$ LeafCO_2Acquisition$_1$)*
isa(WaterVapor$_1$ WaterVapor)	*restrictsIntake(LeafEpidermis$_1$ LeafCO_2Acquisition$_1$)*
isa(LeafIntercellularSpace$_1$ LeafIntercellularSpace)	*restrictsIntakeOf(LeafCuticle$_1$ CO_{21})*
conduitIn(LeafEpidermis$_1$ LeafLightDistribution$_1$)	*restrictsIntakeOf(LeafEpidermis$_1$ CO_{21})*
conduitIn(LeafEpidermis$_1$ LeafLightAcquisition$_1$)	*restrictsEmission(LeafCuticle$_1$ LeafTranspiration$_1$)*
conduitIn(LeafEpidermis$_1$ LeafCO_2Acquisition$_1$)	*restrictsEmission(LeafEpidermis$_1$ LeafTranspiration$_1$)*
conduitIn(LeafEpidermis$_1$ LeafCO_2Distribution$_1$)	*restrictsEmissionOf(LeafCuticle$_1$ WaterVapor$_1$)*
conduitIn(LeafEpidermis$_1$ LeafTranspiration$_1$)	*restrictsEmissionOf(LeafEpidermis$_1$ WaterVapor$_1$)*
transportee(LeafLightDistribution$_1$ Light$_1$)	*status(LeafCO_2Acquisition$_1$ Disabled)*
transportee(LeafLightAcquisition$_1$ Light$_1$)	*status(LeafCO_2Distribution$_1$ Disabled)*
transportee(LeafCO_2Acquisition$_1$ CO_{21})	*status(LeafTranspiration$_1$ Disabled)}*
transportee(LeafCO_2Distribution$_1$ CO_{21})	
transportee(LeafTranspiration$_1$ WaterVapor$_1$)	
source(LeafLightDistribution$_1$ LeafEpidermis$_1$)	
source(LeafLightAcquisition$_1$ LeafAmbientAtmosphere$_1$)	
source(LeafCO_2Acquisition$_1$ LeafAmbientAtmosphere$_1$)	
source(LeafCO_2Distribution$_1$ LeafEpidermis$_1$)	
source(LeafTranspiration$_1$ LeafIntercellularSpace$_1$)	
destination(LeafLightDistribution$_1$ LeafMesophyll$_1$)	
destination(LeafLightAcquisition$_1$ LeafEpidermis$_1$)	
destination(LeafCO_2Acquisition$_1$ LeafEpidermis$_1$)	
destination(LeafCO_2Distribution$_1$ LeafMesophyll$_1$)	
destination(LeafTranspiration$_1$ LeafAmbientAtmosphere$_1$)	

(a) The learning context after the first cycle of comprehension. Inferred facts are presented in italics. **(b)** Some additions to the learning context made during the second cycle of comprehension. Inferred facts are presented in italics; retracted facts appear within brackets <...>.

Figure 5.2: Consequences for the leaf epidermis as a container

reveal new learning opportunities. For example, KI establishes that the leaf acquisition of carbon dioxide is disabled (i.e., the event cannot occur). Because this consequence conflicts with the prior expectation that leaves do acquire carbon dioxide, it is considered anomalous. KI analyzes the justification of this conclusion to identify essential assumptions that, if refuted, would refute the anomalous conclusion. [3] KI then creates a memo suggesting knowledge-base modifications to refute the essential assumptions underlying the anomalous conclusion. Furthermore, a weakest-preconditions analysis of the justification of this conclusion suggests that every instance of $LeafCO_2Acquisition$ will be disabled (i.e., since every such event requires gas passing through the leaf epidermis which, as a consequence of the leaf cuticle, is impermeable to gas). KI creates a memo suggesting that the user consider adding this rule to the knowledge base. However, this rule is not asserted by KI since it generalizes an anomalous conclusion.

Some facts established during elaboration are not consequences of the new information but still reveal useful learning opportunities. For example, the first cycle of elaboration establishes that the leaf epidermis is opaque (Figure 3.4, Rule 4). However, while considering the leaf epidermis as a container, elaboration establishes that the leaf epidermis must, in fact, be translucent in order to permit the leaf's acquisition of light (Figure 5.1, rule 13). [4] That the leaf epidermis must be translucent is not a consequence of the new information, yet it is a useful conclusion that exposes an error in prior knowledge. Thus, reasoning about existing concepts in contexts reveals useful tacit knowledge

[3] This analysis is the same as that for resolving inconsistencies, described in Section 3.5.1.

[4] Note, rule 13 overrides rule 4. Handling such priorities among rules is performed by the TMS.

that can be made explicit. In this case, reasoning about the leaf epidermis' transparency while also reasoning about the leaf's acquisition of light exposes a learning opportunity. KI asserts the general rule that the leaf epidermis is translucent rather than opaque and suggests that the user verify this new rule.

5.2 Consequences for the leaf's use of carbon dioxide

The second cycle of comprehension reveals several implicit consequences of the new information for prior knowledge denoting the leaf epidermis in its role as a containers, and it exposes several learning opportunities. In order to more extensively determine how the new and prior knowledge interact, KI selects a second view comprising beliefs deemed relevant to the learning context.

The view selection process described in Chapter 4 is repeated again. Each of the eleven hypotheticals, introduced as $LeafEpidermis_1 QuaContainer$ is activated, plus the original two, are eligible roots of new views for extending the learning context. The eligible views, along with the interestingness estimates for the most specific of these views, are presented in Figure 5.3, and Figure 5.4 presents the candidate views ranked by their activation scores. The view $LeafCO_2Acquisition_1 QuaResourceAttainment$ has the highest score and is therefore selected for activation. Figure 5.5 shows the view type that structures this view.

Activating the selected view results in adding six new hypothetical objects and twenty-seven new facts to the learning context. This stimulates the completion of 173 inferences, including thirty consequences of both the new information and the contents of the activated view. Figure 5.6 identifies some of the rules that are triggered, and Figure 5.7 shows the learning context

(a) Applicable view types for eligible root concepts

Root Concept	View Type	Status
$LeafEpidermis_1$	QuaContainer	*activated*
$LeafEpidermis_1$	QuaCoveringPart	*eligible*
$LeafEpidermis_1$	QuaPhysicalComponent	*not most specific*
$LeafEpidermis_1$	QuaDevelopingThing	*empty*
$LeafCuticle_1$	QuaBiologicalProduct	*empty*
$LeafCuticle_1$	QuaContainer	*empty*
$LeafCuticle_1$	QuaPhysicalComponent	*activated*
$LeafMesophyll_1$	QuaPhysicalComponent	*eligible*
$LeafMesophyll_1$	QuaDevelopingThing	*eligible*
$LeafIntercellularSpace_1$	QuaPhysicalComponent	*eligible*
$LeafLightAcquisition_1$	QuaResourceAttainment	*eligible*
$LeafCO_2Acquisition_1$	QuaResourceAttainment	*eligible*
$LeafTranspiration_1$	QuaResourceUtilization	*eligible*
$Light_1$	QuaResourceAssimilate	*eligible*
$WaterVapor_1$	QuaProduct	*eligible*
$WaterVapor_1$	QuaResourceAssimilate	*eligible*
CO_{21}	QuaResourceAssimilate	*eligible*

(b) Ranking the eligible views

Root Concept	View Type	Root Concept Interestingness	Estimated View Interestingness
$LeafEpidermis_1$	QuaCoveringPart	31.25	7.8125
$LeafCO_2Acquisition_1$	QuaResourceAttainment	9.75	7.3125
$LeafTranspiration_1$	QuaResourceUtilization	9.75	7.3125
$LeafMesophyll_1$	QuaDevelopingThing	4.75	2.375
CO_{21}	QuaResourceAssimilate	8.76	2.19
$WaterVapor_1$	QuaBiologicalProduct	8.51	2.1275
$WaterVapor_1$	QuaResourceAssimilate	8.51	2.1275
$LeafIntercellularSpace_1$	QuaPhysicalComponent	5.01	1.2525
$LeafMesophyll_1$	QuaPhysicalComponent	4.75	1.1875
$LeafLightAcquisition_1$	QuaResourceAttainment	1.00	0.75
$Light_1$	QuaResourceAssimilate	2.52	0.63

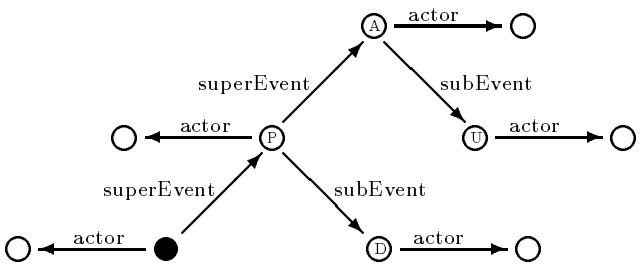
(a) The view types applicable to the eligible root concepts in the learning context and the current status of the view associated with each view type and root concept pair. **(b)** The eligible views ranked by their estimated interestingness scores (computed by multiplying the default interestingness score of the view type, (Figure 4.7), by the interestingness score of the root concept). Only the top ten are admitted as candidate views and created; $Light_1\,QuaResourceAssimilate$ is removed from further consideration.

Figure 5.3: Eligible views for comprehension cycle 3

View	Relevance	Interestingness	Activation Level
LeafCO_2Acquisition$_1$QuaResourceAttainment	0.4451	0.1963	0.2550
$CO_{2\,1}$QuaResourceAssimilate	0.2596	0.3037	0.2301
LeafLightAcquisition$_1$QuaResourceAttainment	0.377	0.1249	0.1374
WaterVapor$_1$QuaResourceAssimilate	0.1094	0.4142	0.1323
WaterVapor$_1$QuaBiologicalProduct	0.1588	0.2607	0.1208
LeafTranspiration$_1$QuaResourceUtilization	0.213	0.1356	0.0843
LeafEpidermis$_1$QuaCoveringPart	0.1367	0.0535	0.0213
LeafMesophyll$_1$QuaPhysicalComponent	0.0769	0.0428	0.0096
LeafIntercellularSpace$_1$QuaPhysicalComponent	0.0866	0.0356	0.0090

The candidate views are created; the view *LeafMesophyll*$_1$ *QuaDevelopingThing* is empty. The remaining views are ranked by activation level (Section 4.4.2).

Figure 5.4: Ranking the candidate views for comprehension cycle 3



Node constraints:
P : ako(P ResourceProvision)
A : ako(A ResourceAssimilation)
U : ako(U ResourceUtilization)
D : ako(D ResourceDistribution)

The view type *QuaResourceAttainment* represented as a semantic-network schema. It contains access paths relevant to representing how a resource is attained, distributed, and utilized. Nodes along the access paths are variables that can bind to knowledge-base constants: constrained nodes are labeled; unconstrained nodes are unlabeled. The shaded node is the position of the root concept.

Figure 5.5: The view type *QuaResourceAttainment*

extended to include this view and some of the resulting consequences.

At this point the autonomous comprehension cycle terminates because the number of executed cycles has reached a threshold parameter; the results of performing knowledge integration are displayed and the user is free to peruse the consequences and suggestion memos generated by KI or to request that the comprehension cycle continue.

Among the most interesting consequences of the new information discovered during elaboration are inferences about the leaf's health. By restricting transpiration, the cuticle inhibits dehydration and facilitates the leaf's good health. This provides a *teleological explanation* of the new information: the cuticle establishes the *biological goal* of facilitating good health. Thus, the "function" of the cuticle, to restrict water loss, has been identified; it explains why leaves have cuticles. However, by restricting the intake of carbon dioxide from the atmosphere, the cuticle inhibits photosynthesis, thereby causing the leaf to starve. This conflicts with prior knowledge which holds that leaves do acquire carbon dioxide from the atmosphere and perform photosynthesis. Therefore, these consequences are labeled as anomalous. [5]

The following three sections discuss in detail how KI exploits several of the most significant learning opportunities afforded by these iterations of the comprehension cycle.

5.3 Resolving anomalies

Anomalous predictions constitute expectation failures, and modifying the knowledge base to resolve these failures constitutes a distinct learning

[5]The determination of anomalies is discussed in Appendix D.

Rule 26 : *Distribution requires attainment.*
[∀ xyz isa(x ResourceDistribution) & isa(y ResourceAttainment) & isa(z ResourceProvision)
& superEvent(x z) & superEvent(y z) & status(y Disabled)
& unless(∃ w isa(w ResourceAttainment) & superEvent(w z) & ¬status(w Disabled))
⇒ status(x Disabled)]

Rule 27 : *Provision requires attainment.*
[∀ xy isa(x ResourceProvision) & isa(y ResourceAttainment) & subEvent(x y)
& status(y Disabled) & unless(∃ z isa(z ResourceAttainment) & subEvent(x z)
& ¬status(z Disabled)) ⇒ status(x Disabled)]

Rule 28 : *Utilization requires provision.*
[∀ xyz isa(x ResourceUtilization) & isa(y ResourceProvision) & resource(x z) & input(y z)
& status(y Disabled) ⇒ status(x Disabled)]

Rule 29 : *Assimilation requires provision.*
[∀ xy isa(x ResourceAssimilation) & isa(y ResourceProvision) & subEvent(x y)
& status(y Disabled) & unless(∃ z isa(z ResourceProvision) & subEvent(x z)
& ¬status(z Disabled)) ⇒ status(x Disabled)]

Rule 30 : *Assimilation requires utilization.*
[∀ xy isa(x ResourceAssimilation) & isa(y ResourceUtilization) & subEvent(x y)
& status(y Disabled) & unless(∃ z isa(z ResourceUtilization) & subEvent(x z)
& ¬status(z Disabled)) ⇒ status(x Disabled)]

Rule 31 : *Denying sugar causes starvation.*
[∀ xyz performs(x y) & isa(y ResourceAttainment) & resource(y z) & isa(z Sugar)
& unless(∃ w isa(w ResourceAttainment) & resource(w v) & isa(v Sugar)
& ¬status(w Disabled)) ⇒ health(x Starving)]

Rule 32 : *Restricting water loss inhibits dehydration.*
[∀ wxyz performs(x y) & resource(x z) & isa(z Water) & restrictsEmission(w x)
& unless(∃ abc performs(a x) & resource(a b) & isa(b Water)
& ¬restrictsEmission(a c)) ⇒ ¬health(x Dehydrating)]

Rule 33 : *Not deteriorating facilitates good health.*
[∀ xy ¬health(x y) & ako(y DeterioratingHealth) ⇒ health(x Facilitated)]

Rule 34 : *Restricting parts suggest restriction role.*
[∀ xyz physicalPart(x y) & restricts(y z) & actor(z x) ⇒ restricts(x z)]

Rule 35 : *CO_2 acquisition is a type of resource attainment.*
[∀ x isa(x LeafCO_2Acquisition) ⇒ isa(x ResourceAttainment)]

Rule 36 : *Photosynthesis is a type of resource attainment.*
[∀ x isa(x LeafPhotosynthesis) ⇒ isa(x ResourceAttainment)]

Rule 37 : *Photosynthesis is a type of resource utilization.*
[∀ x isa(x LeafPhotosynthesis) ⇒ isa(x ResourceUtilization)]

Some of the non-skolemizing rules triggered by ground beliefs asserted in the learning context as the view $LeafCO_2Acquisition_1QuaResourceAttainment$ is activated during the third cycle of comprehension.

Figure 5.6: Rules triggered during cycle 3

{isa(LeafPhotosynthesis$_1$ LeafPhotosynthesis)
isa(LeafCO$_2$Provision$_1$ LeafCO$_2$Provision)
isa(LeafCO$_2$Distribution$_1$ LeafCO$_2$Distribution)
isa(LeafCO$_2$Assimilation$_1$ LeafCO$_2$Assimilation)
isa(Leaf$_1$ Leaf)
isa(Sugar$_1$ Sugar)
superEvent(LeafCO$_2$Acquisition$_1$ LeafCO$_2$Provision$_1$)
subEvent(LeafCO$_2$Provision$_1$ LeafCO$_2$Distribution$_1$)
superEvent(LeafCO$_2$Provision$_1$ LeafCO$_2$Assimilation$_1$)
subEvent(LeafCO$_2$Assimilation$_1$ LeafPhotosynthesis$_1$)
performs(LeafCO$_2$Acquisition$_1$ Leaf$_1$)
performs(LeafCO$_2$Provision$_1$ Leaf$_1$)
performs(LeafCO$_2$Distribution$_1$ Leaf$_1$)
performs(LeafCO$_2$Assimilation$_1$ Leaf$_1$)
performs(LeafPhotosynthesis$_1$ Leaf$_1$)
conduit(LeafCO$_2$Acquisition$_1$ LeafEpidermis$_1$)
conduit(LeafCO$_2$Distribution$_1$ LeafIntercellularSpace$_1$)
conduit(LeafCO$_2$Distribution$_1$ LeafEpidermis$_1$)
conduit(LeafCO$_2$Distribution$_1$ LeafMesophyll$_1$)
source(LeafCO$_2$Acquisition$_1$ LeafAmbientAtmosphere$_1$)
source(LeafCO$_2$Distribution$_1$ LeafEpidermis$_1$)
destination(LeafCO$_2$Acquisition$_1$ LeafEpidermis$_1$)
destination(LeafCO$_2$Distribution$_1$ LeafMesophyll$_1$)
resource(LeafCO$_2$Acquisition$_1$ CO$_{21}$)
resource(LeafCO$_2$Distribution$_1$ CO$_{21}$)
resource(LeafCO$_2$Assimilation$_1$ CO$_{21}$)
resource(LeafCO$_2$Provision$_1$ CO$_{21}$)
input(LeafPhotosynthesis$_1$ Light$_1$)
input(LeafPhotosynthesis$_1$ WaterVapor$_1$)
input(LeafPhotosynthesis$_1$ CO$_{21}$)
product(LeafPhotosynthesis$_1$ Sugar$_1$)
performs(LeafPhotosynthesis$_1$ Leaf$_1$)
occursAt(LeafPhotosynthesis$_1$ LeafMesophyll$_1$)

impermeableToType(Leaf$_1$ Liquid)
impermeableToType(Leaf$_1$ Gas)
impermeableTo(Leaf$_1$ WaterVapor$_1$)
impermeableTo(Leaf$_1$ CO$_{21}$)
impermeableTo(Leaf$_1$ LeafAmbientAtmosphere$_1$)
restrictsIntake(Leaf$_1$ LeafCO$_2$Acquisition$_1$)
restrictsEmission(Leaf$_1$ LeafTranspiration$_1$)
status(LeafCO$_2$Assimilation$_1$ Disabled)
status(LeafPhotosynthesis$_1$ Disabled)
status(LeafCO$_2$Distribution$_1$ Disabled)
status(LeafCO$_2$Provision$_1$ Disabled)
¬*health(Leaf$_1$ Dehydrating)*
health(Leaf$_1$ Facilitated)
health(Leaf$_1$ Starving)
health(Leaf$_1$ Anomalous)}

Some additions to the learning context made during the third cycle of comprehension. Inferred facts are presented in italics.

Figure 5.7: Consequences for the leaf's attainment and use of CO$_2$

opportunity.

5.3.1 Identifying knowledge-base revisions to resolve anomalies

The method for resolving a set of anomalous predictions is similar to the method described in Section 3.5.1 for refuting a single constraint violation. However, it includes an extra step that appraises the interdependencies among the anomalies:

1. The justification of each anomaly is analyzed to identify preferred essential support, and a memo is created that suggests knowledge-base modifications that refute each essential support of the anomalous prediction (Section 3.5.1).

2. The proposed knowledge-base modifications, each of which resolves a particular anomaly, are organized hierarchically. Each node in the hierarchy corresponds to a belief that provides essential support for one or more of the anomalies. Furthermore, the nodes are arranged such that each node provides essential support for each of its descendant nodes in the hierarchy.

The resulting hierarchy reveals the interdependencies among anomalous predictions: the consequences of performing the candidate knowledge-base modifications associated with a particular node include resolving the anomalies associated with each node in the subtree rooted at that node. Those modifications that resolve the greatest number of anomalies appear higher in the hierarchy; those that resolve fewer anomalies appear lower.

Figure 5.8 shows knowledge-base modifications proposed for each anomalous prediction, and Figure 5.9 presents the resulting hierarchy of suggested

1. health($Leaf_1$ Starving)
   :assert [∃ xy isa(x ResourceAttainment) & isa(y Sugar) & performs(x $Leaf_1$) & resources(x y)
   & ¬status(x Disabled)]
   :assert [∃ x isa(x ResourceAttainment) & subEvent($LeafCO_2Provision_1$ x) & ¬status(x Disabled)]
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

2. status($LeafCO_2Acquisition_1$ Disabled)
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

3. status($LeafCO_2Assimilation_1$ Disabled)
   :assert [∃ xy isa(x ResourceProvision) & isa(y ResourceUtilization)
   & subEvent($LeafCO_2Assimilation_1$ x) & subEvent($LeafCO_2Assimilation_1$ y)
   & ¬status(x Disabled) & ¬status(y Disabled)]
   :assert [∃ x isa(x ResourceAttainment) & subevent($LeafCO_2Provision_1$ x) & ¬status(x Disabled)]
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

4. status($LeafCO_2Distribution_1$ Disabled)
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

5. status($LeafCO_2Provision_1$ Disabled)
   :assert [∃ x isa(x ResourceAttainment) & subEvent($LeafCO_2Provision_1$ x) & ¬status(x Disabled)]
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

6. status($LeafPhotosynthesis_1$ Disabled)
   :assert [∃ x isa(x ResourceAttainment) & subEvent($LeafCO_2Provision_1$ x) & ¬status(x Disabled)]
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

7. status($LeafTranspiration_1$ Disabled)
   :assert [∃ x portal($LeafEpidermis_1$ x) & ¬covers($LeafCuticle_1$ x)]
   :assert [partiallyCovers($LeafCuticle_1$ $LeafEpidermis_1$)]

The anomalous predictions established during elaboration. For each anomaly, KI identifies knowledge-base modifications that, if made, would resolve the anomaly (i.e., the modifications would place the anomalous prediction in region 1 of Figure 2.3).

Figure 5.8: Anomalous predictions and suggested resolutions

knowledge-base modifications. This hierarchy guides the user through what would otherwise be an unwieldy set of independent candidate knowledge-base modifications to the select few that would have the greatest beneficial effect. The user can search the tree, beginning with the root node. Any suggested knowledge-base modifications selected by the user will resolve the anomalies associated with all subordinate nodes in the hierarchy. Pushing new anomalies into the hierarchy, indexed by their candidate fixes, naturally reveals the underlying common "root causes" of groups of interdependent anomalous predictions. The hierarchy thus elucidates the interdependencies among anomalies; it achieves what might be called "anomaly reduction" – identifying the underlying common reasons for establishing (and the common fixes for resolving) anomalous predictions.

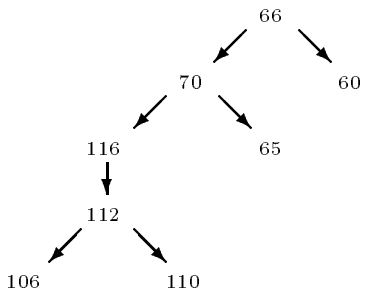
In the example, the fact $impermeableToType(LeafEpidermis_1\ Gas)$, while not deemed anomalous itself, provides essential support to all the anomalous predictions, and retracting this fact would resolve all the anomalies. This fact relies on the assumptions that the cuticle completely covers the leaf epidermis and that the epidermis has no portals not also covered by the cuticle (Figure 3.9a, and rules 8 and 9 of Figure 3.4). Thus, these two assumptions provide essential support for all the anomalous predictions. In accordance with Figure 1.1, KI presents the user with the two alternative suggestions of refuting these assumptions. The user accepts the suggestion that, in fact, the leaf epidermis does have portals.

When this knowledge-base revision is accepted, KI prompts the user for the name of the epidermis portals (i.e., the name of the concept that binds with the variable x in the suggested revision; see Figure 5.9, Suggestion 66).

(a) Nodes in the resolution hierarchy

Suggestion Identifier	Suggestion Effect
Suggestion 66	Refute [impermeableToType(LeafEpidermis$_1$ Gas)]
Suggestion 70	Refute [status(LeafCO$_2$Acquisition$_1$ Disabled)]
Suggestion 116	Refute [status(LeafCO$_2$Provision$_1$ Disabled)]
Suggestion 112	Refute [status(LeafPhotosynthesis$_1$ Disabled)]
Suggestion 106	Refute [status(LeafCO$_2$Assimilation$_1$ Disabled)]
Suggestion 110	Refute [health(Leaf$_1$ Anomalous)]
Suggestion 65	Refute [status(LeafCO$_2$Distribution$_1$ Disabled)]
Suggestion 60	Refute [status(LeafTranspiration$_1$ Disabled)]

(b) The hierarchy



(c) The proposed knowledge-base modifications

Suggestion 66 Refute [impermeableToType(LeafEpidermis$_1$ Gas)]
:assert [∃ (x) portal(LeafEpidermis$_1$ x) & ¬covers(LeafCuticle$_1$ x)]
:assert [partiallyCovers(LeafCuticle$_1$ LeafEpidermis$_1$)]

Suggestion 116 Refute [status(LeafCO$_2$Provision$_1$ Disabled)]
:assert [∃ (x) isa(x ResourceAttainment) & subEvent(LeafCO$_2$Provision$_1$ x) & ¬status(x Disabled)]

Suggestion 106 Refute [status(LeafCO$_2$Assimilation$_1$ Disabled)]
:assert [∃ (xy) isa(x ResourceProvision) & isa(y ResourceUtilization)
& subEvent(LeafCO$_2$Assimilation$_1$ x) & subEvent(LeafCO$_2$Assimilation$_1$ y)
& ¬status(x Disabled) & ¬status(y Disabled)]

Suggestion 110 Refute [health(Leaf$_1$ HealthAnomalous)]
:assert [∃ (xy) isa(x ResourceAttainment) & isa(y Sugar) & performs(x Leaf$_1$) & resources(x y)
& ¬status(x Disabled)]

(a) The nodes in the hierarchy of suggested resolutions for anomalous predictions. Each node depicts a set of alternative knowledge-base modifications. The node identifiers include integers indicating their creation order (e.g., Suggestion 65 was created before Suggestion 66). **(b)** Nodes in the hierarchy are indexed such that the knowledge-base modifications associated with a non-leaf node apply to all of its descendant nodes. **(c)** The alternative knowledge-base modifications proposed at each node to resolve the anomalies included in the sub-tree rooted at that node.

Figure 5.9: Hierarchy of suggested anomaly resolutions

(a) The revision stated in the input language

```
(&  (LeafEpidermis  (portal  (Stomata)))
    (¬  (LeafCuticle  (covers  (Stomata)))))
```

(b) The interpretation of the revision

Rule H : *Each leaf cuticle has (at least one) stoma as a portal.*
[∀ (x) isa(x LeafCuticle) ⇒ ∃ (y) isa(y Stomata) & portal(x y)]

Rule I : *Leaf cuticles have only stomata as portals.*
[∀ (xy) isa(x LeafCuticle) & portal(x y) ⇒ isa(y Stomata)]

Rule J : *Each stoma is a portal in a leaf epidermis.*
[∀ (x) isa(x Stomata) ⇒ ∃ (y) isa(y LeafCuticle) & portal(y x)]

Rule K : *Stomata are portals in only leaf epidermises.*
[∀ (xy) isa(x Stomata) & portal(y x) ⇒ isa(y LeafEpidermis)]

Rule L : *Each stoma is not covered by leaf cuticle.*
[∀ (x) isa(x Stomata) ⇒ ¬∃ (y) isa(y LeafCuticle) & covers(y x)]

Rule M : *Leaf cuticles do not cover stomata.*
[∀ (xy) isa(x LeafCuticle) & covers(x y) ⇒ ¬isa(x Stomata)]

Rule N : *Stomata are a type of portal.*
[∀ (x) isa(x Stomata) ⇒ isa(x Portal)]

Rule O : *Stomata are components of botanical organisms.*
[∀ (x) isa(x Stomata) ⇒ isa(x BotanicalOrganismComponent)]

Fact P : *The class of stomata is a type of tangible object.*
isa(Stomata TangibleObjectType)

(a) The solicited revision presented as a semantic network encoded as nested lists (i.e., in the input language). **(b)** The interpretation of the revision presented as first-order axioms (i.e., in the representation language).

Figure 5.10: Interpreting the solicited knowledge-base revision

The user specifies that these portals are called *Stomata*. [6] KI responds by first constructing a training specification for this knowledge-base modification stated in the input language (Figure 5.10a), then interpreting this training specification (Figure 5.10b), and finally reinvoking the comprehension cycle for the resulting interpretation.

5.3.2 Propagating the revision throughout the learning context

KI extends the learning context with facts that instantiate the knowledge-base revision (Figure 5.11a). As these facts are added, they trigger non-skolemizing rules that propagate their consequences throughout the learning context. Figure 5.11b lists some of the consequences of the revision, and Figure 5.12 presents some of the triggered rules.

As expected, the revision retracts $impermeableToType(LeafEpidermis_1$ *Gas*) and all of its anomalous consequences. Thus, in accordance with Figure 1.1, KI has identified anomalous consequences of the new information, determined two underlying assumptions of these anomalous consequences, and solicited a knowledge-base revision that retracts the underlying assumptions and resolves the anomalies. Furthermore, KI has integrated the revision and demonstrated to the user that the revision has successfully resolved the anomalies (while not introducing others).

[6] KI also prompts the user to verify both that *Stomata* is being added to the knowledge base as a new concept and that, in general, all leaf epidermises have stomata rather than just the particular leaf epidermis under consideration.

(a) Instantiating the revision

{isa(Stomata$_1$ Stomata)
portal(LeafEpidermis$_1$ Stomata$_1$)
¬covers(LeafCuticle$_1$ Stomata$_1$)}

(b) Inferences completed while instantiating the revision

{*conduitIn(Stomata$_1$ LeafCO$_2$Acquisition$_1$)*
conduitIn(Stomata$_1$ LeafCO$_2$Distribution$_1$)
conduitIn(Stomata$_1$ LeafTranspiration$_1$)
permeableTo(Stomata$_1$ CO$_{21}$)
permeableTo(Stomata$_1$ WaterVapor$_1$)
permeableTo(LeafEpidermis$_1$ CO$_{21}$)
permeableTo(LeafEpidermis$_1$ WaterVapor$_1$)
¬impermeableTo(Stomata$_1$ CO$_{21}$)
¬impermeableTo(Stomata$_1$ WaterVapor$_1$)
¬impermeableTo(LeafEpidermis$_1$ CO$_{21}$)
¬impermeableTo(LeafEpidermis$_1$ WaterVapor$_1$)
controlsIntake(Stomata$_1$ LeafCO$_2$Acquisition$_1$)
controlsIntakeOf(Stomata$_1$ CO$_{21}$)
controlsEmission(Stomata$_1$ LeafTranspiration$_1$)
controlsEmissionOf(Stomata$_1$ WaterVapor$_1$)
facilitates(Stomata$_1$ LeafCO$_2$Distribution$_1$)
facilitates(Stomata$_1$ LeafTranspiration$_1$)
facilitates(Stomata$_1$ LeafCO$_2$Acquisition$_1$)
facilitates(LeafEpidermis$_1$ LeafCO$_2$Distribution$_1$)
facilitates(LeafEpidermis$_1$ LeafTranspiration$_1$)
facilitates(LeafEpidermis$_1$ LeafCO$_2$Acquisition$_1$)
facilitates(Leaf$_1$ LeafCO$_2$Distribution$_1$)
facilitates(Leaf$_1$ LeafTranspiration$_1$)
facilitates(Leaf$_1$ LeafCO$_2$Acquisition$_1$)
status(LeafCO$_2$Acquisition$_1$ Facilitated)
status(LeafCO$_2$Distribution$_1$ Facilitated)
status(LeafCO$_2$Provision$_1$ Facilitated)
status(LeafPhotosynthesis$_1$ Facilitated)
status(LeafTranspiration$_1$ Facilitated)
health(Leaf$_1$ Facilitated)

<*impermeableTo(LeafEpidermis$_1$ CO$_{21}$)*>
<*impermeableTo(LeafEpidermis$_1$ WaterVapor$_1$)*>
<*impermeableTo(LeafEpidermis$_1$ LeafAmbientAtm$_1$)*>
<*impermeableToType(LeafEpidermis$_1$ Gas)*>
<*impermeableToType(LeafEpidermis$_1$ Liquid)*>
<*impermeableTo(Leaf$_1$ CO$_{21}$)*>
<*impermeableTo(Leaf$_1$ WaterVapor$_1$)*>
<*impermeableTo(Leaf$_1$ LeafAmbientAtmosphere$_1$)*>
<*impermeableToType(Leaf$_1$ Gas)*>
<*impermeableToType(Leaf$_1$ Liquid)*>
<*health(Leaf$_1$ Starving)*>
<*health(Leaf$_1$ Anomalous)*>
<*status(LeafCO$_2$Acquisition$_1$ Disabled)*>
<*status(LeafCO$_2$Provision$_1$ Disabled)*>
<*status(LeafCO$_2$Distribution$_1$ Disabled)*>
<*status(LeafPhotosynthesis$_1$ Disabled)*>
<*status(LeafCO$_2$Assimilation$_1$ Disabled)*>
<*status(LeafTranspiration$_1$ Disabled)*>}

Note: *LeafAmbientAtm$_1$* abbreviates *LeafAmbientAtmosphere$_1$*

(a) The facts added to the learning context as the revision is instantiated. **(b)** Some consequences of the revision. Inferred facts are presented in italics; retracted facts appear within brackets <...>.

Figure 5.11: Consequences of the knowledge-base revision

Rule 38 : *Portal assumes conduit role.*
[∀ (vwxyz) conduit(v w) & portal(w x) & physicalPart(w y) & transportee(v z)
& impermeableTo(y z) & unless(impermeableTo(y z)) ⇒ conduit(v x)]

Rule 39 : *Permeable portal facilitates transport.*
[∀ (vwxyz) conduit(v w) & portal(w x) & physicalPart(w y) & transportee(v z)
& impermeableTo(y z) & unless(impermeableTo(x z)) ⇒ facilitates(x v)]

Rule 40 : *Permeable portal controls emission.*
[∀ (vwxyz) isa(w Emission) & conduit(v w) & portal(w x) & physicalPart(w y)
& resource(v z) & impermeableTo(y z) & unless(impermeableTo(x z))
⇒ controlsEmission(z v)]

Rule 41 : *Permeable portal controls intake.*
[∀ (vwxyz) isa(w ResourceIntake) & conduit(v w) & portal(w x) & physicalPart(w y)
& resource(v z) & impermeableTo(y z) & unless(impermeableTo(x z))
⇒ controlsIntake(z v)]

Rule 42 : *Permeability enables transport.*
[∀ (xyz) conduit(x y) & transportee(x z) & permeableTo(y z) & unless(status(x Disabled))
⇒ status(x Enabled)]

Rule 43 : *Facilitators facilitate events.*
[∀ (xy) facilitates(x y) & unless(status(y Disabled)) ⇒ status(y Facilitated)]

Rule 44 : *Facilitating parts suggest facilitator role.*
[∀ (xyz) facilitates(x y) & part(z x) & actor(y z) ⇒ facilitates(z y)]

Rule 45 : *Restricting emission facilitates other uses.*
[∀ (vwxyz) restrictsEmission(v w) & transportee(w x) & isa(y ResourceUtilization)
& resource(y x) & performs(y v) & unless(w = y) & unless(status(y Disabled))
⇒ status(y Facilitated)]

Rule 46 : *Facilitating attainment facilitates provision.*
[∀ (xy) isa(x ResourceAttainment) & isa(y ResourceProvision) & status(x Facilitated)
& subEvent(y x) & unless(status(y Disabled)) ⇒ status(y Facilitated)]

Rule 47 : *Facilitating provision facilitates utilization.*
[∀ (xyz) isa(x ResourceProvision) & isa(y ResourceAssimilation)
& isa(z ResourceUtilization) & status(x Facilitated) & subEvent(y x)
& subEvent(y z) & unless(status(z Disabled)) ⇒ status(z Facilitated)]

Rule 48 : *Facilitating essential resource use facilitates health.*
[∀ (xyz) isa(x ResourceUtilization) & status(x Facilitated) & resource(x y)
& isa(y EssentialPlantResource) & performs(x z) ⇒ health(z Facilitated)]

Some of the non-skolemizing rules triggered by ground beliefs asserted in the learning context as the solicited knowledge-base revision is instantiated.

Figure 5.12: Rules triggered by instantiating the revision

5.3.3 Propagating the revision through the adaptation methods

The scope of the TMS (Section 3.4.4) is limited to inferences completed by the knowledge-base's inference engine. Significantly, this excludes the results of KI performing adaptation during each execution of the comprehension cycle. For example, during the first cycle, elaboration establishes that the hypothetical leaf epidermis is impermeable to gases, and adaptation determines that the justification of this conclusion warrants asserting the general rule that every leaf epidermis is impermeable to gases (Section 3.5.2). As a consequence of instantiating the revision, the TMS retracts the fact that the hypothetical leaf epidermis is impermeable to gases but leaves the general rule intact.

To handle revisions properly, KI implements its own 'truth maintenance' facility for the new rules that it defines during adaptation. Each new rule is indexed by the propositions that support it (e.g., those facts that triggered the adaptation methods that created it). When these supporting propositions are retracted by the TMS, KI re-evaluates the rule to determine whether or not it is still warranted, and, if not, the rule is retracted (if it has already been autonomously asserted), and suggestions that reference it are removed from the memos queued for the user.

Extending the standard TMS to handle beliefs (e.g., rules) established by the learning system has turned out to be a very significant feature of KI. Each adaptation method (many of which are not truth preserving) must record the facts and rules in the knowledge base that warrant every new belief established by the method. While this issue of extending a TMS to handle the results of learning systems has been largely unexplored in machine learning research, it remains an essential issue for any incremental learning system that exploits nonmonotonic knowledge or knowledge that may be modified.

5.4 Teleological learning

Figure 1.1 is offered as an example of learning that exemplifies knowledge integration. Among the most interesting learning behaviors demonstrated by this example invovles teleological learning: acquiring descriptions of purprose. Teleological learning plays an essential role in the acquisition of knowledge in many domains. In the biological sciences teleology explains the structure of organisms in terms of their physiological functions. [7] Differences among the anatomies and physiologies of various species can be understood in terms of their relative advantages for survivability. The very appeal of natural selection and evolution as theories in the biological sciences is that they offer such explanations. Similarly, in design domains, teleology explains the design of artifacts (i.e., the intended structure and behavior of their parts) in terms of their purposes (i.e., the intended behaviors or functions of the artifacts) [Fra93]. (Additional discussion of the importance of teleological knowledge is included in sections 1.2.1 and 7.4.4.) This section explains how the teleological learning behavior illustrated in Figure 1.1 is achieved by KI.

In the example, elaboration reveals that the leaf cuticle enhances the leaf's physiology by restricting water loss through transpiration. KI recognizes this as a "teleological" consequence of the new information: the physiological benefit of moderating water loss explains why the leaf has a cuticle. Recognizing that a domain goal has been established triggers teleological learning.

[7] The distinction between behavior and function are often conflated in the literature [Kui85, Fra93]. Here, function is a subset of an entity's behavior that contributes to its achieving a domain goal. A teleological description is one that acribes purpose; it associates an entity with one of its functions and the domain goal (or subgoal) facilitated by that function.

5.4.1 Representing goals

The predicate *goalPredicate* identifies predicates that denote domain goals of the instances of a collection. Initially, the only assertions in the knowledge base referencing *goalPredicate* are:

goalPredicate(LivingObject hasPhysiologicalGoal)
goalPredicate(LivingObject hasPhysiologicalFunction)
goalPredicate(OrganismComponent hasPhysiologicalFunction)

and the only assertion in the knowledge base that establishes a domain goal is:

hasPhysiologicalGoal(LivingObject health Facilitated)

This denotes that it is a "goal" (in particular, a physiological goal) of all living objects to promote their own good health. [8] Initially, there are no assertions referencing *hasPhysiologicalFunction*. The knowledge base has therefore been seeded with only the barest of knowledge about physiological goals and functions in botany.

Each inference completed during elaboration is evaluated to determine if it satisfies some domain goal. In the example, elaboration reveals that the leaf cuticle restricts water loss through transportation. By inhibiting dehydration and preserving water for other uses (e.g., photosynthesis) the leaf cuticle benefits the health of the leaf. A physiological goal is thus established since the leaf, an instance of *LivingObject*, has the goal of facilitating its own good health. Establishing domain goals enables learning from *teleological explanations.*

[8] This does not commit to the position that physiological goals really exist (i.e., that each living thing really "has the goal" of promoting its own health). Rather, this scheme simply reflects the fact that humans often conceive of such goals when considering biological domains. When we study or teach biological subject matter, we interpret or explain many aspects of the domain by postulating the existence of physiological goals in order to impose structure on the domain and render it more explainable.

5.4.2 Identifying teleological explanations

A teleological explanation motivates one or more properties of an object by identifying the beneficial consequences of those properties. One common form of teleological explanation justifies the structural properties of an object by establishing how the behavior of its components contribute to its goals; that is, explaining a thing's *structure* by its *function*. In biological sciences it is common to explain the structure of a living thing by determining how its components contribute to achieving its physiological goals. For example, a teleological explanation of why a plant has leaves is that the leaves enable the plant to feed itself: the leaves generate, through photosynthesis, the essential sugars required to sustain the plant. Furthermore, it is natural to interpret those behaviors that directly contribute to achieving some physiological goal as *physiological functions* (e.g., performing photosynthesis is a physiological function of plant leaves).

Whenever an inference is made that establishes a physiological goal, KI analyzes the supporting explanation to determine if it is teleological. An explanation is *teleological* if structural (i.e., the partonomic or compositional) properties of the beneficiary [9] support establishing a physiological goal. Identifying a teleological explanation involves three steps:

1. Determine that the explanation establishes a domain goal. For example, if the fact established by the explanation is $p(x\ y)$, then

   $$goalPredicate(z\ q)\ \&\ q(z\ p\ y)\ \&\ isa(x\ z)$$

   must be true for some collection z and predicate q.

[9] The *beneficiary* is the thing whose physiological goals are established.

2. Prune each sub-explanation that establishes a structural-property denoting fact and that does not reference any structural properties among its own sub-explanations (e.g., Figure 5.13).

3. Identify the pruned explanation's weakest preconditions (Figure 5.14a).

4. Identify the essential structural properties, those propositions in the weakest preconditions that denote structural properties of the beneficiary (e.g., those structural components of $Leaf_1$; see Figure 5.14b).

The first step simply ensures that the fact established by the explanation is a domain goal. The second step removes from consideration the purely logical support for propositions denoting structural properties. This prevents structural preconditions from being replaced by non-structural preconditions during the weakest-preconditions analysis performed during the third step. When the weakest preconditions do include structural properties of the beneficiary, then the explanation is considered to be teleological: establishing this goal explains, teleologically (and abductively), why the beneficiary has the designated structural properties.

In the example, there are 134 explanations of $health(Leaf_1\ Facilitated))$, thirty-six of which are teleological. However, only eight have unique sets of weakest preconditions, and the explanations with redundant weakest preconditions are omitted from further consideration.

Figure 5.13 presents one teleological explanation of $health(Leaf_1\ Facilitated)$. Pruned from this inference graph are all the subgraphs that establish structural properties that are supported by other structural properties. This guarantees that the weakest preconditions (Figure 5.14a) of the explanation

after pruning include all structural properties referenced in the original explanation that are not simply consequences of other structural properties. For example, the fact $coveringPart(LeafEpidermis_1\ LeafCuticle_1)$ does itself have an explanation: it is a consequence of $isa(LeafEpidermis_1\ LeafEpidermis)$ (Figures 3.9a and Rule A of 3.2c). However, it appears as a leaf-node after the inference graph has been pruned (Figure 5.13). Similarly, each essential structural property (Figure 5.14b) is also a consequence of the fact $isa(LeafEpidermis_1\ LeafEpidermis)$, but after pruning the inference graph they appear as unexplained facts, as leaf nodes. This pruning step is required to preserve the *explanatory coherence* of the inference graph. The purpose of this adaptation method is to explain the structure of an object by identifying how its structural properties establish its goals. Without this pruning step, the structural properties (Figure 5.14b) would be obscurred during the weakest-precondition analysis.

5.4.3 Generalizing teleological explanations

Each teleological explanation identified by KI suggests a reason for why an object has a particular structure. One way that KI attempts to learn from a teleological explanation is by generalizing the scope of the explanation to apply to other beneficiaries. This requires determining which other domain objects (e.g., organisms, components of organisms) would also benefit from the structural properties identified by the teleological explanations. KI then conjectures that these candidate beneficiaries, in fact, share the explained structure.

In generalizing a teleological explanation, care must be given to avoid over-generalizing. While the antecedent-regression methods of explanation-

$health(Leaf_1\ Facilitated)$
$\Leftarrow_{48}\ performs(Leaf_1\ LeafPhotosynthesis_1)$
$isa(LeafPhotosynthesis_1\ ResourceUtilization)$
$resource(LeafPhotosynthesis_1\ WaterVapor_1)$
$isa(WaterVapor_1\ EssentialPlantResource)$
$status(LeafPhotosynthesis_1\ Facilitated)$
$\Leftarrow_{45}\ performs(Leaf_1\ LeafPhotosynthesis_1)$
$isa(LeafPhotosynthesis_1\ ResourceUtilization)$
$resource(LeafPhotosynthesis_1\ WaterVapor_1)$
$transportee(LeafTranspiration_1\ WaterVapor_1)$
$restrictsEmission(Leaf_1\ LeafTranspiration_1)$
$\Leftarrow\ isa(LeafTranspiration_1\ Emission)$
$restricts(Leaf_1\ LeafTranspiration_1)$
$\Leftarrow_{34}\ physicalPart(Leaf_1\ LeafEpidermis_1)$
$\Leftarrow_{11}\ coveringPart(Leaf_1\ LeafEpidermis_1)$
$\Leftarrow_{11}\ epidermis(Leaf_1\ LeafEpidermis_1)$
$actorIn(Leaf_1\ LeafTranspiration_1)$
$\Leftarrow_{11}\ performs(Leaf_1\ LeafTranspiration_1)$
$restricts(LeafEpidermis_1\ LeafTranspiration_1)$
$\Leftarrow_{34}\ physicalPart(LeafEpidermis_1\ LeafCuticle_1)$
$\Leftarrow_{11}\ coveringPart(LeafEpidermis_1\ LeafCuticle_1)$
$actorIn(LeafEpidermis_1\ LeafTranspiration_1)$
$\Leftarrow_{11}\ conduitIn(LeafEpidermis_1\ LeafTranspiration_1)$
$restricts(LeafCuticle_1\ LeafTranspiration_1)$
$\Leftarrow_{11}\ restrictsEmission(LeafCuticle_1\ LeafTranspiration_1)$
$\Leftarrow_{21}\ isa(LeafTranspiration_1\ Emission)$
$conduitIn(LeafEpidermis_1\ LeafTranspiration_1)$
$coveringPart(LeafEpidermis_1\ LeafCuticle_1)$
$transportee(LeafTranspiration_1\ WaterVapor_1)$
$impermeableTo(LeafCuticle_1\ WaterVapor_1)$
$\Leftarrow_{16}\ isa(WaterVapor_1\ Gas)$
$impermeableToType(LeafCuticle_1\ Gas)$
$\Leftarrow_{6}\ isa(LeafCuticle_1\ Cutin)$
$\Leftarrow_{5}\ composedOf(LeafCuticle_1\ Cutin)$

One teleological explanation of $health(Leaf_1\ Facilitated)$. $p\ \Leftarrow_n\ q$ denotes that p follows from rule n triggered by q; the referenced rules are presented in Figures 3.4, 3.7, 5.1, 5.6, and 5.12. Implications with no subscript denote obvious rules that have not been presented.

Figure 5.13: A teleological explanation

(a) The weakest preconditions

{epidermis($Leaf_1$ $LeafEpidermis_1$)
coveringPart($LeafEpidermis_1$ $LeafCuticle_1$)
composedOf($LeafCuticle_1$ Cutin)
conduitIn($LeafEpidermis_1$ $LeafTranspiration_1$)
transportee($LeafTranspiration_1$ $WaterVapor_1$)
performs($Leaf_1$ $LeafTranspiration_1$)
performs($Leaf_1$ $LeafPhotosynthesis_1$)
input($LeafPhotosynthesis_1$ $WaterVapor_1$)
isa($WaterVapor_1$ Gas)
isa($WaterVapor_1$ EssentialPlantResource)
isa($LeafTranspiration_1$ Emission)
isa($LeafPhotosynthesis_1$ ResourceUtilization)}

(b) The essential structural properties

{epidermis($Leaf_1$ $LeafEpidermis_1$)
coveringPart($LeafEpidermis_1$ $LeafCuticle_1$)
composedOf($LeafCuticle_1$ Cutin)}

(a) The weakest preconditions of the teleological explanation presented in Figure 5.13. **(b)** The subset of weakest preconditions that denote structural (e.g., partonomic or compositional) properties of the beneficiary.

Figure 5.14: Identifying a teleological explanation

based learning (EBL) preserve logical validity, they do not necessarily preserve teleological validity. In other words, aspects of the explanation that endowed it with teleological properties can be lost during the generalization process.

For example, consider a hunter with a gun who shoots a quail. The hunter can eat the quail because the hunter possesses the quail (by shooting it) and the quail is dead (by being shot by the hunter). This teleological explanation suggests that perhaps the hunter owns the gun in order to be able to shoot game. However, this explanation logically supports a generalization in which the hunter shooting the quail is not the same one as the hunter who subsequently eats it (e.g., a hunter obtains a quail by buying it). Such a generalization offers no account for *why* a hunter might own a gun. Thus, when

trying to determine, in general, why someone might own a gun, generalizations that allow for some other agent to provide the game should be inhibited.

To avoid such over-generalization, additional constraints are imposed on the generalization process that equate terms in the explanation. Specifically, all references to the beneficiary are equated, and, for each structural component of the beneficiary, all references to that component are equated. These constraints anchor terms appearing as variables in the generalized explanation to preserve the relationship between the beneficiary and its components. They inhibit over-generalization (in a coherence sense, not a logical sense).

To illustrate this, Figure 5.15 presents the overly general rule that results when traditional explanation-based generalization is applied to the teleological explanation in Figure 5.13 (the variables have been renamed in Figure 5.15 for readability). Significantly, this rule fails to reflect the teleological structure of the explanation from which it was generated. Lost is the requirement that what facilitates achieving the goal are structural properties of the beneficiary, and not of some other agent. Instead, the rule indicates that the structural aspects of some arbitrary entity facilitate the beneficiary's goal. Thus, the rule fails to relate the beneficiary's structural properties to the achievement of the beneficiary's goal: the explanatory coherence of the explanation (i.e., its teleological quality) has been lost during generalization.

To preserve the explanatory coherence during generalization, all references in the ground explanation (Figure 5.13) to terms denoting the beneficiary (e.g., $Leaf_1$) are equated; for each structural component of the beneficiary (e.g., $LeafEpidermis_1$ and $LeafCuticle_1$) all references to that component are equated as well. Consequently, variables referenced in the general rule presented in Figure 5.15 are equated as follows:

```
health(?Beneficiary Facilitated)
⇐ performs(?Beneficiary ?ResourceUtilization)
   & input(?ResourceUtilization ?EssentialResource)
   & input(?ResourceUtilization ?OtherResource)
   & transportee(?Transpiration ?OtherResource)
   & transportee(?Transpiration ?WaterVapor)
   & performs(?ShootOrgan ?ResourceUtilization)
   & performs(?ShootOrgan ?Transpiration)
   & epidermis(?ShootOrgan ?Epidermis)
   & conduitIn(?Epidermis ?Transpiration)
   & conduitIn(?OtherPart ?Transpiration)
   & coveringPart(?Epidermis ?Cuticle)
   & coveringPart(?OtherPart ?Cuticle)
   & composedOf(?Cuticle Cutin)
   & isa(?EssentialResource EssentialPlantResource)
   & isa(?Water LiquidTangibleStuff)
   & isa(?Transpiration Emission)
   & isa(?ResourceUtilization ResourceUtilization)
```

The prefix "?" denotes terms that are variables

Figure 5.15: Rule compiled via EBL (no teleological structure)

1. ?Beneficiary ≡ ?ShootOrgan
2. ?Epidermis ≡ ?OtherPart

The resulting general rule (Figure 5.16) retains the teleological structure of the original explanation since it identifies how structural components of the beneficiary contribute to achieving one of its physiological goals.

Generalizing and compiling a teleological explanation produces a general rule that identifies conditions under which a domain goal is established. The resulting general rule is then analyzed to determine what other domain entities (e.g., organisms, organism components) would benefit from the structural properties (e.g., having a cuticle) that enabled the beneficiary (e.g., the leaf) to satisfy the domain goal. To identify other candidate beneficiaries, the antecedent of the general rule is partitioned into two sets:

1. The *endowments* comprise antecedent propositions that do reference structural predicates.

```
health(?Beneficiary Facilitated)
⇐ performs(?Beneficiary ?ResourceUtilization)
   & performs(?Beneficiary ?Transpiration)
   & epidermis(?Beneficiary ?Epidermis)
   & coveringPart(?Epidermis ?Cuticle)
   & composedOf(?Cuticle Cutin)
   & conduitIn(?Epidermis ?Transpiration)
   & transportee(?Transpiration ?WaterVapor)
   & transportee(?Transpiration ?OtherResource)
   & input(?ResourceUtilization ?OtherResource)
   & input(?ResourceUtilization ?EssentialResource)
   & isa(?WaterVapor Gas)
   & isa(?EssentialResource EssentialPlantResource)
   & isa(?Transpiration Emission)
   & isa(?ResourceUtilization ResourceUtilization)
```

Figure 5.16: Rule preserving teleological structure

2. The *qualifications* comprise antecedent propositions that do not reference structural predicates.

In the example, the endowments are:

$$\{epidermis(?Beneficiary\ ?Epidermis)\\ coveringPart(?Epidermis\ ?Cuticle)\\ composedOf(?Cuticle\ Cutin)\}$$

The qualifications are:

$$\{performs(?Beneficiary\ ?ResourceUtilization)\\ performs(?Beneficiary\ ?Transpiration)\\ conduitIn(?Epidermis\ ?Transpiration)\\ transportee(?Transpiration\ ?WaterVapor)\\ transportee(?Transpiration\ ?OtherResource)\\ input(?ResourceUtilization\ ?OtherResource)\\ input(?ResourceUtilization\ ?EssentialResource)\\ isa(?WaterVapor\ Gas)\\ isa(?EssentialResource\ EssentialPlantResource)\\ isa(?Transpiration\ Emission)\\ isa(?ResourceUtilization\ ResourceUtilization)\}$$

The endowments identify structural aspects of an object that, under the right circumstances, facilitate achieving a domain goal. The qualifications identify

precisely what those circumstances are; they determine when the structural properties would benefit an object. Typically, qualifications identify behaviors of a beneficiary (e.g., performing transpiration) that create opportunities for its components to facilitate a goal.

KI identifies as *candidate beneficiaries* those objects that satisfy the qualifications. Each such candidate would benefit from having the structure indicated by the endowment. A three-step process is used to determine the candidate beneficiaries:

1. Argument-typing constraints are identified for variables (other than that denoting the beneficiary) appearing as terms in the qualifications. Since argument typing constraints are often collection specific (Section 4.2.3), the typing constraints applicable to variables in a set of propositions can interact, and identifying a typing constraint on one variable can impose additional constraints on other variables. Consequently, an exhaustive search for all applicable typing constraints could require a bounded, but potentially intractable, number of passes through the qualifications. Therefore, KI uses a greedy algorithm requiring one pass for each unique variable. The applicable typing constraints found for the example are:

   $\{isa(?Transpiration\ Transpiration)$
   $isa(?Epidermis\ TangibleObject)$
   $isa(?OtherResource\ WaterVapor)$
   $isa(?ResourceUtilization\ BotanicalResourceUtilization)\}$

   The collection denoting the most specific typing constraint applicable to each variable is substituted into each proposition referencing that variable (i.e., created figurative propositions), and the variable-typing propositions (i.e., those referencing the predicate *isa*) are removed from the

qualifications. The resulting qualifications comprise figurative propositions that reflect applicable class-membership constraints; they are:

$\{performs(?Beneficiary\ BotanicalResourceUtilization)$
$performs(?Beneficiary\ Transpiration)$
$conduitIn(TangibleObject\ Transpiration)$
$transportee(Transpiration\ Gas)$
$transportee(Transpiration\ WaterVapor)$
$input(BotanicalResourceUtilization\ WaterVapor)$
$input(BotanicalResourceUtilization\ EssentialPlantResource)\}$

2. The knowledge base is queried to determine which concepts are candidates to both share the domain goal and satisfy those properties in the constrained qualifications that reference the beneficiary (e.g., living things that perform transpiration). In this example, this query takes the form:

   $\{x \mid ako(x\ LivingObject)\ \&\ relationType(x\ performs\ y)\ \&\ ako(y\ Transpiration)\}$

   This query is automatically constructed from the figurative propositions in the constrained qualifications that reference the beneficiary; it evaluates to: $\{Stem\ Fruit\ Flower\ Leaf\}$. In other words, stems, fruit, and flowers would all benefit, as do leaves, from having structural properties identified by the endowment.

3. Argument-typing constraints are identified for the variable denoting the beneficiary in the endowment, and the specification of each candidate beneficiary is refined as necessary to satisfy these argument-typing constraints. This involves determining, for each candidate, the maximum specialization of that candidate and the applicable constraints. In the example, the only constraint applied to the beneficiary by the endowment (e.g., $argumentOneType(epidermis\ MorphologicalPart)$) is already satisfied by each of the candidates, so no additional refinement is necessary.

(a) Generalize the new information for stems

(Stem (epidermis (StemEpidermis (coveringPart (?StemCuticle (composedOf (Cutin)))))))

(b) Generalize the new information for fruit

(Fruit (epidermis (?FruitEpidermis (coveringPart (?FruitCuticle (composedOf (Cutin)))))))

(c) Generalize the new information for flowers

(Flower (epidermis (FlowerEpidermis (coveringPart (?FlowerCuticle (composedOf (Cutin)))))))

The teleological explanation of why leaves have cuticles suggests that stems, fruit, and flowers also have cuticles. These conjectures are presented to the user in the input language (i.e., semantic networks encoded as nested lists). There are no constants defined in the knowledge base that denote the epidermis of fruit or the cuticle of the stem, fruit, or flower, so these are denoted by variables to be named by the user.

Figure 5.17: Abductively generalizing the new information

Finally, those candidates for whom every structural proposition (i.e., the endowment) is already known (e.g., $Leaf$) are removed from consideration, and suggestions are generated to propose that the endowment holds for each of the remaining candidates (Figure 5.17).

Thus, KI identifies a teleological explanation that motivates the new information (i.e., explains *why* leaves have cuticles). Generalizing this explanation identifies conditions under which other domain concepts can benefit by having cuticles and supports the abductive conjectures that stems, flowers, and fruit also have cuticles. Similarly, KI motivates and generalizes the solicited revision as well (Figure 5.18). Each teleologically based conjecture is proposed to the user as a suggested knowledge-base modification.

Completing teleological explanations supports the conjecture that the beneficial properties of one domain concept hold for other domain concepts. Acquiring knowledge by this method represents a novel variation on learning by analogy. In traditional approaches [KC85, Car86, Gen89], a problem instance

(a) Generalize the revision for stems

```
(Stem (epidermis (StemEpidermis (coveringPart (?StemCuticle (composedOf (Cutin))))
                                (portal (Stomata)))))
```

(b) Generalize the revision for fruit

```
(Fruit (epidermis (?FruitEpidermis (coveringPart (?FruitCuticle (composedOf (Cutin))))
                                   (portal (Stomata)))))
```

(c) Generalize the revision for flowers

```
(Flower (epidermis (FlowerEpidermis (coveringPart (?FlowerCuticle (composedOf (Cutin))))
                                    (portal (Stomata)))))
```

(d) Generalize the revision for botanical organs

```
(⇒ (BotanicalOrgan (epidermis (Epidermis (coveringPart (?Cuticle (composedOf (Cutin)))))))
   (Epidermis (portal (Stomata)))))
```

The teleological explanation of why leaves have stomata suggests that other domain entities having cuticles also have stomata. These conjectures are presented to the user in the input language (i.e., semantic networks encoded as nested lists).

Figure 5.18: Abductively generalizing the revision

– called the *target* case – is given; to establish the analogy, the learner must identify a solved problem instance – called the *base* – from a library of cases. The base must be sufficiently similar to the target so that its solution can be adapted to solve the target case. Two common subproblems of learning by analogy are thus identifying the base case and determining what properties of the base should be mapped to the target case. In exploiting teleological explanations, KI identifies a base concept [10] as being the beneficiary of a teleological explanation. To establish the analogy, KI uses the explanation to determine which properties of the base are germane to the analogy and should therefore

[10]Here a "case" is really a concept, and the implicit task is to know all that can be known about the concept.

be mapped to the target concepts (i.e., the endowment). This is similar to the approach of *purpose-directd analogy* [KC85] and accords nicely with Gentner's model of analogy based on systematicity [Gen83]: the teleological explanation provides the *higher-order structure* that identifies which properties of the base are to be transfered across the analogy to the target. KI also uses the explanation to determine the prerequisite properties of candidate target cases (i.e., the qualifications). Thus, KI uses the teleological explanation to identify the base and target concepts and to determine precisely which properties of the base should be mapped to the target concepts. Furthermore, this approach is opportunistic: rather than waiting until an unsolved problem (target case) is encountered before attempting the analogy, KI recognizes and exploits analogical learning opportunities as they arise. Despite being explanation-based, this approach is inherently abductive: KI can teleologically explain why each target concept (e.g., stems, fruit, flowers) would benefit by having the endowment (e.g., a cuticle), but these conjectures remain purely speculative until confirmed by the user.

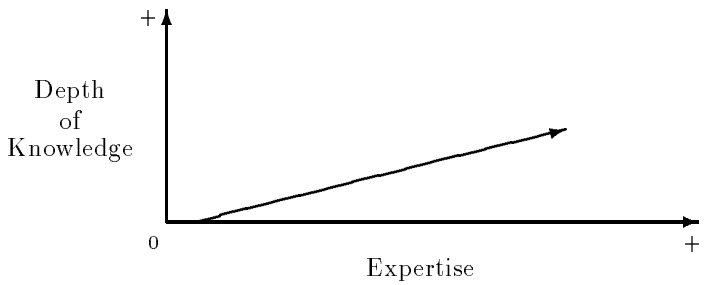
Completing teleological explanations affords the advantage of conjecturing that structural properties of one concept may be shared by other concepts (as illustrated above). It affords other advantages as well, such as identifying why a particular domain rule (e.g., one that defines the structure of a domain concept) might be true: *Leaves have cuticles <u>because</u> cuticles restrict water loss and inhibit dehydration.* Understanding why concepts have structural properties has pedagogical advantages; it enables the system, for example, to answer the query: *Why do leaves have cuticles?* Acquiring teleological reasons for the structural properties of domain concepts is a special case of a general type of learning: acquiring explanations of domain rules.

5.5 Knowledge deepening: explaining "shallow" rules

Acquiring explanations of domain rules enables a system to strengthen an imperfect theory by connecting unexplained rules to the underlying principles and tacit assumptions that justify their use. By identifying underlying principles and assumptions, explanations of rules enable the system to justify and qualify its conclusions to the user [Swa83], to guide knowledge refinement [SWMB85], and, in the case of default reasoning when assumptions are not met, to improve problem solving [SS77].

An imperfect theory is strengthened when new information enables previously unsupported rules to be explained. Initially, a novice's knowledge includes shallow, associational rules such as "birds can fly" or "leaves are green." Such beliefs are accepted by novices that do not understand, and cannot explain, why they are true. However, as knowledge is extended and expertise in the domain increases, explanations for these default rules are discovered, and they become annotated with deeper causal support. Figure 5.19 illustrates the natural transition in the depth of an agent's understanding as expertise in a domain improves. Gagne [Gag85, pages 77–79] illustrates this behavior in people with the following example:

> A student is told *In vitro experiments show that Vitamin C increases the formation of white blood cells.* The student has prior knowledge that white blood cells destroy viruses, and intuitively knows that Vitamin C is taken to fight colds, which are caused by viruses. The student realizes that Vitamin C is capable of fighting colds *because* it stimulates the creation of white blood cells, which subsequently kill cold-causing viruses.



Initially, knowledge is typically shallow: it is associational and cannot be explained. As expertise in a domain increases, the initial shallow knowledge is understood in terms of deeper knowledge such as general domain principles or causality.

Figure 5.19: The transition from shallow to deep knowledge

Thus, for an existing, "shallow" belief, the student identifies a causal explanation that was neither stated in the new information nor previously known. Discovering this explanation provides the student with greater insight into why Vitamin C is taken to fight colds. For example, the student can now explain why Vitamin C is not taken in response to similar symptoms whose causes are unrelated to viruses (e.g., allergies).

Increases in the depth of knowledge enable an agent to explain why the beliefs it holds are true; that is, it improves the *explanatory competence* of the agent. This is but one of several possible manifestations of an increase in expertise. Others include improved competence (e.g., correctly diagnosing diseases) and improved response times (e.g., quickly diagnosing diseases) while performing a task in the domain. While traditional approaches to machine learning do support improving competence and response times for performing criterial tasks they do not support the knowledge deepening effect illustrated in Figure 5.19. Similarity-based learning is restricted to producing only shallow, associational rules that cannot be explained. Traditional applications

of explanation-based learning compile shallow rules from (potentially) deeper rules, but existing learning systems cannot exhibit the knowledge deepening effect since compilation is only triggered in the absence of existing, shallow rules. In other words, EBL is performed only to speed up inference; it does not change the explanatory competence of the agent's knowledge. However, by acquiring explanations of existing domain rules, KI exhibits the knowledge deepening effect and improves the explanatory competence of the knowledge base.

The following two sections describe how KI models this learning behavior during knowledge integration. The first discusses how KI acquires explanations of structural relations by identifying their physiological functions. The second describes how KI discovers proofs of shallow rules; each such proof explains the rule in terms of other, usually more fundamental, domain rules.

5.5.1 Identifying physiological functions

Section 5.4 illustrated how teleological explanations can suggest abductive generalizations of new information; teleological explanations can also suggest why new or existing rules are true. For example, the explanation presented in Figure 5.13 reveals the physiological functions of several components of the leaf. Each essential structural property included in a teleological explanation contributes to effecting a goal; identifying physiological functions of the component referenced by a structural property involves determining precisely how that component contributes (e.g., behaviorally) to effecting the goal. This analysis is performed for each essential structural property of the explanation (e.g., each fact in 5.14b) and involves two steps:

1. The subgraphs that establish the structural property are pruned from the inference graph. This ensures that the structural property will not appear in the graph as a consequence of any properties identified as physiological functions of that component.

2. The behavioral properties (i.e., facts denoting the events that an object participates in or affects) of components that are referenced by the structural properties appearing in the remaining inference graph are identified as physiological functions of those component.

For example, the fact that the leaf epidermis has a cuticle is one of the essential structural properties of the explanation presented in Figure 5.13, and the inference graph references the behavioral fact that the leaf cuticle, the component in this structural fact, restricts transpiration. Thus, establishing a domain goal of the leaf is a consequence of the two facts that the leaf epidermis has the leaf cuticle as a covering part and that the leaf cuticle restricts transpiration. This suggests that restricting transpiration is a physiological function of the leaf cuticle. In this particular example, the behavioral fact is a consequence of the structural fact: covering the leaf epidermis *enables* the cuticle to restrict transpiration. Finally, identified functions of a component are conjectured to provide teleological, not causal, explanations for the essential structural property that references the component: *The leaf epidermis has a cuticle* <u>*because*</u> *the leaf cuticle restricts transpiration.*

Figure 5.20 presents the results of this analysis for the new concepts introduced during the example. Each relevant behavioral proposition is asserted as a physiological function of the component and memos are dispatched suggesting the user verify these assertions.

Identifying the physiological functions of domain concepts has two advantages:

1. New physiological functions extend the representation in the knowledge base of domain goals. Each behavioral property included in a teleological explanation and identified as a new physiological function defines a subgoal of the domain goal established by the explanation, and each concept "has the goal" of performing its physiological functions. Thus, identifying physiological functions extends the representation of domain goals in the knowledge base. Initially, only the generic domain goal of facilitating good health is represented; after identifying the physiological functions (e.g., those in Figure 5.20a) several concept-specific domain goals are also represented, including the goal of the leaf cuticle to restrict transpiration, and the goal of the stomata to permit the leaf's acquisition of carbon dioxide from the atmosphere. During subsequent learning, explanations that establish these new goals can be identified as teleological in addition to those that establish the facilitation of good health.

2. In many domains, function often explains structure [Sim81, DeK85, Fra93]; the physiological functions of a component help to explain the structural properties that reference that component. For example, one good answer to the question *Why do leaf epidermises have cuticles?* is *to restrict transpiration.* Before identifying this function of the leaf cuticle, the knowledge base cannot provide this (or any other) answer. Since each physiological function of a component suggests a teleological explanation for some structural property referencing the component, each identified function improves the system's ability to explain the structure of domain

(a) Acquired physiological functions of the new concepts

1. hasPhysiologicalFunction(LeafCuticle restrictsEmission LeafTranspiration)
2. hasPhysiologicalFunction(Stomata facilitates LeafCO_2Acquisition)
3. hasPhysiologicalFunction(Stomata facilitates LeafTranspiration)

(a) Acquired explanations of structure

1. justifiedBy(coveringPart(LeafEpidermis LeafCuticle)
   restrictsEmission(LeafCuticle LeafTranspiration))
2. justifiedBy(composedOf(LeafCuticle Cutin)
   restrictsEmission(LeafCuticle LeafTranspiration))
3. justifiedBy(portal(LeafEpidermis Stomata)
   facilitates(Stomata LeafCO_2Acquisition))
4. justifiedBy(portal(LeafEpidermis Stomata)
   facilitates(Stomata LeafTranspiration))

(a) The physiological functions identified for the new concepts introduced during the example. (b) Teleological explanations of structural properties enabled by the acquired physiological functions of the new concepts.

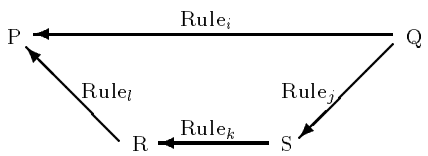
Figure 5.20: Explaining structure by function

concepts. The physiological functions are asserted as teleological explanations of the structural properties, and memos are dispatched suggesting that the user verify these assertions. Figure 5.20b presents the teleological explanations asserted for the new concepts during the example.

Identifying teleological explanations of beliefs is just one way to improve explanatory competence: acquiring proofs of rules is another.

5.5.2 Identifying proofs of rules

Proofs of a shallow domain rule identify subsets of the knowledge base comprising beliefs that collectively ensure the rule's truth. Each proof of a rule identifies a set of underlying principles and assumptions that justify the rule's



Two inference graphs that are co-rooted (i.e., they establish a common consequent): $Q \Leftarrow_i P$ $Q \Leftarrow_j S \Leftarrow_k R \Leftarrow_l P$

Nodes in the inference graphs denote sets of facts; arcs denote rules. Rule$_i$ establishes a consequent Q from a set of antecedents P. An explanation involving rules j, k, and l also establishes Q from antecedents P. If the explanation can be generalized and compiled into a rule that subsumes Rule$_i$ (i.e., its consequent subsumes the consequent of Rule$_i$ and its antecedents subsume the antecedent of Rule$_i$), then the generalized explanation provides a proof of Rule$_i$.

Figure 5.21: Acquiring a proof of a rule

use; different proofs may reveal different principles and assumptions.

Figure 5.21 illustrates a situation in which the proof of a rule can be identified: a fact has been established by two distinct inference graphs. An interesting prerequisite of this adaptation method is that multiple ground explanations for some fact must exist. When this occurs, KI determines whether any explanation can be generalized into a proof of some inference rule that establishes that fact.

Let E be the set of explanations of some fact q, and let r_i be the last rule applied (i.e., r_i concluded q) in some explanation $e_i \in E$. KI evaluates each alternative explanation in E to determine if it can be transformed into a proof of r_i. This involves three steps:

1. KI uses explanation-based generalization to compute the maximal generalization of each $e_j \in E$-e_i. Let $g(e_j)$ be the generalization of explanation e_j.

2. KI compares the consequent of $g(e_j)$ (i.e., the generalized proposition established by the proof $g(e_j)$) to the consequent (i.e., the right-hand side) of r_i. When the consequent of $g(e_j)$ is equivalent to (or subsumes) the consequent of r_i, then $g(e_j)$ is a candidate proof of r_i; otherwise, it is removed from consideration.

3. KI searches candidate proof $g(e_j)$ for a subgraph whose weakest preconditions entail the consequent of $g(e_j)$ (i.e., the subgraph and $g(e_j)$ remain co-rooted at the consequent of $g(e_j)$) and are equivalent to (or subsumes) the preconditions of r_i; each such sub-explanation constitutes a proof of r_i.

Figure 5.22 shows two explanations for the fact $isa(Leaf_1\ PhotosyntheticAgent)$ that were established by elaboration during the cuticle example. The final inference step in one explanation involves the application of the shallow, default, taxonomic rule: *Leaves are photosynthetic.* The second explanation establishes that the leaf is photosynthetic by determininng that the leaf can perform photosynthesis because of its photosynthetic ground tissue, called the leaf mesophyll. Furthermore, this second explanation has a subgraph whose weakest preconditions equal the antecedent of the shallow rule used in the final step of the first explanation. [11] Thus, the second explanation establishes a proof of the shallow rule: *Leaves are photosynthetic <u>because</u> they are capable of performing photosynthesis <u>because</u> their ground tissue, leaf mesophyll, is photosynthetic.*

Establishing a proof of the taxonomic rule $ako(Leaf\ PhotosyntheticAgent)$ improves the explanatory competence of the knowledge base; the system can

[11] Note: only this subgraph is shown in Figure 5.22.

(a) Two explanations of *isa(Leaf$_1$ PhotosyntheticAgent)*

explanation e_1
isa(Leaf$_1$ PhotosyntheticAgent)
$\Leftarrow_{49}$ isa(Leaf$_1$ Leaf)

explanation e_2
isa(Leaf$_1$ PhotosyntheticAgent)
$\Leftarrow_{51}$ canPerformType(Leaf$_1$ Photosynthesis)
$\Leftarrow_{52}$ physicalPart(Leaf$_1$ LeafMesophyll$_1$)
$\Leftarrow$ groundTissue(Leaf$_1$ LeafMesophyll$_1$)
$\Leftarrow_{53}$ isa(Leaf$_1$ Leaf)
isa(LeafMesophyll$_1$ PhotosyntheticAgent)
$\Leftarrow_{50}$ isa(LeafMesophyll$_1$ LeafMesophyll)
$\Leftarrow_{53}$ isa(Leaf$_1$ Leaf)

(b) The rules referenced in the two explanations

Rule 49 : *Leaves are photosynthetic.*
[ako(Leaf PhotosyntheticAgent)]

Rule 50 : *Leaf mesophylls are photosynthetic.*
[ako(LeafMesophyll PhotosyntheticAgent)]

Rule 51 : *Agents that can perform photosynthesis are photosynthetic.*
[$\forall$ x canPerformType(x Photosynthesis) $\Rightarrow$ isa(x PhotosyntheticAgent)]

Rule 52 : *Agents with photosynthetic parts can perform photosynthesis.*
[$\forall$ xy physicalPart(x y) & isa(y PhotosyntheticAgent) $\Rightarrow$ canPerformType(x Photosynthesis)]

Rule 53 : *Each leaf has as ground tissue a leaf mesophyll.*
[relationType(Leaf groundTissue LeafMesophyll)]

(c) The proof of *ako(Leaf PhotosyntheticAgent)*

proof(ako(Leaf PhotosyntheticAgent)
[isa(x PhotosyntheticAgent)
$\Leftarrow_{51}$ canPerformType(x Photosynthesis)
$\Leftarrow_{52}$ physicalPart(x skolem$_i$(x))
$\Leftarrow$ groundTissue(x skolem$_i$(x))
$\Leftarrow_{53}$ isa(x Leaf)
isa(skolem$_i$(x) PhotosyntheticAgent)
$\Leftarrow_{50}$ isa(skolem$_i$(x) LeafMesophyll)
$\Leftarrow_{53}$ isa(x Leaf)])

(a) Two explanations of *isa(Leaf$_1$ PhotosyntheticAgent)*; the notation $p \Leftarrow_i q$ denotes p is inferred from q by rule i. (b) Rules referenced by the two explanations in (a). (c) The discovered proof of Rule 49.

Figure 5.22: Discovering a proof of a shallow rule

now respond to the query: *Why are leaves photosynthetic?* Aside from the pedagogical advantages, acquiring this proof has inferential advantages. For example, this proof identifies tacit assumptions supporting this rule. These assumptions, if violated, can explain why a particular leaf is not photosynthetic: perhaps it has no mesophyll, or perhaps its mesophyll is not photosynthetic. Prior to acquiring the proof, the only condition that could explain why a particular leaf is not photosynthetic is the rule's precondition: perhaps it is not really a leaf.

Chapter 6

An Empirical Analysis of KI

One of the purposes of implementing a computational model as a computer program is to empirically study its behavior. This chapter presents an empirical analysis of KI's learning behavior during several learning episodes. The learning trials used for this analysis are listed in Figure 6.1. These trials fall into three categories:

1. The first three trials of Figure 6.1. are *scripted trials*. These trials were deliberately engineered to demonstrate learning behaviors that exemplify learning as knowledge integration. For each, a targeted learning behavior was identified and the knowledge base was extended and corrected as necessary to support that learning behavior.

2. The fourth through the tenth learning trials of Figure 6.1 are *representative trials*. These were developed as a coherent progression of knowledge base extensions thought to be representative for developing a botany knowledge base. For these trials, minor modifications to the knowledge base were performed in order to facilitate reasonable behaviors. This included, for example, correcting pre-existing knowledge-base errors that prevented any reasonable interpretation of the new information and launched the subsequent search for consequences in spurious directions.

1. The leaf epidermis is covered by the *leaf cuticle*, which is composed of cutin.

2. The chloroplast has as a constituent chlorophyll, which is a catalyst in the chloroplast photosynthetic light reaction event.

3. A *nonendospermic seed* is a type of seed that has no endosperm.

4. Plants have roots, stems, leaves; their ambient habitat includes both plant ambient soil and plant ambient atmosphere.

5. *Water component* is a specialization of the physical-decompositions relation; its domain is geographical regions, and its range is water. The plant microhabitat has *ground water* as its water component; ground water is a type of plant assimilable water and its constituents include both pure water and plant assimilable mineral nutrients.

6. The cactus has as a habitat the *plant desert microhabitat* which (often) has no water.

7. The *plant marine microhabitat* has sea water as its water component; it is the habitat of the *sea lettuce*, which is green, and the *nori*, which is red.

8. *Kelp* is a type of algae; its color is brown and its habitat is the *plant ocean floor microhabitat*, which has no atmosphere component and which has sea water as its water component.

9. *Phytoplankton* is a type of algae; its habitat is the *plant ocean surface microhabitat*, which has no soil component and which has sea water as its water component.

10. The habitats of seaweed include both the plant oceanic floor microhabitat and the plant oceanic surface microhabitat.

11. Chlorophyll has a *chlorophyll electron*, a type of electron. An *excited chlorophyll electron* is the state of the chlorophyll electron in which it is the *energy provider* for photophosphorylation, a subevent of photosynthetic light reactions, a subevent of photosynthesis.

12. Chloroplast photosynthesis is a kind of production that has raw materials plant assimilable CO2 and plant assimilable water, by product oxygen, and end product plant small sugar. The producer is a chloroplast, and the process occurs in a botanical organism component. The process has two steps: the light reactions followed by the dark reactions.

13. In sexual reproduction, there is a male parent actor, which is a physical organism, an a female parent actor, which is also a physical organism. The number of parents is two.

14. The eukaryotic cell is a kind of cell that has a mitochondrion, a cell nucleus, eukaryotic cytoplasm, and eukaryotic plasma membrane. The eukaryotic cell is the basic unit of eukaryotes.

15. A *fleshy fruit* is a kind of fruit whose pericarp is a *fleshy pericarp*. This fleshy pericarp contains the seeds of the fruit. The fleshy pericarp consists of the *exocarp*, the *mesocarp*, and the *endocarp*. The exocarp contains the mesocarp, and the mesocarp contains the endocarp.

16. The *pounds per square inch* of *water in roots* is typically *lower than* that of *water in soil*.

17. Plant transpiration is a transport of water from the intercellular space to the air.

Phrases in italics denote concepts not yet defined in the knowledge base.

Figure 6.1: The learning trials

3. The eleventh through the seventeenth learning trials of Figure 6.1 are *blind trials*. These were desired knowledge-base extensions submitted by knowledge engineers developing the Botany Knowledge Base. No modifications to the knowledge base were performed to facilitate these trials.

Each group of learning trials has a significantly different origin and extent to which the knowledge base was modified to facilitate desired learning behaviors. Consequently, the following empirical analyses include separate consideration for each of these three groups.

6.1 The knowledge base

The version of the Botany Knowledge Base used for this study is implemented within a version (circa 1990) of the CYC knowledge base [LG90]. Because there is no fire-wall separating the two, KI is free to make use of and modify any axiom in the entire CYC knowledge base, and the possible bindings for variables included all constants defined in CYC. Figure 6.2 presents data that quantitatively describe the contents of both knowledge bases prior to any of the learning trials.

6.2 Analyzing elaboration

6.2.1 The productivity of elaboration

One purpose of elaboration is to make explicit to the user the interaction of new and prior knowledge. To accomplish this, KI creates a context comprising hypothetical instances of the concepts deemed relevant to the new information, and then permits the non-skolemizing rules to exhaustively forward chain in this context. In this approach, the creation of hypothetical

Botany Knowledge Base	CYC Knowledge Base
constants: 4,212	constants: 27,018
collections: 2,320	collections: 8,307
predicates: 1,014	predicates: 5,528
attributes: 93	attributes: 564
facts: 255,552	facts: 948,965
rules: 16,849	rules: 39,576
ako: 3,347	ako: 10,884
akoAttribute: 8	akoAttribute: 12
inverseSlot: 928	inverseSlot: 3,369
akoSlot: 832	akoSlot: 1,429
likelyForType: 3,487	likelyForType: 3,503
inheritance: 2,784	inheritance: 8,409
miscellaneous: 658	miscellaneous: 1,379
relation type: 1,689	relation type: 2,749
argument typing: 3,116	argument typing: 12,842
authors: 35	authors: 47

These data provide a profile of the explicit contents of the knowledge bases prior to the learning trials. The category *authors* denotes the number of distinct knowledge engineers who have introduced (i.e., defined) constants to the knowledge base; each author is a distinct knowledge source (Section A.1.2). The rest of the data categories are explained in Section 3.2.

Figure 6.2: The contents of the initial knowledge base

instances can be viewed as a measure of computational expense, and the ratio of inferred facts to hypothetical instances can be viewed as a gross estimate of the productivity of elaboration. Therefore, the following data were collected during the execution of each learning trial:

1. the number of hypothetical instances created
2. the number of facts established by inference
3. the number of consequences of new information
4. the number of inference paths completed
5. the number of inference steps completed

Computed from these data were:

Trials	hypotheticals	inferred facts	(avg)	consequences	(avg)	inference steps	(avg)
1 – 3	17.7	352.7	(20.0)	105.3	(6.0)	24064.7	(22.4)
4 – 10	39.3	1165.1	(29.7)	101.3	(2.6)	4369.6	(4.4)
11 – 17	20.1	348.4	(17.3)	33.6	(1.7)	1885.4	(5.0)
1 – 17	27.6	685.5	(24.8)	74.1	(2.7)	6822.3	(9.0)

Each row summarizes the productivity of elaboration for a group of learning trials. Column 1 indicates the group of trials considered. Column 2 presents the average number of hypothetical instances introduced to the learning context during each trial. Column 3 presents first the average number of facts established by inference per trial and second the average number of inferred facts per hypothetical instance. Column 4 presents first the average number of consequences of new information established per trial and second the average number of consequences established per hypothetical instance. Column 5 presents first the number of inference steps completed during each trial and second the number of inference steps completed for each inference path.

Figure 6.3: the productivity of elaboration

1. the average number of facts inferred per hypothetical

2. the average number of consequences inferred per hypothetical

3. the average number of inference steps per inference path

This analysis is presented in Figure 6.3. While there is no base line (e.g., similar data from an alternative approach) with which to compare these data, they do indicate that in each group of learning trials elaboration in KI makes explicit a substantial number of implicit beliefs about the hypothetical instances. Of these inferred facts, a small but still significant percentage (overall, about 11%) are consequences of the new information. Furthermore, the inference paths that established these inferred facts required substantial numbers of inference steps (overall, about 9 inference steps per inference path). Therefore, many of these inferred facts are the results of fairly deep reasoning.

The number of inference steps completed for each inference path during the first group of trials is significantly higher than for the subsequent trials.

Because the first group includes demonstrations of specific learning behaviors (e.g., that of Figure 1.1), many domain rules required as prior knowledge by the targeted learning behavior were added to the knowledge base. This supplemental knowledge enabled the completion of many more inferences than would otherwise have been possible. The same knowledge engineering was not provided in support of trials 4 through 17. These later trials were executed in the relatively barren inferential environment that was the natural state of the knowledge base at that time. However, the much richer inferential environment provided for the first group of trials is likely to be more representative of what the knowledge base will converge towards during its development than is the state of the knowledge available to the other trials.

6.2.2 The explanation level

Section 3.4.4 discusses two separate subsystems for recording inferential dependencies: the explanation level and the TMS level. The explanation level abstracts the TMS level and includes only those distinctions deemed significant for learning opportunities. The purpose of this abstraction is to reduce the number of inference paths that KI must search for learning opportunities during adaptation.

To evaluate this abstraction the following data were collected during the execution of each learning trial:

1. the number of facts established through inference represented at the explanation level
2. the number of inference paths at the explanation level
3. the number of inference paths at the TMS level

		Explanation Level		TMS Level		
Trials	Facts	Inference Paths	(avg)	Inference Paths	(avg)	Ratio
1 – 3	240.0	1075.0	(4.5)	1114456700000	(4643569700)	1036703940
4 – 10	710.6	1004.9	(1.4)	17396070000	(24481804)	17311984
11 – 17	226.7	380.1	(1.7)	434893800	(1918246)	1144027
1 – 17	428.3	760.0	(1.8)	204010995712	(476661205)	268435521

Each row summarizes the explanation level and the TMS level constructed for a group of learning trials. Column 1 indicates the group of trials considered. Column 2 presents the average number of facts established by an inference represented at the explanation level per trial. Column 3 presents first the average number of inference paths established at the explanation level per learning trial and second the average number of these paths established for each fact. Column 4 presents the same data (as in Column 3) for the TMS level. Column 5 presents the ratio paths established at the TMS level to the paths established at the explanation level.

Figure 6.4: Compression achieved by the explanation level

Computed from these data were:

1. the average number of explanation-level inference paths per fact
2. the average number of TMS-level inference paths per fact
3. the ratio of inference paths at the TMS level to those at the explanation level

This analysis, presented in Figure 6.4, provides evidence for both the necessity and the utility of the abstraction provided by the explanation level. The data support the conclusion that the volume of inference paths created at the TMS level precludes their exhaustive and individual consideration. The data further show that the abstraction achieved by the explanation level during these learning trials reduces by many orders of magnitude the number inference paths explicitly represented.

	All Concepts			New Concepts		
Trials	hypotheticals	view types	(avg)	hypotheticals	view types	(avg)
1 – 3	17.7	37.7	(2.1)	1.0	3.0	(3.0)
4 – 10	39.3	94.7	(2.4)	1.3	4.3	(3.3)
11 – 17	20.1	39.3	(2.0)	1.6	1.7	(1.1)
1 – 17	27.6	17.9	(2.2)	1.4	3.0	(2.2)

Each row summarizes the applicability of view types for a group of learning trials. Column 1 indicates the group of trials considered. Column 2 presents the average number of hypothetical instances introduced to the learning context during each trial. Column 3 presents first the number of view types applicable to all hypothetical instances and second the average number of view types per hypothetical instance. Column 4 presents the average number of hypothetical instances of new concepts considered during each trial. Column 5 presents first the number of view types applicable to instances of new concepts and second the average number of these view types per hypothetical instance.

Figure 6.5: the coverage of view types

6.3 Analyzing recognition

6.3.1 The coverage provided by view types

In order to usefully guide elaboration, the view mechanism requires the availability of view types that are applicable to the concepts (e.g., the hypothetical instances) contained in the learning context. To assess the coverage of the existing view types, the number of view types applicable to hypothetical instances of existing concepts contained in the learning context and the number view types applicable to hypothetical instances of new concepts were counted during the execution of the learning trials (see Figure 6.5). The data indicate that during these examples there were on average 2.2 views types applicable to each hypothetical instance, regardless of whether the hypothetical instantiated a new or existing concept. Thus, during these examples, both existing and new concepts were fairly well covered by the view types.

6.3.2 Sensitivity of view selection

The view selection method should facilitate reasoning about concepts along a variety perspectives within different contexts. Each particular view type should be used in those situations that it (i.e., the pattern of relations it identifies) is most relevant. Therefore, a variety of learning scenarios should define a variety of learning contexts which, in turn, should cause the selection of a variety of view types. Specifically, if the view selection mechanism is sensitive to the learning context, it should be unlikely that a variety of learning scenarios would be guided by a single, or a very small set, of view types.

To test this expectation the number of unique view types used to define the views selected during the learning trials was counted and is presented in Figure 6.6. Overall, eleven different view types were used to create the thirty views selected during these learning trials. The data indicate that while some view types were deemed to be very relevant during several of the learning trials, a variety of view types were indeed selected to process these trials. Thus, a diversity of view types were used during the learning trials, and no single view type dominated selection of the background knowledge.

6.3.3 Utility for selecting relevant background knowledge

KI's method of recognition uses views to guide elaboration by selecting background knowledge that is deemed relevant to the new information. This method can be evaluated by comparing its effectiveness to that of another recognition method, such as spreading activation. One approach to measuring the effectiveness of a recognition method is to objectively appraise its productivity, computed as the ratios of the inferred facts and of the inferred consequences of new information to the number of hypothetical instances created

Trials	views	(avg)	view types	(avg)
1 – 3	6	(2.0)	5	(0.8)
4 – 10	14	(2.0)	5	(0.4)
11 – 17	10	(1.4)	7	(0.7)
1 – 17	30	(1.8)	11	(0.4)

Each row summarizes the number of view types selected for a group of learning trials. Column 1 indicates the group of trials considered. Column 2 presents first the total number of views selected during the trials and second the average number per trial. Column 3 presents first the total the number of unique view types used to define the selected views during the trials and second the average number of unique view types per selected view.
Note: during learning trials 13 and 14 a paucity of background knowledge prevented the creation of any view from the view types applicable to concepts referenced by the new information.

Figure 6.6: the diversity of selected view types

(e.g., Figure 6.3). A second approach to is to subjectively score the relevance of facts established during elaboration to the new information. Experiments were conducted to evaluate KI's view mechanism as a method for recognition by using both these approaches to compare its effectiveness with that of spreading activation.

Experimental design: A version of KI was programmed to exhibit a spreading-activation behavior during comprehension. The essential features that distinguish this control mechanism from KI's standard view mechanism is that for spreading activation:

- each view includes all and only access paths of length one emanating from the view's root concept
- every applicable (unactivated) view is selected for activation each cycle

In other words, using spreading activation, recognition selects every proposition that shares a constant with any fact already in the learning context. As before,

the selected propositions are operationalized (as ground facts that reference hypothetical individuals) and added to the learning context, and non-skolemizing rules are then allowed to forward chain exhaustively.

The two versions of KI were both applied to the first learning trial. Both were allowed to run until they exhibited the targeted learning behavior (e.g., that the cuticle facilitates the leaf's good health by restricting water loss during transpiration but also endangers the leaf's health by inhibiting photosynthesis).

Objective evaluation: Figure 6.7 identifies for each elaboration cycle during the first learning trial the number of hypothetical instances created, the number of facts inferred, the number of consequences of new information inferred, and the average interestingness score for all facts established in the learning context. These measurements are provided for elaboration controlled both by the view mechanism (Figure 6.7a) and by spreading activation (Figure 6.7b). Note that using spreading activation, the numbers of hypothetical instances and facts added to the learning context each cycle grows explosively while the percentage of established facts that are consequences of the new information and the ratio of these to the hypothetical instances drops for most cycles. However, using views, both the percentage of facts that are consequences of the new information and the ratio of consequences to the hypothetical instances remains relatively high across cycles. Thus, in this one trial, the view mechanism more efficiently reveals consequences of the new information. Furthermore, the estimated interestingness of facts established using spreading activation is significantly lower than the interestingness of facts established using views.

(a) Elaboration Results Using Views

Cycle	hypotheticals	inferred facts	(avg)	consequences	(avg)	interest score
1	2	24	(12.0)	24	(12.0)	0.82
2	11	109	(9.9)	77	(7.0)	0.67
3	5	54	(10.8)	36	(7.2)	0.67
Total	18	187	(10.4)	137	(7.6)	0.69

(b) Elaboration Results Using Spreading Activation

Cycle	hypotheticals	inferred facts	(avg)	consequences	(avg)	interest score
1	2	28	(14.0)	27	(13.5)	0.82
2	10	84	(8.4)	18	(1.8)	0.41
3	25	396	(15.8)	105	(4.2)	0.44
4	66	1022	(15.5)	104	(1.6)	0.39
Total	103	1530	(14.9)	254	(2.5)	0.41

The number of hypotheticals instances, inferred facts (and the average per hypothetical), consequences of the new information (and the average per hypothetical), and interestingness scores established by cycle during the execution of the first learning trial.

Figure 6.7: Objective comparison of views vs. spreading activation

Subjective evaluation: An independent evaluator was recruited to subjectively estimate the relevance of the facts established during elaboration by the two versions of KI. Because elaboration during spreading activation established so many facts, the evaluator was not able to consider each fact individually, and a selection method had to be devised to determine which facts would be evaluated.

The evaluator was presented a list, sorted alphabetically, of all the constants denoting the hypothetical individuals that were created during the first learning trial performed with the spreading-activation control mechanism. The evaluator subjectively assigned to each constant a score indicating its relevance to the new information. This relevance score was an integer ranging from 1 (denoting very low relevance) to 5 (denoting very high relevance).

(a) Relevance Scores Using Views

	Relevance Score						
cycle	1 (very low)	2 (low)	3 (medium)	4 (high)	5 (very high)	**Total**	(avg)
1	0	3	4	1	6	14	(3.7)
2	0	5	5	16	47	73	(4.4)
3	1	5	10	14	22	52	(4.0)
Total	1	13	19	31	75	139	(4.2)

(b) Relevance Scores Using Spreading Activation

	Relevance Score						
Cycle	1 (very low)	2 (low)	3 (medium)	4 (high)	5 (very high)	**Total**	(avg)
1	0	3	4	1	6	14	(3.7)
2	1	12	14	20	24	71	(3.8)
3	2	6	12	22	54	96	(4.2)
4	76	46	48	34	14	218	(2.4)
Total	79	67	78	77	98	399	(3.1)

Subjective estimates of the relevance of facts established during elaboration controlled by the view mechanism (a) and by spreading activation (b). Columns 2 – 6 present the number of facts having each relevance score. Column 7 presents first the total number of facts scored for each cycle and second the average score for these facts.

Figure 6.8: Subjective appraisal of views vs. spreading activation

Next, the evaluator was presented with a list of all facts established during elaboration that referenced either any of the ten constants having the highest relevance scores or any of the ten constants having the lowest relevance scores. The evaluator assigned to each fact a score indicating its relevance to the new information.

For each control mechanism, the facts scored by the evaluator were tabulated by both relevance score and by the cycle in which the fact was established. The results are presented in Figure 6.8.

There were 399 facts established using the spreading activation control mechanism that referenced any of the twenty constants having the highest and lowest relevance ratings; 139 of these facts were also established using

the view control mechanism. The cumulative average relevance score for facts established using view control is 35% higher than the average score for facts established using spreading activation. This difference is statistically significant at a .99 level of confidence. Furthermore, it is interesting to note that while the average relevance score remained fairly constant across the different cycles using view control, using spreading activation it dropped dramatically for the last cycle. It is likely (due to the explosive number of hypotheticals and facts added each cycle) that the average relevance of facts established using spreading activation would continue to decrease asymptotically if additional cycles were executed.

6.4 Analyzing Adaptation

6.4.1 Diversity of learning behaviors

KI was designed to exploit a method of searching for the consequences of new information that was not dedicated to a single adaptation method. The methods for elaboration and recognition reveal the consequences of new and relevant prior knowledge; a suit of adaptation methods then searches these consequences for learning opportunities. This approach separates the search for the consequences of new and prior knowledge from the detection and exploration of learning opportunities. This separation affords a single, uniform method for identifying consequences that can be used seamlessly and concurrently with a variety of adaptation methods and thus support a variety of learning behaviors.

To provide evidence for this, the frequencies for each type of learning opportunity that was detected and exploited during the examples are summarized in Figure 6.9. The data indicate that the learning opportunities were both substantial and diverse: a variety of learning behaviors were exhibited during

Trials	Acquired Rules							Suggestions		
	tax	inh	rel	arg	teleo	abd	total	gaps	expls	conflicts
1 – 3	2.7	9.7	53.0	4.0	6.0	1.0	72.0	13.7	8.0	99.7
4 – 10	2.1	20.0	105.1	14.9	1.3	0.7	143.0	8.4	16.1	554.3
11 – 17	3.4	10.6	71.4	11.9	4.9	0.0	100.6	6.9	9.6	62.6
1 – 17	2.8	14.3	82.1	11.7	3.6	0.5	113.0	8.7	12.0	272.6

The average quantities of acquired rules per learning trial by type. Presented are the average numbers of acquired taxonomic rules (column 2), inheritance rules (column 3), relation-type rules (column 4), argument-typing constraints (column 5), rules resulting from teleological learning (column 6), rules resulting from other abductive reasoning (column 7), and all acquired rules (column 8). Also presented are the average number of suggestions to fill knowledge-base gaps (column 9), to explain new or existing beliefs (column 10), and resolve conflicts (column 11).

Figure 6.9: Scope of learning opportunities

the learning trials as demonstrated by the diversity of the types of knowledge acquired.

Note that there are many more conflicts detected during the second set of learning trials. This is primarily due to these trials having many more facts generated during elaboration (see Figure 6.3). It also may be partially due to the extent of interaction among the learning trials of this set. These trials comprise a sequence of interacting learning episodes (i.e., the training of one episode is relevant to the training of the next in the sequence). Because so much interaction occurs, the conflicts revealed by an earlier trial may have also been encountered by a later trial. There was little interaction among the learning trials in the other two sets.

6.4.2 Utility of conflict-resolution hierarchies

KI sorts conflicts into conflict-resolution hierarchies as they are encountered during elaboration. Separate hierarchies are maintained for the two types of conflicts, anomalies and errors (e.g., constraint violations), since their

repair strategies tend to be so different. Each hierarchy identifies subsumption relations among the alternative knowledge-base revisions. One revision subsumes another when it resolves every conflict that is resolved by the other. The hierarchies prioritize for the user the alternative knowledge-base revisions that resolve many conflicts over those alternative revisions that resolve few conflicts. The utility of the hierarchies can be measured by the ratio of all knowledge-base revisions that are subsumed in the hierarchies by the most powerful alternative revisions.

Figure 6.10 summarizes the potential gains achieved by sorting suggested conflict resolutions into hierarchies. The data indicate that both hierarchies have the potential to significantly reduce the number of candidate fixes that must be considered (e.g., by the user). Overall, only 20.3% of all candidate revisions sorted into the error-resolution hierarchies are not subsumed by other alternative revisions. The hierarchies automatically identify those most powerful 20.3% of the revisions. Similarly, the hierarchies automatically identify the 21.6% of the knowledge-base revisions identified for resolving anomalies that are not subsumed by alternative revisions. Thus, by sorting the alternative candidate knowledge-base revisions, the conflict-resolution hierarchies provided substantial assistance during these learning trials in prioritizing the relatively few, most powerful revisions over the relatively many, subsumed revisions.

6.4.3 Measuring learning gain

The obligation of every non-trivial learning system is to acquire knowledge beyond the literal content of new information (Section A.3). Learning gain is defined as the amount of acquired knowledge (measured in terms of the number of beliefs asserted or retracted) not included explicitly in the new

Trials	Errors				Anomalies			
	conflicts	all fixes	common fixes	%	conflicts	all fixes	common fixes	%
1 – 3	11.3	70.0	26.7	31.0	5.3	29.7	2.3	7.9
4 – 10	93.0	522.6	86.7	16.6	9.1	34.3	11.3	32.9
11 – 17	11.6	62.6	28.9	46.1	0.0	0.0	0.0	
1 – 17	45.1	253.3	51.4	20.3	4.7	19.4	5.1	26.1

Presented are, for both errors and anomalies, the average number of conflicts encountered, the average number of possible knowledge-base modifications identified to resolve the conflicts, and the average number of modifications that are identified by roots of subtrees in the conflict resolution hierarchies.

Figure 6.10: The utility of the conflict-resolution hierarchies

information; it provides a natural measure to estimate the effectiveness of a learning program.

When new information is not expressed in the representation language, it is not always clear precisely what the explicit contents of that information might be. The new information must be interpreted, and the interpretation methods often introduce learning gain by augmenting the literal translations of new information. However, even when the new information must be interpreted, learning gain can be used to compare two different agents, each executing a common set of knowledge-base modification tasks. The *relative learning gain* is defined as the amount of knowledge acquired by one agent (e.g., a learning program) beyond that acquired by another (e.g., a knowledge engineer).

To determine the relative learning gain of KI, professional knowledge engineers were recruited to perform the set of learning trials listed in Figure 6.1. These knowledge engineers were quite familiar with the representation language but only marginally familiar with botany and the contents of the knowledge base. However, most of these trials involve only a basic and common knowledge of botany.

Trials	KE	KI	gain
1 – 3	5.0	81.3	76.3
4 – 10	10.1	176.4	166.3
11 – 17	17.4	141.3	123.9
1 – 17	12.2	145.2	132.9

The relative learning gain (column 4) is computed as the difference between the number of axioms produced by KI and the number of axioms developed manually by a knowledge engineer (KE).

Figure 6.11: Relative learning gain

For each trial, a knowledge engineer was provided with the new information presented both as a semantic network and as a statement in English. The knowledge engineers were free to make any knowledge-base modifications they felt were appropriate and to inquire about either the domain or the contents of the knowledge base. They were encouraged to follow their normal practices when formalizing and entering knowledge.

The number of axioms produced manually by the knowledge engineers was then compared to the number of axioms produced automatically by KI. Figure 6.11 presents the results of this experiment. The relative knowledge gain exhibited by KI is significant. Overall, KI derives many times more axioms during these learning trials than was derived manually.

6.4.4 Measuring learning utility

While the data in Figure 6.11 indicate that KI identifies a relatively large number of learning opportunities during the learning trials, it does not indicate how useful are the new axioms that result from those opportunities. Traditionally, the utility of acquired knowledge is demonstrated by showing that after learning the system's performance has improved on a set of test queries (i.e., instances from the task domain). This approach is problematic

for evaluating the utility of acquired foundational knowledge since there is no assumed application task with which to test the system's performance. However, the relative measure of utility can be estimated by subjectively comparing the axioms produced by KI with those produced by the human knowledge engineers.

For each learning trial, the axioms produced by KI that "correspond" to the axioms produced manually by knowledge engineer were selected. Two axioms correspond if they are the same or if the predates match and most of the arguments match (e.g., (*genls GroundWater Water*) and (*genls GroundWater PlantAssimilableWater*) correspond). [1]

Next, for each learning trial, the selected KI axioms were compared to the corresponding axioms developed by the knowledge engineer, and three sets of axioms were defined. The first set includes axioms produced both by KI and the knowledge engineer (i.e., those produced by KI that differed from manually produced axiom only by variable names or by the order of literals). The second set includes axioms produced only by the knowledge engineer. The third set includes axioms produced only by KI. For each trial, the second and third sets were randomly labeled as resulting from *Method 1* and *Method 2*.

Finally, for each trial, a knowledge engineer (other than the knowledge engineer who performed the learning trial) assessed the utility of the axioms that were produced by either KI or the knowledge engineer but not both. For each *Method 1* axiom the evaluator was asked to indicate how much she agreed with the statements *This axiom is useful* and *This axiom is subsumed by axioms*

[1] The knowledge engineers did not produce axioms corresponding to the targeted learning behaviors of the first three trials. Therefore, these engineered learning behaviors were not included in this study.

Trials	KE		KI	
	all	unique	all	unique
1 – 3	3.6	2.2	4.5	3.8
4 – 10	4.3	2.2	4.9	4.6
11 – 17	4.7	4.0	4.5	3.5
1 – 17	4.5	3.2	4.7	3.8

The subjective utility scores for all axioms produced by the knowledge engineer (column 2), axioms produced only by the knowledge engineer (column 3), all axioms produced by KI (column 4), and axioms produced only by KI (column 5).

Figure 6.12: The utility of acquired axioms

of Method 2 and the prior knowledge base. For each statement, the evaluator scored each *Method 1* axiom with an integer ranging from 1 (denoting strong disagreement with the statement) to 5 denoting (denoting strong agreement with the statement). The evaluator was then asked to perform a similar evaluation of the *Method 2* axioms. The axioms that were produced by both KI and the knowledge engineer were given the scores of 5 both for utility and subsumption.

Figure 6.12 presents the average utility score for axioms produced by KI and for axioms produced by the knowledge engineer. The overall utility score for axioms produced only by KI was 0.6 (or about 19%) higher than the scores for axioms produced only by the knowledge engineer. This difference is statistically significant at .95 level of confidence.

Figure 6.13 presents the extent to which axioms produced by the human knowledge engineer were subsumed by axioms produced by KI. In almost every learning trial, both KI and the knowledge engineer produced axioms that transcend the explicit content of the new information. Learning systems that exploit significant bodies of background knowledge are inherently idiosyncratic, and it would be unreasonable to expect that any learning system (e.g., KI) to

Trials	all KE axioms	useful KE axioms
1 – 3	3.8	4.4
4 – 10	4.8	5.0
11 – 17	4.4	4.4
1 – 17	4.5	4.6

The subjective estimates of the extent to which axioms produced by a knowledge engineer were subsumed by the axioms produced by KI and the prior knowledge base. Column 2 presents the scores for all manually produced axioms. Column 3 presents the scores for those manually produced axioms deemed useful (e.g., having a utility score greater than 3).

Figure 6.13: KI's coverage of manually produced axioms

completely subsume the learning behavior of another learning system (e.g., a knowledge engineer). However, the data indicate that KI was fairly effective at producing axioms during these learning trials that subsume the useful axioms produced by human knowledge engineers. Overall, KI scored a 4.6 out of a possible 5.0 for subsuming the useful axioms produced manually by professional knowledge engineers on these learning trials. Statistical analysis determined that with a 95% confidence coefficient this score would range between 4.4 and 4.8.

Chapter 7

Related Work

This chapter reviews research related to the task of knowledge integration and the computational methods implemented in KI. The first section discusses research on formalizing belief revision which is relevant to the overall task of knowledge integration. The following three sections review research on computational methods relevant to performing elaboration, recognition, and adaptation.

7.1 Belief Revision

Belief revision comprises a community of researchers in philosophy, mathematics, and computer science that are interested in formally defining specifications (called postulates) for algorithms that implement operators to change knowledge. These postulates formalize three such operators:

1. expansion (denoted by the symbol $+$) adds beliefs to existing knowledge
2. contraction (denoted by the symbol $-$) removes beliefs from existing knowledge
3. revision (denoted by the symbol $\mp$) modifies existing knowledge to consistently include beliefs

The postulates specify what invariant properties should be true of these three operations. The most prominent goal underlying the proposed postulates is to promote *minimal change* [Har86]: *knowledge changes should be minimal in both the addition of new beliefs and the loss of prior beliefs.*

7.1.1 Belief revision and belief sets: The AGM model

Most formal approaches to belief revision are restricted to classical propositional logic and consider theories represented as *belief sets* which explicitly include their inferential closure. [1]

The Alchourron, Gardenfors and Makinson (AGM) Model defines rationality postulates for each of the three operators applied to belief sets [Gar92]. For example, there are six basic postulates for belief revision defined in terms of a belief set Δ and a statement in the representation language Υ in order to characterize $(\Delta \mp \Upsilon)$, the revision of Δ for Υ (i.e., the operation that modifies Δ to consistently include Υ). The six basic revision postulates are:

1. the revision of a belief set with respect to a new statement is a belief set
2. the new statement is included in the resulting belief set
3. the revision is a subset of an expansion with the new statement
4. when the new statement is consistent with prior knowledge, the revision is equivalent to an expansion with the new statement
5. the revision is inconsistent if and only if the new statement is inconsistent

[1] By assuming inferentially closed theories the learning goal of economy (e.g., compactness), discussed in Appendix B.2 and of considerable interest to machine learning, is not relevant to this formalization of belief revision.

6. the revisions of a belief set for equivalent statements are equivalent (i.e., revision is not sensitive to syntactic variations in the form of the input statement)

The *Levi identity* defines revision in terms of contraction and expansion:

$$\Delta \mp \Phi = (\Delta - \neg\Phi) + \Phi$$

The *Harper identity* defines contraction in terms of revision:

$$\Delta - \Phi = \Delta \cap (\Delta \mp \neg\Phi)$$

From a learning point of view, these postulates constrain the classes of permitted learning behavior when integrating Υ into Δ (i.e., they prescribe *admissibility criteria*, see Section B.1).

The application of these postulates is illustrated with the multiple-extension problem [Gin87], one of the essential issues in the study of the non-monotonic nature of knowledge (and related to the *credit-assignment problem* in machine learning [Die82]). Consider the following learning situation:

prior knowledge: (1) $a \Rightarrow b$
(2) $b \Rightarrow c$
(3) $c \Rightarrow d$
(4) a

new information: $\neg d$

Which prior belief should be retracted? Any three out of the four initial beliefs are consistent with the new information, so there are many possible extensions; however, many possible extensions are precluded by the AGM revision postulates. The initial belief state includes:

$$\{a\,(a \Rightarrow b)\,b\,(b \Rightarrow c)\,c\,(c \Rightarrow d)\,d\}$$

The second AGM postulate requires the inclusion of $\neg d$ in the revised belief set, and the fifth postulate requires the absence of d. The four possible extensions, each corresponding to retracting one of the four initial beliefs, are:

$$\{a\,(a \Rightarrow b)\,b\,(b \Rightarrow c)\,c\,\neg d\}$$

$$\{a\,(a \Rightarrow b)\,b\,\neg c\,(c \Rightarrow d)\,\neg d\}$$

$$\{a\,\neg b\,(b \Rightarrow c)\,\neg c\,(c \Rightarrow d)\,\neg d\}$$

$$\{\neg a\,(a \Rightarrow b)\,\neg b\,(b \Rightarrow c)\,\neg c\,(c \Rightarrow d)\,\neg d\}$$

Thus the six basic AGM postulates for revision mitigate, but do not completely solve, the multiple extension problem, and it's solution requires additional mechanisms (or postulates).

One response to the multiple extension problem is to maintain all consistent extensions as alternative contexts; this is the policy of the ATMS [DeK86]. While this has the advantage of permitting very efficient changes from one extension to another (e.g., when an extension becomes contradicted by subsequent information), it has the disadvantage of having to maintain, potentially, an exponential number of contexts.

A second possible response to the multiple extension problem is to maintain a single possible extension selected at random. This is the policy of many standard TMSs [Doy79]. It has the advantage of avoiding the problem of maintaining a potentially exponential number of contexts but has the disadvantage of introducing nondeterminism into the contraction and revision operators (i.e., they are not pure functions). This violates the so-called *recovery postulate* which requires that first contracting and then expanding a belief set by a belief should result in the original belief set.

Within the belief revision community, the most common response to the multiple extension problem involves prioritizing beliefs. *Epistemic entrenchment* reflects the notion that not all beliefs are of equal utility. Beliefs of greater utility should be held with greater commitment than beliefs of lesser utility. The epistemic entrenchment value of each belief denotes a given level of commitment to that belief. When a contraction or revision is necessary, beliefs held with less commitment should be discarded before beliefs with greater commitment. Fully prioritizing beliefs by their entrenchment value establishes a notion of minimal change for the belief set that resolves multiple-extension problems, such as the one above, and enables contractions and revisions to be deterministic. For example, were the assignment of entrenchment values to beliefs consistent with the following ordering:

$$a > (a \Rightarrow b) > (b \Rightarrow c) > (c \Rightarrow d)$$

then the first of the four possible extension above would be deterministically selected.

There are postulates characterizing how the entrenchment of beliefs interact [Gar92]. For example, one postulate states that epistemic entrenchment is transitive; another states that a belief is less entrenched than its consequences. The motivation for this second postulate is that in order to give up commitment to a consequence one also must give up commitment to some (at least one) of the premises from which that consequence is derived; however, one can give up commitment to the premises without rejecting their consequence. Thus, the relative entrenchment of two beliefs varies inversely with their inferential dominance over each other.

KI exploits a partial order provided by the three-valued conviction associated with beliefs in the knowledge base (i.e., preferring to refute assump-

tions before nonmonotonic beliefs, and preferring to refute nonmonotonic beliefs refuting beliefs labeled as monotonic; see Section 3.5.1). Because this ordering is only partial, when revising knowledge to resolve a conflict KI identifies an equivalence class of least entrenched prior beliefs (rather than a single least entrenched belief) and relies on other criteria (e.g., subsumption in the conflict hierarchies, the user) to select from among the equivalence class.

7.1.2 Belief revision and belief bases

Hansson [Han92] proposes postulates that characterize knowledge revision on *belief bases*, which are not assumed to be inferentially closed, rather than on belief sets. This has obvious computational advantages, since computers cannot explicitly store the infinite inferential closure of significant knowledge bases. It also has advantages in providing an intuitively appealing, less conservative, interpretation of minimal change.

Because they are finite, belief bases link inferred beliefs to their justifications; this differs significantly from belief sets. Consequently, in a belief set, when all the justifications of a belief in a are rejected, the belief itself should also be rejected but is preserved [Gar88]. However, in belief bases, implicit beliefs are closely tied to the beliefs used in their derivations; consequently, rejecting beliefs required for the derivation of an implicit belief also causes that implicit belief to be rejected as well. For example, given explicit beliefs a and $a \Rightarrow b$, the closed belief set would include b, but the open belief base would not. The revision of the belief set for $\neg a$ would include b, while the similar revision of the belief base would not, and b would no longer be in the inferential closer of the revised belief base. Thus, the notion of minimal change is extremely conservative when applied to revising belief sets, but can be naturally relaxed

in ways that accord with our intuitions when applied to revising belief bases [Han92].

Formalizing belief revision for belief bases also permits *reorganization* a type of knowledge change that has significant pragmatic importance but is not meaningful for belief sets. A reorganization is a change to a belief base that does not change its inferential closure. This can have economic benefits (i.e., by reducing the memory required to store the knowledge or by reducing the response time through facilitating faster derivations of some beliefs). But it also can have consequences for subsequent revisions [Han92]. For example, reorganizing a knowledge base containing a and $a \Rightarrow b$ to also explicitly include b removes the dependence of b on the prior two beliefs. Revising this reorganized knowledge base for $\neg a$ now includes b, while as noted above, this revision on the unreorganized knowledge base does not include b.

Formalizing belief revision for belief bases permits approximate implementations of the AGM model for the three belief change operators for first-order theories [DW93, Wob94]. Belief bases can be stored in finite space, but their potentially infinite inferential closure precludes ensuring consistency during belief change operators. Consequently, such implementations must sacrifice a guarantee of either termination or consistency.

7.1.3 Belief revision and coherence

Most approaches to formalizing belief revision do not permit the option of rejecting or revising the new information rather than conflicting prior knowledge (e.g., the Levi identity). However, some formal approaches to belief revision recognize that this is an essential operation for evolving knowledge, especially in the context of modeling communication: the recipient of an utter-

ance may choose to disbelieve the utterance or to believe some revision of the utterance [Gal92].

In addition to the minimal change policy, Galliers adopts the *maximal coherence* policy [Har86]: *knowledge revision (e.g., the gain of new beliefs and the loss of prior beliefs) is permitted only when it sufficiently increases the coherence of the resulting knowledge.* Coherence requires consistency and is established by support (e.g., by justifications established through inference, so that a belief and each of its consequences are deemed mutually coherent). [2]

While from one point of view the concerns of belief revision seem quite germane to machine learning, from another point of view they are quite divergent. For example, both learning and belief revision are concerned with how to revise knowledge when new information is encountered. However, the tenet of minimal change (e.g., the fourth AGM postulate for revision) is incompatible with the premise that non-trivial learning acquires more than what is explicitly included in the new information. The very idea of making an inductive leap directly violates the essence of minimal change. However, the tenet of maximizing coherence begins to bridge the gap between learning and belief revision by permitting non-minimal change when doing so increases overall coherence (e.g., the ability to both compactly represent a concept and correctly derive its extension).

[2]Note that the symmetry of this interpretation of coherence accords with the inference-based interpretation of relevance implemented in KI.

7.2 Elaboration

There is a paucity of research into computational methods for detecting the consequences of new information for prior knowledge that do not commit to narrow assumptions about the eventual uses of the knowledge. This sections discusses two approaches. KI differs from each of these methods by one or more of the following:

1. KI permits a very expressive representation language
2. KI permits a very large knowledge base
3. KI identifies deep consequences of new information without sacrificing a guarantee of termination

7.2.1 FIE: Integrating propositions into a theorem prover

In FIE, Cohen studied how existing knowledge could be used to critique proposed extensions to a deductive knowledge base [Coh84]. FIE computes shallow consequences of new clauses and reports those it estimates as interesting to the user. Results that contradict the user's expectations suggest bugs. Results that confirm the user's expectations support the correctness of the extended system. Thus, FIE critiques proposed extensions for the user, as does KI.

FIE shares with KI the basic approach of using forward-chaining inference to identify the consequences of new information. However, some important differences exist. FIE is limited to propositional theories, while KI uses hypothetical reasoning to handle a first-order theory. Furthermore, FIE addresses the problem of controlling the forward-chaining inference with a

domain-independent assessment of "interestingness." Interestingness is measured in terms of the domain of theorem proving (e.g., number of literals referenced by a clause) rather than in terms of the domain being modeled. Inference paths for which the conclusions are deemed uninteresting are terminated. In contrast, KI attempts to restrict reasoning to background knowledge that is deemed to be relevant to the new information (e.g., using views).

FIE computes the shallow entailment of the training. The consequences of each new clause is determined by repeatedly using a modified form of resolution to resolve the new clause with each existing clause until no resolvants are interesting. No attempt is made to identify and isolate a subset of existing beliefs determined to be uniquely relevant to the new information. Consequently, deep consequences are not computed and the size of the extension and initial domain theory must be small.

7.2.2 Inference in an implementation of AGM belief revision

Dixon and Wobcke have achieved an approximate implementation of AGM postulates for belief revision adapted for finite belief bases [DW93, Wob94]. Their implementation attempts to identify and remove all redundancy and inconsistency during changes in belief. Since the system operates on a first-order theory, complete elaboration of new information is not possible. This system, at the user's discretion, exploits a bound on the number of times a formula can be used in a derivation (sacrificing completeness), or inference is simply allowed proceed without any bounds (sacrificing a guarantee of termination). Therefore, despite being implemented and applied to small examples [DW95], this approach is not feasible for large theories.

7.3 Recognition

There are two veins of research that bear on the use of views to focus attention. The first vein involves a community of researchers that study explanation generation and natural language processing. In this research, views correspond to segments of the knowledge base which are used to guide comprehension and to restrict the contents of generated explanations. The second vein involves the qualitative reasoning community. In this research, views correspond to fragments of qualitative models which are composed to complete a model that is sufficient to perform a specified simulation task.

KI differs from each of the methods discussed by one or more of the following:

1. KI guides inference without requiring a specific query or assuming a criterial task

2. KI interleaves performing inference and selecting a view to guide subsequent inference

3. a relatively small set of manually defined view types can be used to define a vast number of views

4. KI automatically generates views on demand

5. KI heuristically selects one view to use each processing cycle from among the many alternative candidate views

6. multiple views can be defined for each concept

7. multiple views can be composed during inference

7.3.1 Focus spaces

One of the first uses of view-like structures for natural language processing was a method developed by Grosz for structuring background knowledge with *focus spaces*. Focus spaces guide dialog comprehension by restricting the comprehender's attention to relevant portions of background knowledge [Gro86]. This approach assumes that background knowledge is encoded as a semantic network; it uses a semantic-network partition to define each focus space. Hendrix developed network partitions called spaces to represent abstractions, hypothetical situations, and the scoping of quantified variables [Hen75]. Sets of spaces could be organized into *vistas* which were used to restrict accessible contents of the network (e.g., while completing inferences). Focus spaces extend the use of network partitions to permit multiple, overlapping partitions being imposed on the same network.

Information explicitly referenced in the dialog is considered "in focus" (i.e., relevant) and made accessible for inference. Focus spaces are used to determine which additional knowledge, that which is not explicitly mentioned, is also relevant to comprehending the current part of the dialog and should also be considered in focus (and accessible during inference).

Focus spaces do not completely restrict which fragments of background knowledge can be accessed; rather, they order the accessibility of background knowledge to facilitate considering those concepts and relations that are within the current focus space prior to considering concepts that are outside of the focus space. Thus, they heuristically guide attempts to find bindings to variables when disambiguating a dialog utterance (e.g., a definite noun). For example, while comprehending the dialog:

The key labeled A opens the office door.

The key labeled *B* opens the car door.

Enter the office.

The door will be locked so use the key to open it.

the referent "the key" in the last utterance is ambiguous. The strategy of recency would incorrectly bind "the key" to the key labeled *B*. However, focus spaces overcome this failing. The prior sentence triggers a focus space that includes the key labeled *A* and excludes the key labeled *B*. So while disambiguating the ambiguous reference in the last sentence with focus spaces, "the key" correctly binds to the key labeled *A*. Thus, focus spaces are useful for resolving ambiguous references in definite noun phrases by improving on the simple strategy of searching for the most recent match in the dialog.

While the overall motivation of guiding comprehension is shared by both focus spaces and KI's view mechanism, the methods that define their use are significantly different. One important difference is that all focus spaces are assumed to be predefined explicitly and permanently. An important feature of KI's view mechanism is that (a potentially infinite number of) views are constructed automatically from a relatively small set of predefined view types. Furthermore, views are created dynamically on demand. This promotes views automatically reflecting the system's current beliefs despite the evolution of knowledge and the introduction of new concepts.

A second important difference involves the issue of changing focus. KI attempts to perform deep reasoning about each presented fragment of new information. Since the emphasis is on comprehending a single fragment of new information rather than comprehending many utterances during a discourse, the problem of automatically changing focus during a learning scenario has not been significantly addressed. Switching focus involves changing the set of

concepts and relations that are given prioritized access during inference. KI has no mechanism to do this other than to initialize a new learning context by starting a new learning scenario and identifying a completely new set of relevant fragments from the background knowledge. KI relies on the user/teacher to indicate when to start a new learning context (e.g., by beginning a new learning event). In contrast, Grosz has developed methods to automatically change focus by recognizing significant changes in the explicit references made in the dialog. Her approach assumes the dialog concerns questions and advice about how to perform a set of pre-enumerated problem-solving tasks and uses a hierarchical model of the assumed problem-solving tasks in order to decide when the focus must be changed. For example, the focus is changed whenever the dialog references a new task that is not subordinate or superordinate to the task that is the current focus.

7.3.2 Perspective hierarchies

McKeown proposes an approach similar to focus spaces that uses goal hierarchies to include in explanations only information of interest to user [MMM85]. The system provides advice to a user who is assumed to be trying to achieve one of a given set of goals (e.g., registering for the college courses appropriate for their major). Knowledge is organized by *perspective hierarchies*; each possible goal of the user corresponds to one perspective hierarchy. Collectively the hierarchies partition the domain knowledge; each cell of the partition includes knowledge relevant to the goal corresponding to the hierarchy.

Selecting the appropriate subset of relevant domain knowledge corresponds to determining which goal the user is pursuing. There is a separate hierarchy of goals which is used to identify an overall discourse goal from the

goals associated with each utterance. The discourse goal is found by computing a minimal generalization within the goal hierarchy of the individual goals associated with each utterance. The discourse goal is then used as the user's goal for completing explanations.

When the system must answer a query, it makes available to a rule base only the facts from the perspective hierarchy indexed by the discourse goal; facts from the other perspective hierarchies are not available to the rule base. The query is answered by the rule base and the trace of the reasoning is provided as an explanation. Thus, by restricting what facts are made available to the rule base, perspectives control what rules can be fired and what explanations can be constructed.

This approach of controlling inference by first selecting a portion of relevant background knowledge is very similar to KI. However, as with focus spaces, this approach relies on predetermined explicit views, while KI automatically defines views from a relatively small set of predefined view types.

7.3.3 Romper

McCoy's Romper system [McC85, McC89b] addresses the difficult problem of determining what properties to include in a description of an object. This problem is particularly difficult because as a problem-solving context changes so to must the description of objects within that context.

KRL [BW77] and focus spaces (discussed above) proposes methods of considering a concept from different perspectives; each perspective is associated with a different context and identifies different properties of the concept that are relevant in the associated context. Romper extends the representation of a perspective to permit a perspective to apply to more than a single con-

cept. Thus, Romper distinguishes perspectives defined for domains from the perspectives defined for individual concepts in KRL and focus spaces. A domain perspective can be applied to any concept in the domain to identify which properties are relevant to describing that concept in the context associated with that perspective.

Domain perspectives are declaratively represented; each specifies a level of salience for every relation in the domain. For example, Nixon can be represented in either a political context or a religious context. In the former (*politicalParty Nixon GOP*) has very high salience; in the later (*religion Nixon Quaker*) has very high salience. Thus, in different contexts, estimations of similarity can use context-specific measures of salience. In a political context Nixon can be judged similar to other republicans regardless of their religion; in a religious context Nixon can be judged similar to other Quakers regardless of their political affiliation.

The distinction between domain and concept perspectives established by Romper is significant. It separates declarative knowledge about what is true in a domain from declarative knowledge about what contexts are useful within the domain. Furthermore, it suggests that a relatively few domain perspectives can be specified and applied to a relatively large number of domain concepts; each such application defining a context-specific description of the concept. Thus, the distinction between domain and concept perspectives in Romper provides the groundwork for the distinction between view types and views in KI. The significant differences between Romper and KI are:

1. Romper identifies the relations relevant to a domain perspective by attributing salience values to all relations in the domain. High salience values denote high relevance to the perspective. However, the salience of a

relation (independent of its arguments) need not be constant throughout the perspective. For example, (*capacityOf Auditorium* 1300) and (*capacityOf Car* 4) are not equally relevant when planning a conference or planning a trip. Similarly, (*isa Nixon Republican*) and (*isa Nixon Quaker*) share the common relation isa and so must share a common salience level. The node constraints in KI's view types permit discriminating among propositions rather than only among relations. Furthermore, properties used to estimate the relevance of a view, such as coreference and interestingness, are established for the facts contained within the view rather than only for relations.

2. Romper perspectives are "flat"; they are limited to including a subset of the relations directly applicable to a concept. In contrast, view types can contain access paths of arbitrary length. Consequently, the views of a concept are not restricted to only direct properties (e.g., facts having the concept as their first arguments) and typically do have indirect properties.

3. Perspectives in Romper are domain specific (e.g., global), rather than concept specific. They identify what contexts are useful throughout the entire domain, what relations are relevant to general tasks that can be pursued in the domain. The approach of Romper assumes that a single monolithic domain perspective can be defined (perhaps by somehow composing the salience levels defined by several active perspectives). Furthermore, determining this single active perspective is left as an open problem. In contrast, KI addresses the problem of view selection by determining a set of candidate views and selecting one from among them, and KI does not require a single point of view to be adopted and maintained. Rather, during each processing cycle KI selects the view deemed

most relevant to the existing overall learning context. Different perspectives of a single concept or of multiple concepts can be composed and included in the learning context.

7.3.4 View dimensions

Suthers addresses the problem of automatically generating views from view specifications [Sut88]. He proposes a set of *epistemological parameters* by which views can be specified and automatically defined:

1. the *topic* specifies a concept of interest
2. the *model* specifies which among different ways to consider a topic (e.g., light as wave vs. light as particle) [3]
3. the *organization* identifies a category of relations to be included in the view (e.g., relation categories such as chronological, taxonomic, structural, procedural)
4. the *detail* indicates how much knowledge should be included (i.e., how large the resulting view should be in terms such as *Summary, Moderate, All*)

He further sketches out a procedure that extracts views given a knowledge base and a specification in terms of the epistemological parameters:

1. starting at the topic node follow paths of relations that are members of the slot category specified by the organization parameter

[3] This seems to correspond to modeling assumptions in qualitative reasoning.

2. extend paths in parallel through intermediate nodes only when those nodes are included in the model parameter

3. stop when the view is sufficiently inclusive as indicated by the detail parameter

Note that this procedure only extracts subgraphs of the knowledge base, as does KI.

This approach is quite similar to KI's view mechanism in that it cleanly separates knowledge about what is true in a domain and knowledge about what contexts are useful in the domain (as did Romper), while attempting to keep the granularity of the resulting contexts (e.g., views) at the concept level rather than at the level of the entire domain. Suthers' specifications for views require substantially more information than does KI's view mechanism (which simply requires a root concept and an applicable view type). Each combination of the last three parameters corresponds to a view type that could be defined. Consequently, these parameters should provide a way of succinctly defining sets of useful view types. Furthermore, Suthers does not provide methods for composing views or for selecting from among alternative views.

7.3.5 View Retriever

Acker's View Retriever program [Ack92, AP94] uses views to extract coherent sets of beliefs from a knowledge base while answering a given query. The View Retriever identifies four categories of view types and for each category defines a specification template (i.e., a format for specifying the view to be extracted from the knowledge base) and an algorithm for extracting the specified view. Each algorithm corresponds to a set of similar view types (e.g., a common pattern of access paths and node constraints).

The intended contribution is primarily one of content rather than method: while "KI's method for generating views is general-purpose ... [the View Retriever provides] a fairly complete set of [view] types useful in all physical domains" [Ack92, page 115]. In other words, the View Retriever identifies a small subset of all the view types that could be hand-crafted and proposes this subset as being useful and substantially complete for physical domains. Aside from the View Retriever's focus on content rather than method, there are three significant differences between the View Retriever and KI's view mechanism.

The first difference is the amount of information required to specify a view. KI requires only a root concept and a view type to unambiguously specify a desired view. The view retriever requires additional information, such as a *reference concept*, in order to specify a particular view. This additional information enables defining smaller, more focused views from a fewer number of view types than does KI's view mechanisms, but establishing the additional information (e.g., selecting the appropriate reference concepts) can be a significant requirement.

The second (related) difference is that KI's view mechanism addresses the task of selecting from among many possibly-relevant views. The View Retriever does not address this difficult problem. KI's view mechanism involves distinguishing which views are appropriate in a particular context (i.e., view-type preconditions), which appropriate views are most relevant to a given context (i.e., by measuring a view's interestingness and coreference with respect to the context). Although the View Retriever identifies a relatively small number of view-type categories, the space of possible views remains very large due to range of possible values of the additional parameters (e.g., the reference concept). Given new information, the View Retriever offers a plethora of possible

views that could be extracted and does not provide assistance with the problem of selecting among them. [4]

The third difference involves heuristics used to decide which facts will be included in the resulting view. In particular, when representing a concept as a type of one of its generalizations, the View Retriever excludes facts that are common to both the concept and the generalization. For example, if the knowledge base includes the facts:

(performs PhotosyntheticOrgan Photosynthesis)
(contacts PhotosyntheticOrgan LightEnergy)
(constituent PhotosyntheticOrgan Chlorophyll)
(hasColor PhotosyntheticOrgan Green)
(performs Leaf LeafPhotosynthesis)
(contacts Leaf LightEnergy)
(constituent Leaf Chlorophyll)
(hasColorLeaf Green)
(ako Leaf PhotosyntheticOrgan)
(ako LeafPhotosynthesis Photosynthesis)

the view resulting from the View Retriever representing *Leaf* as a kind of *PhotosyntheticOrgan* would include

(performs Leaf LeafPhotosynthesis)
(ako Leaf PhotosyntheticOrgan)
(ako LeafPhotosynthesis Photosynthesis)

but would not include

(contacts Leaf LightEnergy)
(constituent Leaf Chlorophyll)
(hasColor Leaf Green)

[4] This difference is not intended as a criticism of the View Retriever. Rather it is a natural consequence of the different uses of views by the View Retriever (answering queries) and by KI (guiding the elaboration of new information). In the former task, it is assumed that the query will contain information sufficient to complete a specification template for the View Retriever; therefore, each query specifies a single view or a small set of views. However, in the later task, the new information can not assumed to include information sufficient to complete a specification template (e.g., the reference concept), so many candidate views are viable and the problem of selecting from among the alternative views must be addressed.

Thus, the View Retriever excludes those facts that the subordinate concept inherited from the superordinate concept. However, when appraising the consequences of new information in this type of context (e.g., new information for leaves considered as photosynthetic organs), these common facts are quite essential. Consequently, they would be included in the views generated by KI.

7.3.6 Compositional modeling

The problem of determining a set of primitive facts from which to reason extensively also arises in model-based reasoning. Typically, a particular model-based reasoning task will not require reasoning with the entire model, and, for tractability concerns, only a portion of the model (one that is sufficient for the task) will be used. Determining this portion of the model (i.e., what components to include) is the model-selection problem.

Falkenhainer and Forbus propose an approach, called *compositional modeling*, that automatically performs model-selection [FF91]. Their approach advocates decomposing the representation of a model into coherent pieces called *model fragments*. When a reasoning task is encountered, the relevant model fragments can be identified and pieced together to form a customized model that is sufficient for the task but remains as simple as possible.

In this approach model fragments form a meta-theory: antecedents are constraints on the relevance of domain knowledge (e.g., modeling assumptions and operating constraints); consequents are domain knowledge (e.g., modeling equations). The antecedents of model fragments serve a similar role as do the preconditions of view types; both determine necessary conditions for when a set of domain rules should be considered. The consequents model fragments include sets of mutually interdependent axioms of domain knowledge

(e.g., qualitative modeling equations) and thus correspond to views.

Compositional modeling occurs in the context of a reasoning task, such as a query about the behavior of a physical system in some specified state. The terms in the query are used to index a set of model fragments. These fragments initialize the custom model being constructed. However, reasoning about some aspects of a model often requires reasoning about other aspects. Therefore, the initial model must often be extended.

Extending the initial model involves two phases. In the first phase, rules about the interactions of modeling assumptions are used to identify what aspects of the system must be considered in order to properly constrain the aspects contained in the initial model (e.g., the variables referenced by the query). In the second phase, a partonomic hierarchy is used to expand the model with additional model fragments until it contains some single system that includes as partonomic subordinates all components already contained in the model. This system is identified by finding a minimal cover in the partonomic hierarchy over all components already contained in the model.

The compositional modeling approach establishes the relevance of model fragments using coreference (e.g., terms common to both a model fragments and the query), rules about the interactions of modeling assumptions, and partonomic containment. The compositional modeling approach also advocates a fairly small grain size: model fragments often contain only two or three modeling equations. The grain size of KI's views can vary significantly but are generally larger than this. In the learning trials considered in Chapter 6, views created by KI contained an average of 28 propositions each.

The background theory involves quantified formulae; however, during a preprocessing step called *scenario expansion* the quantified formulae are

instantiated for the particular system being reasoned about. This results in a model of the entire system, represented as a set of ground modeling axioms organized into composable model fragments. Search for model components relevant to a particular query can be restricted to these ground model fragments. This represents a significant difference with KI which searches the quantified domain theory to identify sets of relevant axioms and only instantiates those that are selected for use. For example, when provided with new information about the leaf cuticle, KI instantiates only relevant concepts, such as the leaf, the leaf epidermis, leaf transpiration, etc., rather than an entire ground model of a plant.

Compositional modeling also differs from KI's view mechanism in that it requires a query to determine what background knowledge is relevant. KI must determine what is relevant to presented information without the benefit of a particular query to answer. Furthermore, only after the model is constructed does the compositional modeling approach reason with it. KI interleaves model construction (e.g., seeding the learning context with the new information and relevant background knowledge) with reasoning. The results of prior reasoning are used to guide subsequent model extensions (i.e., the explication of consequences during one processing cycle helps to determine the more interesting lines of reasoning to extend with the selection of an additional view during the next processing cycle).

7.3.7 Tripel

Rickel and Porter describe a system called Tripel that addresses the following problem [RP94]: given a set of ground model fragments and a prediction question (e.g., how does one variable, called the *driving variable*, affect

another variable, called the *variable of interest*) construct a model comprising a subset of the model fragments adequate to answer the question.

Tripel improves on the original compositional modeling approach by using chaining among the modeling equations in order to escape dependence on the partonomic hierarchies while extending the initial model. Using a type of backward chaining, Tripel identifies paths of modeling equations that constrain the variable of interest; each modeling equation included in such a path is included in the resulting model. This procedure ensures that all significant modeling equations connecting the dependent variables (e.g., the variable of interest) to independent variables (e.g., the driving variable) are included in the model. A variable is allowed to be independent only if it is not constrained by the driving variable or a dependent variable. Furthermore, Triple integrates time scale constraints to significantly simplify the resulting models by ignoring model fragments that don't have effects within the selected time scale.

In this approach, grain size is minimal: a view corresponds to a single domain rule (e.g., a single modeling equation). Modeling equations are relevant if they participate in a path of modeling equations that constrains the values of a dependent variable (e.g, the variable of interest) by the value of another dependent or an independent variable. As with conventional compositional modeling, Triple differs significantly from KI by its dependence on a given goal query to determine which modeling equations to include in the model and by not interleaving reasoning with the model (e.g., simulation) with model construction.

7.3.8 Composite model fragments

Iwasaki and Levy also propose improvements on the original compositional modeling approach by organizing model fragments into larger structures called *composite model fragments* [IL94].

As in Tripel, backward chaining from the terms (i.e., variables of interest) mentioned in the goal query identifies individual model fragments that are relevant to answering the query. Each identified model fragment indexes a candidate composite model fragment. The model-selection algorithm selects and integrates those composite model fragments that have compatible modeling assumptions.

Each composite model fragment includes a set of model fragments that represent behaviors of a common set of domain entities under different operating conditions. Therefore, the grain size of composite model fragments is larger than the model fragments composed by Tripel and the original compositional modeling approach. The composite model fragment approach suggests that model selection need not consider operating constraints. The justification of this position is that during simulation it is likely that the behavior of modeled entities will pass through several operating regions, therefore it is prudent to not exclude model fragments solely on the basis of operating conditions.

As with Tripel and conventional compositional modeling, the composite model fragment approach differs significantly from KI by its dependence on a given goal query to determine the relevance of prior knowledge and by not interleaving model-based reasoning with model construction.

7.4 Adaptation

This section surveys other research relevant to KI's suit of adaptation methods. KI differs from the described methods by one or more of the following:

1. KI does not assume a (or restrict the) criterial task
2. KI does not exploit (or require) a large set of positive and negative task instances
3. KI permits a very expressive representation language (e.g., one that includes skolem functions)
4. KI performs incremental learning
5. KI learns from general rules presented as new information
6. KI acquires new explanations of domain beliefs as well as resolving inconsistencies and filling gaps in the knowledge base

7.4.1 Explanation-based learning

Traditional approaches to explanation-based learning require a criterial task to determine when some path through an inference graph should be compiled into a shallow rule: compile the rule when doing so improves performance (e.g., in terms of response time) at that criterial task [Min88, Kel88]. Since KI cannot assume a criterial task, it must exploit other criteria to determine when an inference path should be compiled into a shallow rule. The rule macros of the representation language (Section 2.2.2) provide such criteria: compile an inference path into a shallow rule when that rule can be denoted by a rule macro and is not subsumed by an existing rule macro. All the advantages of the rule macro (e.g., efficient implementations of methods to complete

inferences with the rule, to index and present the rule, and to permit meta-reasoning about the rule) are attained by compiling the inference path into the rule. KI thus exploits the design of the representation language to determine when to engage in rule compilation rather than relying on a predefined criterial task.

The representation language, including the rule macros, reflects the intuitions of its designers about what distinctions and basic knowledge-base operations may be most useful in the domains represented and therefore are not purely task independent in a truly absolute sense. However, the rule macros are defined for basic types of inference (e.g., propagating inheritance, determining set membership, propagating slot inverses and generalizations, enforcing typing constraints on the arguments of predicates, etc.). Each such type of inference is so abstract as to be a completely different kind of "task" than criterial tasks common to traditional applications of explanation-based learning (e.g., identifying a cup or performing a set of benchmark calculus problems).

Thus, while KI's *learning goals* for explanation-based learning are explicit (which is the case for any explanation-based learning system), these learning goals are not defined by specific and relatively narrow expectations about the *application goals*. The *goal concepts* that are being learned are not defined by a criterial task as they are with traditional explanation-based learning systems (e.g., [MKKC86, DM86, Min88, Kel88]).

7.4.2 Theory refinement and inductive-logic programming

FOIL: FOIL improves on traditional approaches to concept acquisition and theory refinement by operating over a *clausal* representation language that is much more expressive than the attribute vector representation languages as-

sumed by the traditional leaning methods. The clausal representation language is a restricted form of first-order logic similar to logical programming languages in which all variables are implicitly universally quantified and functions (e.g., skolem functions) are not permitted. Statements in this language are clauses which include a single positive literal as a consequent and one or more literals as an antecedent. Inference proceeds by backward chaining through the consequent literal. The set of clauses whose consequent literal share a common predicate provide a clausal definition of that predicate in terms of the predicates referenced by the literals in the antecedents. Each antecedent defines sufficient criteria to satisfy the predicate. In other words, the set of all clauses is a logical theory in a stylized conjunctive-normal form; the antecedents of all clauses whose consequent literals reference a common predicate provide a definition of that predicate in disjunctive-normal form.

The learning task that FOIL addresses is:

Given: (a) the name of a target predicate
(b) n, the arity of the target predicate
(c) background knowledge comprising predicates whose definitions are known
(d) a set of n-ary ground tuples that are classified as positive or negative examples of the predicate

Find: a correct clausal definition of the target predicate in terms of the background knowledge

As a simple example, FOIL is shown to learn the binary predicate *connected* given a directed acyclic graph represented with the background binary predicate *linked* and the sets of binary tuples that completely define *linked* and *connected* for the given graph. The resulting clausal definition is:

$$connected(x\ y) \Leftarrow linked(x\ y)$$
$$connected(x\ y) \Leftarrow linked(x\ z)\ \&\ connected(z\ y)$$

This definition has been learned by analyzing only one graph but is reported to be generally valid. [5]

FOIL's method for performing this task involves performing a search for a set of clauses that admit all the positive examples while rejecting all the negative examples. This search is achieved by two nested iteration loops: each iteration of the outer loop generates a clause (i.e., a disjunct of the target concept); each iteration of the inner loop generates a literal to be included in the antecedent of the clause being constructed (i.e., a conjunct within that disjunct). Each clause admits some subset of the positive examples of the predicate while excluding the negative examples. After a clause is constructed, the positive examples it admits are removed from further consideration, and the subsequent clauses focus on admitting the remaining positive examples. This proceeds until no more positive examples remain or until no more clauses can be usefully constructed. Similarly, the first positive literal in each clause's antecedent [6] admits some subset of outstanding positive examples, and each additional positive literal imposes additional criteria that rejects negative examples.

This search is dominated by the inner loop, which picks a literal to add to the growing clause. FOIL uses a decision-theoretic metric in order to

[5]Note that this definition may not terminate if applied to a graph that contains cycles, yet the example learning task was not stated to be restricted to acyclic graphs.

[6]This nomenclature is consistent with the description of FOIL but may be confusing to those having standard background in logic: a *positive literal in the antecedent* corresponds to a negative literal in standard conjunctive-normal form

heuristically rank the expected contribution of each available predicate. Basically, this metric determines the number of correctly classified examples for each literal considered as a candidate for including in a derived clause. FOIL performs a hill-climbing search guided by this metric and is subject to non-optimal solutions when its metric guides it into local maxima. *Relational pathfinding* is an enhancement of FOIL's approach that uses a spreading-activation search to avoid local maxima while acquiring first-order concept definitions [RM92].

FOIL's learning method is not especially well suited to incremental learning because of its rather heavy reliance on the adequacy of the provided examples of the target concept [Qui90]: "FOIL requires all tuples of a relation to be available before any clause is generated." Therefore, as with all non-incremental learning methods, FOIL doesn't integrate new information into existing knowledge; it has no distinction between new and prior knowledge. However, it should be noted that other approaches to theory refinement have been shown to facilitate various degrees of incrementalness by simply accepting as input a previously revised theory [Moo92].

While FOIL improves on traditional approaches to concept learning by extending the expressiveness of the representation language it adheres to the ubiquitous assumption in traditional machine learning that learning occurs only within a problem-solving context.

KR–FOCL: KR–FOCL addresses the problem of revising incorrect (rather than just extending incomplete) knowledge bases [PB91]. KR–FOCL is an extension of the learning program FOCL (itself an extension of FOIL) which uses both induction and explanation-based learning to fit a given theory to a set of solved task instances.

As with other traditional approaches to theory refinement (and unlike KI) learning in FOCL necessarily occurs within a problem-solving context. Given a target concept defined in a nonoperational language, a clausal rule base that maps operational expressions into nonoperational expressions, and a set of positive and negative examples of the target concept presented in the operational language, FOCL derives rules expressed in the operational language that correctly classify all the examples.

FOCL proceeds by using explanation-based learning, guided by the information-theoretic metric of FOIL, to operationalize the target concept. This involves compiling inference graphs that establish the correct classifications of the positive examples into single-step rules. However, if the existing rules are incorrect and fail to reject all negative examples, literals are inductively added to the antecedents of the derived rules to ensure all negative examples are rejected. Similarly, if the existing rules are incomplete and fail to admit all positive examples, new rules are inductively derived to ensure that all positive examples are admitted. In both cases, the information-theoretic metric is used to select the literals inductively added to the derived rules.

While FOCL is capable of using an initial theory that may be incomplete or incorrect to derive operational rules that correctly classify the training examples, it does not correct the initial theory. KR–FOCL exploits the results of FOCL and four heuristics to suggest to a knowledge engineer candidate corrections to the initial theory. For example, if a rule from the initial theory is never used by FOCL (i.e., is never operationalized) KR–FOCL might suggest that the rule is spurious and should be discarded. A second heuristic suggests that an induced literal added to reject negative examples might be added to the antecedent of some superordinate rule in the inference graph that establishes

the target concept. KR–FOCL serially asks the user to consider each possible revision independently.

FOCL requires a set classified examples of the target concept, and learning occurs only in the context of classifying the given training examples. Explanation-based learning only occurs when the rules of the initial knowledge base are used to establish how a positive example is admitted by the target concept. Induction occurs only when positive examples are not admitted or negative examples are admitted. The set of classified training examples is essential to the information-theoretic metric used to guide learning.

In contrast, KI is not endowed with a set of solved task instances. Instead, KI must discover learning opportunities among the consequences of new and relevant prior knowledge. Learning opportunities that suggest inductive refinements to the knowledge base arise when conflicting facts are established (e.g., facts that violate argument-typing constraints). Rather than relying on an information-theoretic metric to guide learning by fitting the domain theory to a set of solved task instances, KI exploits the structure of the inference graph that establishes the conflicting facts to identify a set of candidate knowledge-base revisions (i.e., KI identifies the essential facts that support the conflict). Furthermore, KI exploits the different levels of confidence of rules participating in the nonmonotonic inference graphs that establish a conflict: KI prefers knowledge-base revisions that refute beliefs established with the weakest conviction (e.g., refuting assumptions is preferred to refuting facts having monotonic support). Finally, rather than serially presenting to the user each candidate knowledge-base revisions, KI uses the conflict-resolution hierarchies to sort the candidate knowledge-base and to identify which revisions are strongest in the sense that they correct the greatest number of encountered conflicts. The user

can peruse conflict-resolution hierarchies to review the candidate revisions in order of strongest to weakest.

Demand-driven concept learning: In BLIP [Wro89] there are no strong commitments made to fixed and narrow learning goals. The user and system work together to develop a theory that correctly characterizes a set of ground observations. When new information introduces inconsistencies the learning methods modify the exiting theory (e.g., by introducing new concepts) to resolve the inconsistencies. However, BLIP includes no mechanisms to guide elaboration of new information to detect tacit inconsistencies and is therefore restricted to either shallow elaboration or small theories represented with inexpressive languages. In contrast, an essential aspect of KI involves using views to guide elaboration. This permits the completion of deep inference paths using very large theories that are represented with a very expressive language. Furthermore, BLIP conforms to the traditional machine-learning paradigm of *fitting a theory to ground observations*: learning behaviors only resolve inconsistencies and inductively characterize ground observations according to a set of pre-enumerated rule schemas. In contrast, KI is capable of learning from general domain rules without the use of ground observations: hypothetical simulation is performed to detect tacit learning opportunities. Consequently, explanation-based learning methods are adopted in KI over similarity based, data-intensive, inductive learning methods.

7.4.3 Knowledge acquisition

Traditional approaches to machine learning assume that target knowledge (representations of valued domain knowledge) cannot be directly identified and added to the knowledge base; rather the user must think of an appro-

priate sequence of examples in order to guide the learning system to formulate the target knowledge. Consequently, the user must understand the learning system, its strategies and background knowledge in order to lead it to the desired target knowledge [Mor91]. Thus, in general, traditional machine learning systems construct target knowledge from examples of task instances provided by a domain expert. In contrast, knowledge acquisition systems elicit target knowledge from a domain expert. Typically, knowledge acquisition systems are guided by a problem-solving model and instantiate the problem-solving model with the provided domain-specific knowledge.

EXPECT [Gil94] is a knowledge acquisition system developed to identify unharmonious interactions between new and exiting knowledge and thereby prevent gaps, redundancies, and inconsistencies. Traditional knowledge acquisition tools assume a particular problem-solving method that identifies how each piece of knowledge may be used during problem solving. For example, when a new class is introduced, classification systems can solicit knowledge to determine how instances of the class are to be recognized. The problem-solving method guides the acquisition of knowledge, but it also restricts the applicability of the knowledge acquisition tool since the tool is designed to support acquiring knowledge only for systems that exploit that particular problem-solving method. This applicability restriction is permanent since the expectations about the assumed problem-solving method are (typically) programmed into the tool.

The motivation of EXPECT is to retain the advantages of assuming a problem-solving method (e.g., determining what knowledge is relevant and what interactions between knowledge fragments may occur) while avoiding the restricted applicability from permanently committing to a single problem-

solving method. To do this, EXPECT requires the problem solving method to be defined explicitly and declaratively as a set of problem solving modules. These modules specify how problem-solving goals can be decomposed into subgoals, and thus identify how domain knowledge can interact during problem solving. EXPECT can then examine these problem-solving modules to determine what knowledge is relevant to, and what interactions may occur among knowledge fragments during, problem solving in order to guide knowledge acquisition. Because the assumed problem-solving method is represented explicitly and declaratively in the knowledge base rather than being programmed into the knowledge acquisition tool, different problem-solving methods can be represented by changing the models. Therefore, EXPECT does not rely on, and is not restricted to, any single problem-solving method. However, unlike KI, it does require explicit models of the anticipated application tasks in order to guide knowledge acquisition.

When gaps or inconsistencies are detected, EXPECT creates suggestion memos for the user that identify both the problem and possible fixes, and it pushes these memos on to an agenda where the user can peruse them rather than interrupting problem solving. This approach is very similar to KI.

A static analysis identifies what properties may be used during problem solving for various classes of concepts. For example, a problem-solving module indicates that to provide advice about transportation the system may need to know which seaports are available at a location. The results of this static analysis are expectations about what knowledge should be provided along with instances of classes of concepts. For example, when the user introduces Los Angeles to the system as an instance of location, a suggestion is created that solicits the user to indicate what seaports can be found in Los Angeles.

Unfortunately, this technique appears too permissive: the system would also solicit the seaports local to all known locations (e.g., Kansas City, Treaty Oak, the MCC parking lot, Fred's house, etc.,).

EXPECT has a catalog of problems and for each has a set of repair strategies. For example, when a suggestion is created to solicit a particular property of a particular concept (as in the case above of when Los Angeles is introduced), it identifies for the user the following options:

1. specify the value of the property for the concept
2. remove the concept
3. modify the problem-solving method so that it will not require the property

Having a catalog of errors and repair strategies for each is similar to KI's approach. Unlike KI, EXPECT also has a cached set of analyses it performs whenever new information is provided. For example, when a new instance is introduced to the knowledge base, the analysis that EXPECT performs is:

1. If the class given for the instance is too general (e.g., if Los Angeles were specified to be an instance of *Thing* rather than *Location*), then present the user with specializations of the indicated class and request that one be selected as the new class for the instance.
2. Identify which properties of the class given for the instance may be used during problem solving, and for each property that does not already have a value request the user specify a value.

There are five methods used to integrate changes to problem-solving modules. Three of them are:

1. If the goal of the new module does not achieve any subgoal of known modules, then notify the user.

2. If there are syntactic errors in the definition of the new method, then notify the user.

3. If the new method uses a property for a class that was not required by other problem-solving modules, then request the user specify the value of the property for every instance of the class.

This last method may be problematic. For example, if a new module for a diagnostic system references the blood pressure of the parents of a person, then the system requires the user to specify the blood pressure for every ancestor of every person!

Unlike EXPECT, KI's learning behaviors transcend resolving inconsistencies and knowledge-base gaps to include explanation acquisition, rule compilation, and teleological learning.

7.4.4 Teleological learning

Among the most interesting learning behaviors demonstrated by example of Figure 1.1 is the teleological learning. Elaboration reveals the prediction that by restricting transpiration, the cuticle inhibits dehydration and facilitates the leaf's good health. This provides a *teleological explanation* of the new information: the cuticle establishes the *biological goal* of facilitating good health. Thus, the "function" of the cuticle, to restrict water loss, has been identified; it explains why leaves have cuticles.

Type of Knowledge	Type of Query
structural	what are the static properties?
behavioral	what are the dynamic properties?
causal	how is a dynamic property achieved?
teleological	why is a dynamic or static property included?

The knowledge types appearing in the first column support answering the types of queries appearing in the second column. These knowledge types are (typically) related hierarchically: teleology requires causality; causality requires behavior; behavior requires structure [Kui85].

Figure 7.1: Types of knowledge

Teleological knowledge (i.e., knowledge of purpose) plays an essential role in our understanding of biological domains [Sim81, Dow90] and engineering domains [Sim81, DeK85, Fra93]. Teleology, like causality, plays an important role in learning by providing a higher-order description of many important situations in a domain. These higher-order descriptions are essential to understanding and, therefore, to learning about those situations. Teleological knowledge supports answering a different class of queries than do other types of knowledge [Fra93] (see Figure 7.1). Consequently, some knowledge acquisition tools for knowledge-based systems attempt to explicitly acquire causal and teleological knowledge [KBR91, Gru91]. Furthermore, tools explicitly exploit teleological explanations to guide analogical reasoning in a variety of domains [KC85].

Franke has noted that teleology plays several essential roles in design domains:

1. good explanations of design rely on purpose [DeK85]

2. knowledge of purpose can guide the diagnosis of broken artifacts so that

only those components mandated to achieve the violated design specification are considered

3. knowledge of purpose in prior designs can guide subsequent design by identifying prior design properties that achieve behaviors desired for the current design [KC85]

He has developed TeD, a formal language to represent teleological descriptions [Fra93]. The representation and acquisition of teleological knowledge in his approach are discussed next.

Representation of teleological descriptions: In Franke's approach, descriptions of purpose are represented as guarantees that some specification will be satisfied by a given design. This approach provides a well-defined account of purpose; however, it does not completely capture some intuitive aspects. Specifically, using *guarantee* as a primitive for teleological descriptions in some cases is too strong: static or dynamic properties may simply promote, or contribute to, rather than guarantee, achieving goals. For example, in behavioral domains, certain actions promote, but do not guarantee, desired outcomes (e.g., consuming low fat foods, exercising, advertising a product, studying for a test), yet we'd like to be able to discuss the teleological properties of these behaviors. Thus, the notion of "guarantee" is too strong when a property promotes or contributes to achieving goals. [7]

This concern is partially addressed in Franke's ontology by permitting *conditional guarantees*. A conditional guarantee states that under certain con-

[7]Note that Franke focused on qualitative models for device design; in this context the primitive *guarantee* seems more appropriate.

ditions a static or dynamic property is guaranteed to achieve a goal. However, this approach requires explicitly stating the conditions in which the goal is achieved. In many situations this will be quite difficult due to the qualification problem [McC77] (see Section 1.2.1). KI's representation of purpose is weaker; it denotes that assuming every thing is as expected in a situation it is reasonable that a given property (e.g., having cuticle) contributes to the achievement of a goal (avoiding dehydration). Achieving the goal is not guaranteed by the property (e.g., even leaves with cuticles can become dehydrated), but explicitly enumerating all the conditions under which the goal is guaranteed is not required.

Acquisition of teleological descriptions: In Franke's approach, teleological descriptions are acquired during the design process. When a design is modified to achieve a previously unattained design goal, the designer triggers the acquisition of the teleological description for the design modification. For example, when adding a new component to a device design causes the device to satisfy some required specification, then the acquired teleological description for that component is that it guarantees the satisfied specification. Subsequently, a query about the purpose of that component can be answered by referring to the specification it satisfied.

Capturing teleological descriptions during the design process has some intuitive and pragmatic appeal. For example, it permits referencing behaviors (that are guaranteed not to occur) which the final design couldn't ever produce [Fra93]. However, when several aspects of the design each play individually necessary and jointly sufficient roles in achieving the specification, and are added to the evolving design in separate stages, only the aspects included in

the last design change will be referenced in the teleological description. KI avoids this by analyzing the explanation of how a domain goal is achieved in order to identify all behaviors of all components that enable the goal to be achieved; each such behavior of each component is identified as a teleological description of the component (i.e., an account of its purpose).

The treatment of teleological descriptions in KI and in other research are complimentary. TeD provides a formal language for representing teleological descriptions that is not specific to particular goals. KI demonstrates that very general domain-specific goals (e.g., facilitating good health) can be used during learning to automatically acquire teleological descriptions, and that new teleological descriptions can guide further knowledge acquisition (via analogies). Purpose-directed analogy [KC85] provides a method for suggesting properties about a target concept when given both a purpose and a set of previously-described base concepts. KI automatically (in the course of elaboration) suggests the purpose of a property and then exploits the explanation of that purpose to automatically identify target concepts which might also have the property.

Chapter 8

Conclusions

One of the primary goals of Artificial Intelligence is to develop systems that learn. Achieving this goal requires developing computational learning tasks as well as methods to perform them. This dissertation describes exploratory research that investigates knowledge integration as a machine learning task.

8.1 Summary

Knowledge integration is the task of identifying how new and prior knowledge interact while incorporating new information into a body of existing knowledge. This task is pervasive because substantial bodies of knowledge must be developed incrementally: segments of knowledge are added separately to a growing body of knowledge. This task is difficult because knowledge engineers cannot anticipate precisely how new and prior knowledge will interact, and unexpected interactions may require additional changes to the knoweldge base. Performing knowledge integration involves determining and affecting these changes. The goals of this research include characterizing knowledge integration as a machine learning task, developing a computational model for performing knowledge integration, and implementing the model as a machine learning program.

The study of knowledge integration and methods that perform it is important both for pragmatic concerns of building knowledge-based systems and for theoretical concerns of understanding learning systems. By identifying conflicts and gaps in knowledge, knowledge integration facilitates building knowledge-based systems. By avoiding unnecessary restrictions on learning situations, knowledge integration reveals important sources of learning bias and permits learning behaviors that are more opportunistic than do traditional machine learning tasks.

REACT is a computational model that identifies three essential activities for performing knowledge integration. *Elaboration* identifies how new and prior knowledge interact. The limited resources to explore the interactions of new and prior knowledge requires methods to focus attention. This focus is achieved by restricting elaboration to consider only particular, relevant segments of prior knowledge. *Recognition* selects the prior relevant knowledge considered during elaboration. By identifying the consequences of new information for prior knowledge, recognition and elaboration reveal learning opportunities, such as inconsistencies and gaps in the knowledge base. *Adaptation* detects and exploits these learning opportunities by modifying the new or prior knowledge.

KI is a machine learning program that implements the REACT model. It identifies and resolves conflicts between new and prior knowledge as it integrates new information into a knowledge base. KI builds on existing methods of machine learning both to learn from general rules, rather than fitting a theory to ground training instances, and to learn in the absence of strong use expectations. It builds on approaches to first-order belief revision by addressing the difficult problem of guiding inference while searching for significant consequences of new information. KI uses views to structure the knowledge base

into contexts of mutually relevant beliefs; view types support the automatic construction of views. Views appear to be an effective mechanism to focus attention. KI implements a suite of adaptation methods that detect and exploit learning opportunities. These methods support both traditional learning behaviors, such as knowledge compilation, and novel ones, such as *teleological generalization* and *knowledge deepening*. Empirical studies demonstrate that KI provides significant assistance (e.g., *learning gain*) to knowledge engineers while integrating several test scenarios into the Botany Knowledge Base.

8.2 Research contributions

Knowledge integration reflects an evolution in knowledge-based systems from expert systems, which are dedicated to performing a single task (such as classification) in a restricted domain, to more broadly scoped knowledge-based systems, such as those containing foundational knowledge that can be applied to a variety of tasks within one or more domains. The task of knowledge integration differs from traditional machine learning tasks by rejecting commitments to specific predetermined learning goals based on strong use expectations (such as the speed or accuracy of performing classification). Consequently, it facilitates more opportunistic learning methods than do traditional tasks since learning opportunities are not excluded due to constraints on either content of the training or the eventual uses of the acquired knowledge.

Intuitively, knowledge integration is intended for a different learning situation than are traditional machine learning tasks. The traditional tasks are motivated by the concern that the primary barrier to developing expert systems is elucidating and representing the domain knowledge required to perform the application task. Traditional machine learning methods address this concern

by fitting a domain theory to a set of ground observations; they usually acquire or optimize general rules that reflect regularities in the training data. The task of knowledge integration is motivated by a different concern: the interaction of new and prior knowledge is hard to accurately predict; learning methods should determine those interactions to resolve conflicts and fill gaps in the knowledge. Consequently, the elucidation of general domain rules from ground observations is not the primary concern, and, for example, knowledge integration permits learning from new information that includes general domain rules.

Evidence for the feasibility of performing knowledge integration has been established by developing:

- REACT: a computational model for performing knowledge integration
- KI: a learning program that implements the model and performs the task

REACT provides a functional decomposition of the task; it identifies three activities that appear individually necessary and collectively sufficient to perform knowledge integration. KI provides an existence proof that the computational model can be implemented and the task performed. To escape the strong use expectations associated with traditional machine learning tasks, KI adopts generic and domain-appropriate learning goals (e.g., promoting consistency and acquiring teleological explanations). To escape intractability, KI admits resource bounds, sacrificing "complete" solutions (which are not possible, in general) for approximate and tractable ones.

This dissertation identifies use expectations as an important aspect of learning tasks and the methods that perform them. It suggests that use expectations guide learning systems by specifying what to learn, and it suggests that

use expectations are perhaps the single most powerful source of both learning bias and brittleness in knowledge-based systems. Consequently, casting learning as knowledge integration challenges the assumption that learning occurs (only) within the context of strong use expectations (e.g., during problem solving) and suggests that for many learning opportunities the appropriate focus of learning is the *comprehension* of new information rather than its *application*.

REACT identifies three essential activities for performing knowledge integration:

- during recognition, the learner identifies prior knowledge that is relevant to the new information
- during elaboration, the learner identifies interactions between the new and prior relevant knowledge
- during adaptation, the learner modifies new or prior knowledge to satisfy the learning goals (e.g., to resolve inconsistencies)

Methods that perform these activities determine, more or less directly, what is learned and are important sources of learning bias.

Two important issues for recognition methods are determining both the grain size of the segments of relevant prior knowledge and a *principle of relevance* that prioritizes alternative segments for consideration during comprehension. KI uses view types to determine grain size, and its principle of relevance is based on the notions of the connectedness, interestingness, coreference (i.e., the extent to which two segments of knowledge share common terms), and (most importantly) user-defined patterns of mutually-relevant beliefs.

One important issue for elaboration methods is operationalizing knowledge (e.g., new information or retrieved prior knowledge) so that the system's inference procedure can be applied to it. KI uses a form of hypothetical reasoning to explore the consequences of knowledge encoded as general rules. A second important issue is distinguishing inference paths that are significant for learning from those that are not; KI provides an abstraction of the TMS level that preserves only those inference paths reflecting conceptually (vs. formally) distinct reasons for establishing a fact.

An important issue for adaptation methods is extending the system's truth-maintenance capabilities so that knowledge acquired by learning (e.g., compiled rules) is indexed by the beliefs used to acquired it. For example, when a nonmonotonic fact is retracted, the rules formed by compiling inference paths that establish the fact must be reviewed; if they are no longer warranted, they, too, should be retracted.

8.3 Agenda for further research

This research adopts the *generate and test* methodology of experimental research in computer science. The results reported here represent roughly two cycles of constructing, then studying, an implemented computational system, roughly five years of committed study. As such, these results are inherently exploratory and preliminary. They represent a beginning, not an end, to the study of machine learning as knowledge integration. The following sections identify some important issues for further study.

8.3.1 Integrating top-down and bottom-up control

Traditional machine learning systems and KI represent two extremes of a continuum defined by the generality of the learning goals: traditional systems adopt very specific learning goals based on very strong use expectations; KI adopts very general learning goals based on very weak use expectations. While it is important to fall back on general learning goals when reliable use expectations are not available (e.g., during incidental learning), it is also important to exploit whatever reliable use expectations are available to guide learning.

Currently, learning in KI occurs through a primarily bottom-up process: comprehension (e.g., the activities of recognition and elaboration) explores the consequences of new information to reveal learning opportunities. Adaptation methods are "consumers" of this exploration: they detect and exploit learning opportunities revealed by the exploration without directing its course. However, when reliable and strong use expectations are available, it might benefit learning to permit a top-down control regime to supersede the default, bottom-up process. Selected methods of adaptation that are "activated" by the available use expectations could intervene during comprehension to determine whether particular learning opportunities can be established in a goal-directed fashion. The *spontaneous use expectations* (discussed in Appendix B.2.2) are one general category of adaptation methods that can be triggered by new information and used to guide learning. A second general category includes domain-specific learning goals (also discussed in Appendix B.2.2). For example, given new information describing a seedless grape, the domain learning goal of determining the consequences of a missing part, together with prior knowledge of the reproductive function of seeds, would actively guide learning

to acquire knowledge about how the grapes are produced. Future research must study flexible schemes of adapting a range of both general and specific learning goals to particular learning events, so that the strongest reliable available use expectations guide learning.

8.3.2 Next generation views and view types

The problem of focusing attention (e.g., controlling inference) is perhaps one of the great computational issues of artificial intelligence. KI's use of views and view types provide a promising approach. Their advantage over traditional schemas is that they provide an extra level of indirection: only a relatively small set of view types must be defined explicitly rather than every possible individual view; individual views are then defined automatically only as they are needed. While the notion of views is not original with this research, the use of view types to construct views automatically is. View types enable manual specification of common patterns of mutually relevant propositions. Furthermore, they identify a type of meta-knowledge useful for controlling inference. This contribution has triggered significant additional investigation of different approaches to representing this meta-knowledge and exploiting it while generating descriptions of domain concepts [Ack92]. Other important areas of future work include developing strategies for acquiring this meta-knowledge (e.g., learning view types) and making reasoning with views more purposeful.

Learning view types: While view types drastically reduce the effort in structuring knowledge, they still must be defined manually. It remains an interesting problem to develop methods that automatically acquire view types. Two possible methods pursued while experimenting with KI are:

1. *Bundling the predicates that are likely for a collection into groups based on paths through the taxonomic hierarchy:* Each unique path through the taxonomic hierarchy that connects a concept to the taxonomic root concept (e.g., *Thing*) suggests a different view type comprising the predicates introduced along that path. The resulting view types are "flat" (i.e., their access paths are of length one). However, these view types could form primitive building blocks that compose into deeper view types (i.e., those having longer access paths).

2. *Extracting views from inference graphs that establish interesting facts:* Each derivation of a fact comprises a set of inference paths. By changing the ground terms referenced in the inference path into variables, the inference paths can be abstracted into a set of general access paths that define a view type. Each taxonomic fact (e.g., facts whose predicate is *isa*) in the inference path suggests a taxonomic node constraint in the view type. The resulting view type identifies contexts in which the inference path can be re-established. The inference path provides a basis for the mutual relevance of those propositions included in views that instantiate the resulting view type.

While preliminary research in these directions is promising, automatically acquiring view types remains a largely unexplored problem.

Making view types more purposeful: The inferences completed using views should have greater focus and purpose. As Minsky suggests [Min81], not only should contexts identify relevant concepts, they also should identify the typical questions that are relevant and interesting within the context. For

example, in biology a view type *QuaVisualObject* could include standard questions, such as:

1. What colors and shapes does it have?
2. What sexual or symbiotic partners or prey are attracted to it?
3. What predators cannot perceive it (e.g., due to camouflage)?
4. What predators are repulsed by it (e.g., due to warning signals)?

Ideally, these view-type questions should be learned after a few examples in which the learner selected a view (relying on the default view selection heuristics) and "noticed" that it led to a significant result (i.e., some particularly interesting fact was established). The view-type questions focus attention towards re-establishing the significant result during subsequent use of that view type.

There is a trade-off between remaining opportunistic during elaboration and incorporating predetermined focus. One possible approach to finessing this trade-off is to chain forwards along all paths to depth n, and then chain backwards on context goals to depth m, where m is significantly greater than n. Investigating such trade-offs remains an important direction for further research.

8.3.3 Evaluation via field tests

Future work must adapt KI to a variety of applications. The best way to evaluate the utility of the learning gain achieved by KI is to have it used during the construction of a variety of knowledge-based systems. In the course of interacting with KI (e.g., by accepting, rejecting or modifying the

knowledge that KI acquires), knowledge engineers will provide an important and pragmatic assessment of KI's utility for facilitating knowledge-base construction.

A related and very important direction of future work involves exploring suitable methodologies to evaluate systems that acquire and use foundational knowledge. Current methodologies for evaluating knowledge-based systems commit to strong use expectations; methodologies must be developed to evaluate systems (e.g., learning systems) that do not make these same commitments.

Appendix A

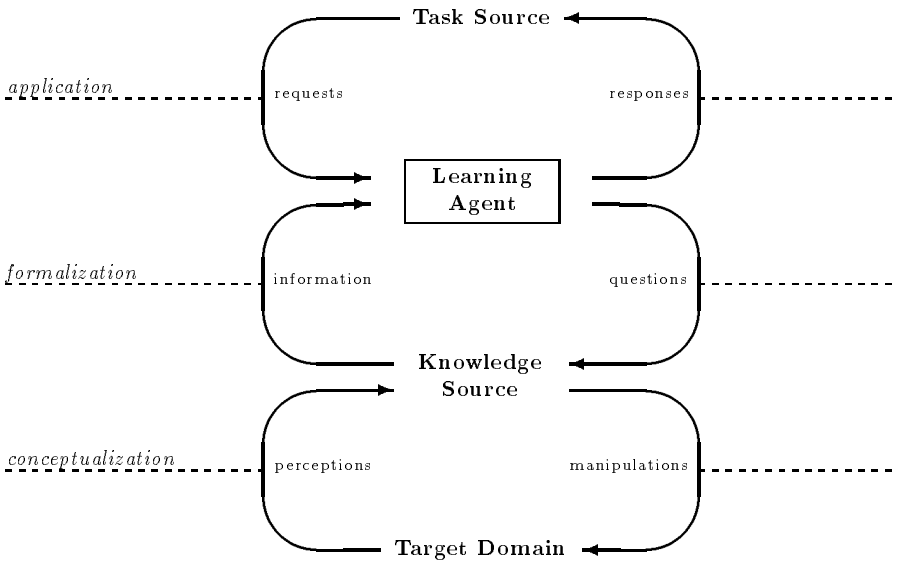
A Characterization of Learning

When defining a new learning task, it is useful to consider the context in which that learning is assumed to take place. The first section of this appendix describes the learning environment; it identifies the essential components within a learning situation and their interactions, and it considers the consequences of learning only within problem-solving contexts. The next section develops an intuitive definition of learning appropriate for this environment. The final section characterizes machine learning in terms of a state-space search problem.

A.1 The learning environment

In order to develop a learning task specification, it is convenient to first characterize the learning agent as well as the circumstances in which learning occurs. Figure A.1 illustrates a model [1] of a learning environment comprising four significant components: the target domain, the knowledge source, the learning agent (e.g., a knowledge-based system), and the task source. The model further identifies six types of interactions among these components and three significant thresholds in the evolution of knowledge about the target do-

[1] This model relects a synthesis of the work of Bransford [Bra79], Morik [Mor89, Mor91], Nilsson [Nil91, GN87], Newell [New90], and Porter [PLM+88].



A componential model of the learning environment identifying four roles and six types of interaction that can occur among the roles and three thresholds that are crossed during the evolution of knowledge.

Figure A.1: The Learning Environment

main: conceptualization, formalization, and application.

A.1.1 The target domain

The target domain is a set of phenomena about which the learner is to reason. This can be a natural, or real world, domain – such as geology, botany, medicine, or meteorology – or it might be an artificial domain – such as mathematics, chess, or Tolkien's Middle Earth.

The target domain does not prescribe any particular application task; rather, a range of general tasks can exist. For example, tasks such as diagnosing conditions from symptoms, determining treatments appropriate for conditions, predicting what conditions will follow from current conditions, teaching basic

anatomy, physiology and medical procedures, etc. could all apply in the domain of human medicine. Furthermore, the target domain can change over time.

A.1.2 The knowledge source

The knowledge source defines a *conceptualization* of the target domain and provides information about this conceptualization to the learner. A knowledge source is any entity that can fill this role. It may be some set of perceptual components of the learning agent that interpret direct experiences of the target domain (e.g., a vision subsystem that provides information about a physical scene), or it can be some intermediary that independently acquires beliefs about the target domain and then presents them to the learner (e.g., a human teacher, a textbook, another knowledge-based system).

The conceptualization is a characterization of the target domain that is presented to the learner. It identifies the concepts (e.g., entities, relations, attribute values) and their interrelationships that are presumed to exist in the target domain [GN87]. There may be biases (e.g., personal, cultural) in how the domain is conceived that are reflected in the conceptualization. The conceptualization arises from either interactions (direct or indirect) with the target domain or from interactions with other knowledge sources (e.g., while never having visited Australia, a Texas school teacher can present material gleaned from books on Australian wild life). Aspects of the conceptualization need not actually exist in the target domain; they are required only to exist in the way the domain is conceived by the knowledge source. For example, the notion of *causation* may or may not actually exist in physical domains such as medicine, but it has an esteemed place in many conceptualizations of such domains.

One or many knowledge sources can interact with the learner during learning events, and each knowledge source that interacts with the learner can define a more or less different conceptualization of the target domain. Therefore, the learning agent cannot rely on receiving only instruction that reflects either a single conceptualization or a set of mutually consistent conceptualizations. [2] Conceptualizations can differ on the concepts they recognize, or on the beliefs they hold to be true about those concepts; even experts often disagree about some aspects of their domains. Furthermore, the conceptualization of an individual knowledge source is not necessarily static and can vary over time.

Conceptualizations restrict what concepts are referenced by the information presented to the learner. Through this restriction, the knowledge source partially determines what aspects of the target domain are worth knowing. Therefore, knowledge-source conceptualizations manifest an important bias in the knowledge acquired by the learner (e.g., the propagation of cultural bias). Furthermore, the contents of the information included in training reflects a prioritization among the aspects of the conceptualization. For example, rather than providing either a complete account of all available knowledge in a domain, or a random sampling of knowledge that values all beliefs equally, textbooks contain a finite amount of selected accounts. The choices that determine what is included, and in what order, reflect the author's intuitions about

[2] This observation is not at all intended to constrain or prescribe what role the learning agent takes in responding to (e.g., resolving) incommensurate conceptualizations. For example, the learning agent may search input for specific types of incompatibilities (e.g., task specification errors [FN88]), or the learning agent may permit different conceptualizations and resolve them internally (e.g., with contexts [Reb83, Cla90, Guh91]), or the learning agent may not address or even notice incommensurate conceptualizations until they result in conflicts in the growing knowledge base (e.g., KI, the program described in Chapters 3 – 5. However, KI does interactively present the consequences of new information so that incommensurate conceptualizations might be noticed by the users as they are defined).

what domain knowledge is more important for the reader to possess and, consequently, what domain knowledge is less important. These choices may be made with few or many expectations about the eventual applications of this knowledge; they are certainly made without a complete knowledge of specific tasks in the reader's future that might reference the target domain. [3]

A.1.3 The learning agent: a knowledge-based system

The learner is a knowledge-based system. It is essentially a representation system; that is, it reflects one or more conceptualizations of the target domain. This representation can be used to respond to requests about the target domain. The system's representation of the target domain is called the *domain theory*, or simply the theory, and it comprises a set of statements expressed in a *representation language*.

The language of representation defines a space of (i.e., a set of candidate) legal statements; the space of knowledge bases is the powerset of this set of legal statements. *Propositions* are simple and basic statements in the language; they are composed of constituent constructs (e.g., terms, relations). [4] Propositions that apply only to specified individual concepts (e.g., terms appearing as arguments that are constants rather than variables) are called *ground*

[3]Some textbooks (e.g., ones on how to play poker) are much more focused on solving particular tasks than other textbooks (e.g., on medicine, history, biology, ...). The knowledge included in some textbooks (and in some knowledge-based systems) must be determined by their authors without the benefit of knowing which specific tasks in the domain will be encountered by the readers (or the knowledge-based systems) and, therefore, what specific knowledge in the domain will utlimately be required. Thus sometimes considerable use expectations may be available (e.g., authoring a recipe book), but very often few or only very general use expectations may be available (e.g., authoring an encyclopedia or a textbook or a knowledge base on botany).

[4]Note that the language of representation does not specify an ontology or any elements of an ontology; rather, it specifies how statements, including those that define the ontology, can be expressed. Often in ML literature, the language includes ontology; here it does not.

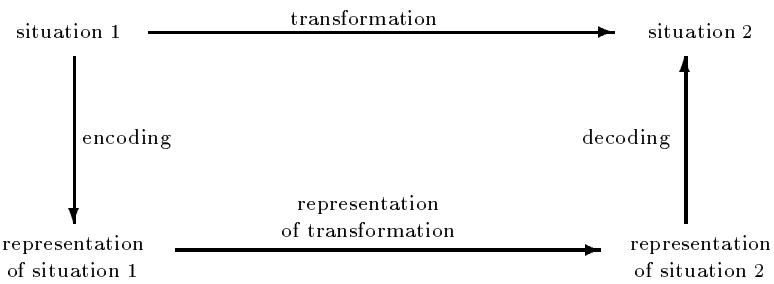


Figure A.2: The Representation Law

propositions or *facts*, while those propositions that reference or apply to sets of concepts (e.g., they include references to quantified variables) are *rules*. The representation language also specifies both a set of operators (or procedures) that apply to propositions and the semantics (behavior) of those operators (e.g., rules of inference for using the operators to derive propositions from other propositions).

The domain theory comprises a set of *beliefs*; each belief is a proposition that is presumed to be true in the target domain. Beliefs currently cached in memory are said to be *explicit*; those not cached in memory but that can be derived from explicit beliefs using the representation language's operators are *implicit*. [5] The *knowledge base* comprises the explicit beliefs.

Newell characterizes the essential obligation of a representation with *The Representation Law* [New90] (Figure A.2). In Figure A.2, the notion of

[5]Note that this *explicit* vs. *implicit* distinction is not the same as the *explicit* vs. *tacit* distinction made in contemporary schema theory in human psychology: the former relates to whether or not beliefs are directly stored in memory; the latter relates to whether or not beliefs are known consciously. In both, the status of particular beliefs can shift between explicit and otherwise.

transformation is a very general one. Transformations include not only processes in the target domain that physically manipulate or alter domain objects but also any conceptual mapping from one situation to another. Situations include sets of propositions as well as the constituent constructs of propositions, such as terms and relation constants. Transformations thus include decomposing sets of beliefs into individual beliefs and decomposing beliefs into constituent constructs (e.g., relations and terms). The representation law requires that a correspondence occur between concepts in the target domain and constants in the representation and that this correspondence be preserved through transformations. *Commitment* to the representation by the system's users is warranted only to the extent that the representation law holds between the domain theory and the target domain in those situations and transformations relevant to the application tasks.

The *target theory* is that which learning attempts to produce: a representation of the knowledge sources' conceptualizations adequate to support correct responses to all requests from the task sources, and thereby to support strong commitments. It is the theory-wide analog of the *target concept* in concept learning; it maximizes the correspondence of every constant symbol (i.e., every object in the universe of discourse, every relation) in the theory to what that symbol denotes in the conceptualization of the domain. The target theory is an idealization that may never be realized. The domain theory changes over time through learning as it evolves towards the target theory.

The *ontology* of the theory is the set of defined constants; this includes constant symbols denoting collections, individual entities, relations, and attribute values. The number of identifiable objects or concepts recognized by the conceptualization and defined in the target domain can be infinite; conse-

quently, the theory's universe of discourse may be infinite. While the set of relations recognized by the conceptualization and defined in the target theory is probably not infinite, at any given point in its evolution the set of relations defined in the domain theory need not be complete. Therefore, the ontology is extendable: new information can include references to symbols not previously defined (e.g., symbols denoting new collections, new entities, new relations, or new attribute values).

Information provided by the knowledge source is expressed in a *specification language* and is *interpreted* by the learner. Interpretation produces, at least, a translation of new information in the internal language of representation. [6] In addition to stipulated beliefs about the target domain, new information can include meta-information, such as the identity or type of the knowledge source or bounds on the computational resources that can be consumed by the learner while processing the new information.

A.1.4 The task source

The task source provides requests to the knowledge-based system. Each request and response pair constitutes one problem, or *task instance*; collections of problems sharing general specifications constitute problem types, or *application tasks*. Multiple task sources can be interacting with the system during problem solving.

As with knowledge sources, a conceptualization of the target domain is associated with each task source. Each application task is conceived within

[6] A special case of this learning situation occurs when the information is expressed by the knowledge source in the internal representation language (e.g., when the learning system itself is the knowledge source) and, thus, requires no interpretation. This simplifying special case has been called *the same-representation trick* [Die82, pages 368–369].

(i.e., in terms of the concepts recognized by) the task source's conceptualization. Task-source conceptualizations also change over time. It is a fundamental requirement of the representation effort that the conceptualizations of both the task sources and the knowledge sources are sufficiently similar to permit the existence of a single theory (e.g., the target theory) capable of supporting correct responses to the requests. However, during learning, the conceptualizations of the task sources are often not known and so must be assumed. A notorious source of failure by knowledge-based systems occurs when such assumptions are violated.

For an information system, the task sources include the system's user community; for a robot, this may also include its physical environment. Knowledge sources can act as task sources, and vice versa: teachers and textbooks pose questions about presented material, and employers train their employees in preparation for subsequent work assignments.

The possible requests provided to the system by a task source can include inquiries about the truth of any aspect of that task source's conceptualization. Therefore, the scope of the tasks can include querying the truth of any proposition that is expressible in the representation language and that involves terms recognized by any conceivable conceptualization of the target domain.

The *criterial task* is the set of requests used to evaluate learning. Traditionally, learning is appraised by changes in the system's performance on the criterial task [Bra79, pages 3–11]. However, this tradition is warranted only to the extent that the criterial task is representative of the application tasks actually encountered.

Commitments of the knowledge sources and learning system to as-

sumptions that constrain the application tasks are a very important aspect of every learning environment. As Chapter 1 discusses, traditional approaches to machine learning assume the application is completely (although perhaps implicitly) known by both the learning system and the knowledge source. However, learning often occurs when the learner has no expectations about the application tasks and when information presented by a knowledge source reflects only very general intuitions about its application. In general, it is not warranted to assume the learner and the knowledge sources will precisely predict subsequent application tasks.

Learning without strong preconceptions about the eventual use of acquired knowledge offers the important advantage of acquiring *foundational* (i.e., multi-purpose or task-independent) *knowledge*. Knowledge bases comprising foundational knowledge are of interest to engineers of knowledge-based systems for several reasons:

1. Building separate knowledge bases for every pairing of domain and task (e.g., separate knowledge bases for diagnosing malfunctions, for design, for education, for marketing) is not practical because so much of the domain knowledge required for one task would also be required for other tasks [PLM+88]. Acquiring the knowledge for each of the many individual task-specific theories within a common domain would duplicate, and therefore waste, effort. More significantly, the effort to maintain (e.g., debug and extend) the separate knowledge bases would significantly magnify the cost, and divergence among the many separate bodies of knowledge would seem inevitable. This suggests that maintaining a single body of foundational knowledge is eminently preferable to maintaining a large set of task-specific knowledge bases.

2. Very few theories are stable: they are neither complete nor is their change monotonic. Virtually no traditional knowledge sources (e.g., humans, books, databases, etc.) can boast complete knowledge in any significant or natural domain. It seems reasonable, therefore, to consider many domains as, in principal, infinite, and to expect that any particular knowledge base will have gaps. Furthermore, our knowledge of most domains fluctuates: new knowledge is discovered as old beliefs are refuted; new theories are proposed as old theories are rejected; new paradigms are adopted as old ones are discarded [Kuh70]. Ongoing evolution is inescapable for a knowledge base.

3. Very few application tasks are stable enough and clearly understood enough to warrant precise predictions for the use of a knowledge base by its designers. *Task drift* is evolution in the set of tasks a system is expected to perform; task drift has several causes:

   (a) Changes in domain knowledge affect domain tasks. For example, the introduction of a new treatment, the discovery of a new side effect of an existing treatment, or the discovery of a new indicator (i.e., symptom) all change the task performed by a medical diagnostic knowledge-based system (e.g., MYCIN). Similarly, new communication technology changes the tasks performed by political or marketing advising systems, and new policies in law and law enforcement change the tasks performed by legal advising systems. Many problem solving methods, such as heuristic classification, assume stable aspects of the domain, such as the set of possible solutions. Any change to these aspects causes task drift.

(b) Changes in the concerns of the users affect tasks. For example, the users of a medical diagnostic advising system might show increasing concern with the financial or ecological cost of diagnostic tests, or they might no longer be content with simply a diagnosis but require an explanation of the diagnosis as well.

(c) Changes in the populations of users affect tasks. For example, an investment consulting system developed for professional investors might fail miserably for less sophisticated investors who do not adequately understand the risk and assumptions underlying various investment strategies. Advising a professional and advising a novice are different tasks.

Task drift can have enormous repercussions for a knowledge-based system, depending on the extent to which that system is tailored to its expected uses. If a system's ontology cannot make the distinctions required to perform new application tasks, and if the system was designed under the assumption that these distinctions would not be necessary, then task drift can be devastating, even terminal. By avoiding a close coupling to narrow use expectations, foundational knowledge resists the potentially dramatic impact of task drift.

4. An advantage of declarative knowledge is that it is capable of being used to solve tasks that are unforeseen by the knowledge engineers who craft it [Nil91]. In other words, each unit of declarative knowledge is capable of being combined with other units in novel ways to infer conclusions or answers to queries that were not explicitly provided for by the system's designers. Encoding knowledge in logic achieves this quality at a micro-level: each atomic statement can interact with other statements,

according to the logic's rules of inference, to derive new and unanticipated statements. Foundational knowledge attempts to preserve this property at a broader, macro-level of the knowledge base by excluding tacit assumptions that are only appropriate for particular application tasks. By adapting to unanticipated uses, foundational knowledge is more versatile than conventional, task-specific expert systems.

Topics such as granularity of declarativeness (at what level units of knowledge can interact in novel, non-scripted, ways) or the efficacy of task-specific compiled knowledge versus that of task-independent declarative knowledge are of interest to research in the knowledge representation and knowledge-based systems communities and are not inherently germane to machine learning. However, questions such as how to acquire foundational knowledge, how to learn from foundational or task-independent information, and how to support incidental learning are germane to machine learning and form some of the core issues of research on knowledge integration.

A.1.5 Knowledge transitions

Knowledge engineers face a daunting task. Developing representations of complex domains that support automated reasoning is difficult because of the subtlety and complexity of the domains and because of obstacles such as the qualification problem. Furthermore, the number of possible alternative conceptualizations and representational schemes is staggering, and there are few constraints available to guide selections among the alternatives. These problems dominate the representation process and must be confronted each time a representation is formulated or reformulated.

Figure A.1 illustrates a decomposition of the representation problem

by identifying three important thresholds in the evolution of the representation: conceptualization, formalization, and application. Crossing each threshold constitutes part of the representation problem. It is important to consider what factors guide the transitions across these thresholds and how these transitions interact. Learning affects the transition from conceptualization to formalization, while problem-solving affects the transition from formalization to application; both are influenced heavily by the conceptualization. These thresholds thus provide a framework to reassess the nature of use expectations and their role in the representation problem.

Since it is true that only what is included in the conceptualization can be included in the formalization, and only what is included in the formalization can be included from the application, these three thresholds define a *hierarchy of inclusion*. However, the value of any knowledge-based system is inevitably judged by the quality of its application, that is, by how well it performs the tasks actually put to it, rather than by the qualities of the underlying formalization or conceptualization. Therefore, in practice, this hierarchy is turned upside down and becomes a *hierarchy of exclusion* that prescribes: include in the formalization (only) what will be required by the application; include in the conceptualization (e.g., the ontology) (only) what is required by the formalization. The exploitation of use expectations thus transcends machine learning: it dominates other (e.g., manual) approaches to developing the formalization, and it dominates the development of the conceptualization.

The practice of guiding the design and development of representations with use expectations is summarized by the *Use-directed Representation Principle*:

Representation tasks occur in response to (or in the context of) an

> agent attempting to perform a particular problem-solving task, and that problem-solving task identifies precisely what distinctions and capabilities are required in the resulting representation.

The use-directed representation (UR) principle is ubiquitous in the traditional pursuits of artificial intelligence. How-to books for constructing knowledge-based systems admonish designers to begin by defining precisely the requirements of the application task [BBB+83, BD81]. It is inherent to the most fundamental computational methodologies used in artificial intelligence, such as the notion of a state-space search. The expectation to begin with a problem specification is ingrained, a fundamental tenet of computer science: it is taught as basic programming methodology; it is assumed in the construction of computer applications, such as data-base systems; it is assumed for the application of formal methods for program analysis. Often the task specifications are assumed to be the responsibility of users and therefore outside the realm of computer science.

Methodologies for developing expert systems make extensive use of the UR principle. The performance task dictates both what knowledge to include and how to encode it. However, the tacit assumption that the encoded knowledge will be used only for performing the anticipated performance task appears to be a primary source of the brittleness that plague expert systems. If the performance task drifts at all, this assumption may be violated. Essential knowledge will be missing and the system's performance will be very poor or erratic [LF87]. It is no accident that projects like Botany [PLM+88] and Cyc [LG90] do not commit to a specific set of performance tasks, since doing so would have the advantage of guiding knowledge-base development and the disadvantage of introducing brittleness. The bias that guides knowledge-base

development is one and the same bias that introduces brittleness; they are two sides of the same use-expectations coin. The relationship between use expectations and brittleness is summarized by the *use-commitment brittleness conjecture*:

> Exploiting use expectations during the conceptualization or formalization of knowledge introduces brittleness.

The use-committment brittleness (UB) conjecture suggests that the single strongest bias for acquiring knowledge for contemporary knowledge-based systems, by either automated (e.g., machine learning) or manual means, is also the greatest source of brittleness that plague the resulting knowledge-based systems.

The convincing attribution of brittleness in contemporary knowledge-based systems to missing knowledge has been made elsewhere [LF87]. This conjecture goes further by identifying use expectations as the predominant cause of the missing knowledge, and, consequently, of the brittleness.

McCarthy hypothesizes that one important characteristic of common-sense reasoning is that the reasoner cannot predetermine what knowledge will be required by common sense tasks [McC89a]. As discussed earlier, this is also a property of foundational domain knowledge, which transcends common sense. Let the class of abilities that have this general property be called *flexible intelligence*; it is the complement of brittle, idiot-savant-like intelligence that achieves very high levels of performance but only within a very narrowly specified set of tasks. The consequences of the UB conjecture are profound: no method that commits to strong use expectations can attain representations sufficient for supporting flexible intelligence. This is why projects such as Botany and Cyc, which are enormously risky and expensive, are also enormously important.

The UB conjecture and its consequences could be dismissed by proponents of traditional system-building methodologies for a variety of reasons:

1. *The concern of computer science is restricted to developing solutions to specified computational problems; it is not concerned with developing specifications of computational problems. As a subfield of computer science, artificial intelligence shares this restriction. Thus, use expectations are inherent to computer science in general, and artificial intelligence in particular.* However, even if this restriction applies to most subfields of computer science, it cannot apply to artificial intelligence (and thus identifies an importance difference between artificial intelligence and mainstream computer science). For many of the essential problems studied within artificial intelligence (e.g., common sense reasoning, general learning and comprehension, teaching, natural language processing) there are no complete task specifications. [7] It is part of the enterprise of artificial intelligence to develop an understanding of these tasks so that they can be specified computationally; such tasks specifications would be considered major results.

2. *In the absence of precise use specifications, how can we as artificial intelligence researchers and practitioners evaluate our computational artifacts?* This imposing concern pales before the reality and the difficulty of evaluating human knowledge. Should the difficulty of evaluating human

[7] There are partial task descriptions. For example, each formal learning task represents the hypothesis that any method that performs the task constitutes learning. Therefore, each such task hypothesizes one sufficient condition of learning. However, traditional machine learning tasks are nowhere close to covering all the behaviors that could generally be called learning; so much of the task of learning, what it means to learn, remains unspecified.

learning preclude our attempts to effect it? If evaluating human knowledge is so complex and imperfect, should we expect simple and precise methods to evaluate general artificial intelligence? To avoid work on the essential problems just because the methodology is not worked out is no solution. Research on appropriate methodology, such as methods of evaluation, must proceed in tandem with, not restrict, research on the task specifications and computational methods to perform those tasks; developing such methodologies should be considered major results.

3. *There are many tasks that are well specified and that would be useful to automate; restrict ourselves to these tasks.* Essentially, this position accepts brittleness as inherent, and thus limits artificial intelligence to idiot-savant-like behavior.

4. *There are no other alternatives; use expectations are the only source for guiding representation tasks.* This is simply not true, as evidenced by the volumes of textbooks and other knowledge sources which commit to conceptualizations and specify bodies of archival knowledge but do not commit to strong use expectations.

Research in artificial intelligence should re-evaluate its reliance on UR. In some (perhaps many) situations, representation tasks could be guided by other factors, such as the conventional wisdom about each domain as reflected in the domains' existing archival knowledge.

Use expectations should not be entirely eliminated from approaches to knowledge representation. In some sitautions they are warranted; in others they may be unavoidable. In those cases, a *least commitment* approach to their use should be adopted. However, for many representation tasks, their

use will be speculative and suspect. The only time their validity is unconditionally guaranteed is during problem solving, when a particular use is directly availablle. Delaying their use past conceptualization and formalization prevents use-based bias from shaping these thresholds and, consequently, restricts the brittleness woven into the representations at each level. This is a major motivation for acquiring foundational knowledge, and for studying knowledge integration, which does not commit to fixed or narrow use expectations, as an approach to machine learning.

A.1.6 Component interactions

The compositional model in Figure A.1 identifies four roles and six types of transactions among them. The possible sequences of transactions among the components define the types of interactions that are permitted. Every sequence denoting a completed learning event will conclude with transaction type *information*. Different sequences of transactions exemplify different assumptions about the context in which learning occurs:

1. The sequence [*request, question, information*] illustrates learning during problem solving.

2. The sequence [*question, information*] illustrates intentional learning.

3. The sequence [*information*] illustrates incidental learning.

Furthermore, it is possible for more than one role to be assumed by a single entity:

1. When the learner is also the knowledge source *introspective learning* occurs: learning occurs when new knowledge is discovered from existing knowledge (e.g., AM [Len76]).

2. When the task source is also the knowledge source, the learner is called a *learning apprentice*: learning occurs when instruction on how to solve a particular problem follows the learner's failure to solve that problem (e.g., PROTOS [PBH90]).

A.1.7 Discussion

An important aspect of this model of a learning environment is the conceptualization of the target domain from which the knowledge source selects information to present to the user. The conceptualization is essential to learning that facilitates the construction of knowledge-based systems. One of the fundamental difficulties in building knowledge-based systems is determining sufficient (e.g., consistent and complete with respect to the application tasks) conceptualizations, and the consequences of an inadequate conceptualization can be extremely traumatic. Consequently, many approaches to knowledge acquisition are dedicated to assisting knowledge engineers in defining and then developing conceptualizations [Boo85]. To assume that the initial conceptualization is sufficient and remains constant is to ignore one of the fundamental problems of constructing knowledge-based systems.

A.2 What is learning?

In order to formulate a formal specification of a learning task appropriate to the described learning environment, it is convenient to first characterize informally what is meant by learning, that is, to assess what kinds of changes to a learning system are indicated by the notion of learning.

There has been considerable debate among machine learning researchers about proposed definitions of learning. There are two predominant traditions:

one casts learning as improving task performance; the other casts it as acquiring knowledge.

A.2.1 Learning as enhancing task performance

Learning commonly occurs during problem solving. In fact, some researchers claim that learning occurs only in the context of performing a task. For example, Langley and Simon (1981) define learning as:

> any process that modifies a system so as to improve, more or less irreversibly, its subsequent performance of the same task or tasks drawn from the same population.

This definition assumes that each learning event is paired with a task (or population of tasks); the system's performance at that particular task improves as a result of learning. Any modification to a system that improves its performance at the task constitutes learning, and all learning improves the system's performance at some task.

Although it is clear that problem solving does have an important role in learning, this task-oriented definition advocated by Langley and Simon is unsatisfactory. The suggestion that any process that causes a system's performance to improve constitutes learning is too permissive and includes processes that defy our common beliefs about what learning should be [Sco83]. For example, a new leather shoe improves its performance at providing comfortable and unencumbering support and protection by selectively stretching to conform to the shape of the wearer's foot. Howver, this clearly violates our common understanding of what learning is. Therefore, improving task performance is not a sufficient condition for learning.

This definition of learning is also too restrictive. It requires that learning improve a system's *subsequent performance at the same (population of) task(s).* The implication seems to be that learning occurs during (or in response to) the system's attempt to perform a particular task, and that what is learned is necessarily relevant to that task. Using the componential model, this constraint restricts learning to the transaction sequence [*request ... information*]: a particular request is associated a priori with each learning event. However, this restriction excludes many events that satisfy our common understanding of learning: simply acquiring knowledge that may be useful at some future task constitutes learning [SV83]. For example, while reading an L.M. Boyd article, a subject might be informed that the English word "hello" translates to "marhaba" in Turkish. If this information was not previously known (and is remembered), the subject has learned something new even though the subject certainly was not (consciously) performing a task relevant to this new information. Alternatively, consider the behavior of the student in Figure 1.1: learning occurs as the student comprehends the teacher. The student's learning does not primarily result in improved performance at comprehending the teacher; rather, it results in a greater knowledge of botany. Therefore, the context of performing an application task relevant to what is learned is not a necessary condition for learning.

Furthermore, by requiring *improved* subsequent performance, the definition is too restrictive. Learning can, in fact, be seen to decrease task performance, depending on the measures used to evaluate that performance [Bra79]. A subject is told that one cause of condition Y is precondition X. The subject is later queried for the single best reason causing each of a particular set of patients to have condition Y and responds, correctly, that the cause might be

precondition X. Later, the subject is told that preconditions A through W also cause condition Y. Subsequently, the subject is asked for the single best reason causing each of a new set of patients to have condition Y, and the subject declines to speculate the possible cause. In fact, unbeknownst to the subject, the conditional probability of precondition X being the cause of condition Y is 90%. However, by learning an extension to the set of possible causes, the success rate for the subject declines dramatically. Alternatively, consider again the learning event presented in Figure 1.1. In response to new information, the student's knowledge of botany is in some ways enhanced, as evidenced by the prediction that other shoot organs also have cuticles, but in other ways it is corrupted, as evidenced by the prediction that leaves with cuticles starve. Whether or not learning improves performance depends entirely on the criterial task [Bra79].

Despite these objections, problem solving does have great significance for learning. First, a problem-solving context can guide learning processes. Second, the function of learning can be explained in terms of improved task performance. Third, changes in task performance are often the only proof that learning has occurred. Therefore, problem solving is very important to the process of learning and to our study of learning. However, a definition of learning should not so strongly couple learning and problem solving: it should not permit any process by which task performance is improved; it should not require improved subsequent performance; and, it should not restrict assessments of performance changes to the same task.

A.2.2 Learning as knowledge acquisition

In contrast to the task-oriented definition of learning advocated by Langley and Simon, other machine learning researchers have proposed knowledge-oriented definitions of learning:

1. Scott and Vogt define learning as: *the construction of an organized representation of experience* [SV83].

2. Michalski defines learning as: *constructing or modifying representations of what is being experienced* [Mic86].

3. Dietterich defines one important type of learning as: *the acquisition of knowledge* [Die86].

To simplify matters, this discussion considers only *conceptual learning* in which "representations of experience" can be interpreted as knowledge, as opposed to other possible phenomena (e.g., scars, changes in muscle mass, etc.) that are not naturally recognized as knowledge.

Each of these definitions identifies knowledge acquisition as the essential effect of learning. Specifically, each claims that the acquisition of any knowledge (i.e., affecting the representation of any experience) constitutes learning, (and the first two appear to claim that all learning involves the acquisition of knowledge). In terms of the componential model, this position admits as learning the simple transaction [*information*], requiring only that the information endows the system with new beliefs. Although it is clear that knowledge acquisition does have an essential role in learning, these knowledge-oriented definitions are also unsatisfactory because they conflate knowing with learning.

Devices that simply react to sensed aspects of their environment can be said to have, and therefore to acquire, knowledge of their environment. For example, a thermostat senses the ambient temperature, and, depending on its target temperature setting, turns an air conditioner on or off. The thermostat acquires new knowledge each time it senses fluctuations in the ambient temperature; consequently, each such fluctuation results in an event that satisfies the knowledge-oriented definitions of learning. [8] Alternatively, consider an alarm clock: does it learn as each moment passes? When the current time equals the alarm setting the clock demonstrates its acquisition of this new belief by activating its alarm, but it hasn't learned. There is a difference between knowing and learning; consequently, there must also be a difference between knowledge acquisition and learning.

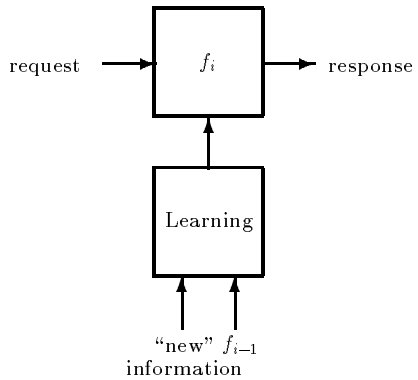
A.2.3 Learning as response-function evolution through knowledge acquisition

In contrast to the task-oriented definition of learning advocated by Langley and Simon, and the knowledge-oriented definitions of Scott, Vogt, Michalski, and Dietterich, the following definition of conceptual learning is proposed:

> Learning is any process by which a system acquires knowledge that changes its response function.

Here, knowledge denotes information, held to be true, in any form of representation, that can be used by the system to influence its response function. The

[8] The first two definitions are even more permissive. For example, simple sensing devices – such as thermometers, scales, cameras, audio recorders, etc. – are all capable of representing aspects of their experiences, and as such they all qualify as learning systems under the first two definitions.



The system's response function f evolves through learning: f_0, f_1, ..., f_{i-1}, f_i, f_{i+1}, ... Each learning event advances the response function to the next step in the progression.

Figure A.3: Learning as response-function evolution

response function is the system's behavior potential; it comprises the mapping from system inputs (i.e., requests) to outputs (i.e., responses).

This definition of learning resolves the problems noted with the task-oriented and knowledge-oriented definitions. It unifies the two by retaining the essential roles of problem solving (from the task-oriented view) and knowledge acquisition (from the knowledge-oriented view) while excluding notorious examples of non-learning, such as the leather shoe and thermostat examples.

As Figure A.3 illustrates, each learning episode results in a new response function: for at least one possible request, the new function produces a different response than the prior function. This distinguishes between simply acquiring knowledge (such as sensing) and learning. For example, the response function of a simple thermostat can be characterized by the formulae:

$$[(temperature \succ target) \Rightarrow state(AC\ On)]$$

$$[(temperature \preceq target) \Rightarrow state(AC\ Off)]$$

These formulae completely, although implicitly, specify a table of triples that associate the system's response (i.e., state of the air conditioner) with every possible input (i.e., the ambient temperature and the target temperature). Sensing the ambient temperature or adjusting the target temperature doesn't change this response function and, consequently, doesn't constitute learning. However, if the thermostat were also designed to date stamp and remember sensed temperatures, then the response function would be extended to include:

$$[recorded\text{-}temperatures(target\text{-}date,\ X) \Rightarrow display(X)]$$

This formula constrains the system's response function but does not completely specify it. The variable X is not an independent parameter; rather, it is defined by the relation *recorded-temperatures* (e.g., a set of date and tempreature pairs stored in the device's memory); each fragment of knowledge that identifies a new pair that satisfies this relation changes the system's response function. Consequently, sensing and recording a new temperature changes the response function and constitutes learning.

An essential condition of this definition is that acquired knowledge changes the system's behavior: its response to some possible request must be altered by acquired knowledge. The acquired knowledge constitutes what is learned; the change in behavior demonstrates that learning has occured. In order to influence behavior, knowledge must be accessible. For example, the complete and correct rules of chess entail how to play every possible game of chess, including the moves, if any, that lead only to wins for every possible chess situation. However, simply learning the rules of chess does not support playing perfect chess if the knowledge of the perfect moves are inaccessible (which must

be the case for any realizable system).

This definition permits learning in the absence of new external experiences (i.e., *introspective learning*). For example, learning may result from altering the representation of knowledge to transform inaccessible beliefs into accessible beliefs (i.e., improving the theory's compactness; see Section B.2.1). Learning actually does occur in this situation if the newly accessible beliefs cause a change in the system's response to some possible request.

According to this definition, each learning event acquires knowledge that changes the system's response function. When learning occurs, some task relevant to the acquired knowledge can, in principle, always be identified, since each possible response can be paired with a request that requires precisely that response. For example, learning the Turkish word for "hello" improves the learner's ability to communicate politely with Turks; it certainly supports responding to the query: *What is the Turkish word for "hello"?* In general, the acquisition of belief p can always be paired with the request: *Is it true that* p*?* This reflects the important relation between learning and problem solving: learning always changes the system's performance at *some* set of tasks. [9] However, the affected application tasks may be completely independent of the learner's activities during the learning event, and the learner may be oblivious to them when learning occurs.

A.2.4 Teaching vs. learning

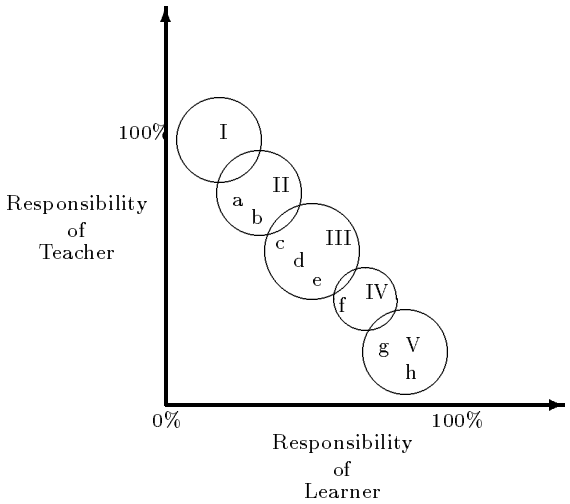
In every representation task, there are two essential roles: that of the learner and that of the teacher. The responsibility of effecting the represen-

[9] In fact, learning necessarily improves the system's performance at *some* set of tasks.

tation is distributed between these two roles. For example, the teaching role may be responsible for carefully selecting the training and presenting it in a particular sequence. The learner may rely on the teacher to present only training having some particular form or content. The teacher may be expected to provide worked-out solutions to tasks (e.g., classified training instances) or to ensure that there is no noise in the provided training. The teacher may be required to answer questions, or to ratify proposed new beliefs suggested by the learner.

Each representation task will necessarily distribute the responsibility for effecting the new representation between the teaching role and the learning role. When the teacher adopts some measure of responsibility (R), the remainder (1–R) is apportioned to the learner. For example, during introspective learning, the responsibility apportioned to the learner's role may be complete; during programming, the responsibility assumed by the teacher's role may be complete. Learning occurs only to the extent that responsibility for effecting the resulting representation is assumed by the agent in the learning role. Consequently, pure programming, which certainly can change the response function of a system, does not constitute learning.

As Figure A.4 illustrates, most machine-learning methods commit to some particular distribution of responsibility for effecting the new representation while most machine-learning tasks commit to some range of distributions. Ideally, learning tasks and methods should not commit to narrow ranges of the distribution of responsibility during learning. By being flexible on the apportionment of responsibility, learning methods remain opportunistic, ready to exploit whatever abilities both the teacher and the learner bring to each particular learning situation. Learning tasks should remain flexible to permit this



I. Programming
II. Automated Knowledge Acquisition
(a) THEIRESIAS
(b) PROTOS
III. Supervised Learning
(c) ID3
(d) AQ11
(e) LEX
IV. Unsupervised Learning
(f) COBWEB
V. Discovery
(g) BACON
(h) AM

The relative responsibility for effecting a new representation apportioned to machine-learning systems (in their roles as learners) and users (in their role as teachers).

Figure A.4: Teacher Responsibility vs Learner Responsibility

opportunism in learning methods that perform them.

A.3 Learning as test incorporation

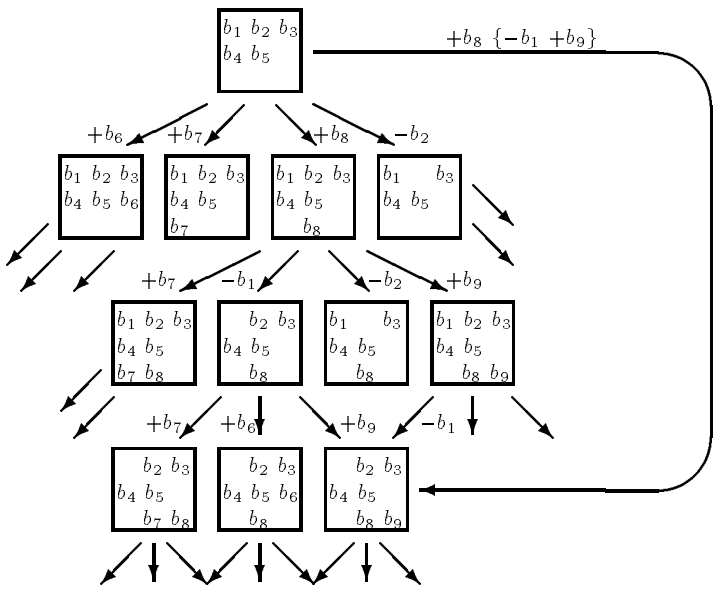
Constructing a knowledge base can be viewed as a search through a state space: each state is a candidate knowledge base, and each operator that moves between states is a knowledge-base modification (e.g., an extension, retraction, or revision). The contents of new information is equated with one

operator that can be applied to the current state (i.e., the state denoting the current knowledge base). Thus, the enterprise of machine learning can be formally defined as automating, to some extent, search in this state space, and acquiring knowledge beyond the explicit content of new information is a form of *test incorporation* [Tap80].

As Figure A.5 illustrates, acquiring knowledge beyond that explicitly contained in the new information causes the application of a single operator to expand into the application of a sequence of operators. All states that do not include the results of applying the entire sequence of operators are removed from the state space, thereby effectively reducing the size of the state space. If the size of the state space is n and the average number of operators applied during each state change is m (including operators denoting both the training and the additional learned knowledge), then the effective size of the state space is n/m, and the reduction in the size of the state space, as well as the gain from learning, is $n - n/m$.

This framework permits formally defining significant and trivial forms of machine learning. *Significant learning* acquires knowledge beyond the explicit content of training, thereby compressing the knowledge-base state space; it is the obligation of every non-trivial learning method. In contrast, *trivial learning* acquires only what is explicitly presented; this includes the most banal forms of rote learning, such as the behavior of a text editing program (e.g., emacs).

The extent to which the search space is reduced varies directly with the amount of additional knowledge that can be gleaned from the training, which in turn varies with the number of different types of interaction between new and prior knowledge that a learning method is sensitive to and can exploit



Machine learning, (i.e., knowledge-base development) viewed as a state space search: states are alternative knowledge bases; operators that move between states are knowledge-base modifications. In some initial state, the current knowledge base includes beliefs b_1 through b_5. Several operators can be applied to this state, one of which is adding belief b_8. A learning method, reacting to interactions detected between new and prior knowledge, indicates that adding b_8 also requires retracting b_1 and adding b_9, thereby collapsing a path of operators (i.e., a set of knowledge-base modifications) into a single step (the arc on the right-hand side). Learning has reduced the size of the state space by effectively removing many of the states; in the example, the state $\{b_1\ b_2\ b_3\ b_4\ b_5\ b_8\}$ is removed from the state space. Without learning, no such interaction is detected, and the complete space of states and of possible operator sequences must be searched.

Figure A.5: Learning as Test Incorporation

to acquire additional knowledge. By actively investigating diverse ways that new and prior knowledge can interact, learning methods can more effectively guide knowledge-base development. By acquiring knowledge beyond the explicit content of the new information, they enhance the knowledge base and compress the space of alternative knowledge-base extensions.

Appendix B

The Learning Task of Knowledge Integration

One of the goals of exploratory research in Artificial Intelligence is to develop precise, computational specifications of new tasks. This appendix describes knowledge integration as a machine learning task. The first section proposes a formal specification for knowledge integration as an information processing task. The second section discusses some general criteria to guide learning methods that perform this task.

B.1 A formal definition of knowledge integration

The formal specification of a new task is useful in two ways. First, it defines precisely the scope of a body of research. The definition facilitates determining whether particular problems are or are not instances of the task and whether different methods are or are not applicable to the task. Second, it constitutes a hypothesis that the formal definition adequately characterizes the phenomenon being studied. Subsequent research may reveal inadequacies, such as failures in coverage, and may propose refinements to the task definition.

For the purpose of constructing a knowledge base, the learning task of knowledge integration is defined as the information processing task presented in Figure B.1.

The admissibility predicate defines an invariant condition that must

Given:
1) *Representation Language:* a language (RL) to describe beliefs, and an inference procedure ($\vdash$).
2) *Knowledge Base:* A set of beliefs (Δ) expressed in RL.
3) *Training:* A set of beliefs (Θ) expressed in RL
4) *Admissibility Criteria:* A predicate (Γ) over Δ, Θ, $\vdash$, and the resulting knowledge base, Δ', specifying criteria that beliefs of Δ' must satisfy (e.g., consistency requirements).

Determine:
Revised Knowledge Base: a set of beliefs (Δ') that satisfies the admissibility criteria (i.e., $\Gamma(\Delta\ \Theta \vdash \Delta')$ is true).

Figure B.1: The Knowledge Integration Task

be preserved across knowledge-base modifications. As the knowledge base is modified (e.g., extended with new information) the learning methods are obligated to ensure this condition is satisfied by the resulting knowledge base. Satisfying the admissibility predicate may require learning methods to affect further changes to new or prior knowledge. These further changes constitute learning beyond the explicit contents of the new information (as discussed in Chapter 1). Thus, the admissibility predicate characterizes the *learning goals*; it determines the knowledge changes that learning must affect.

Machine learning can be viewed as achieving test incorporation in the *generate and test* search paradigm (Section A.3), and this view makes clear the role of the admissibility criteria. In this paradigm, knowledge-base construction is cast as a state-space search problem: the current knowledge base corresponds to the current state, and each possible modification of the knowledge base corresponds to a search operator that maps the current state into a new state. The admissibility predicate specifies necessary conditions that each new state (i.e., each candidate knowledge base) must satisfy. The learning methods incorporate into the process of generating a new state the test that the admissibility criteria is satisfied by that state. Consequently, only states that satisfy the admissibility criteria are generated. States that fail the

admissibility criteria are never generated and so are tacitly removed from the search space. Thus, machine learning is a form of test incorporation: it facilitates knowledge-base construction by enforcing the admissibility condition on all generated knowledge bases, thus reducing the space of candidate knowledge bases.

The admissibility predicate defines the learning goals. Making this an explicit parameter of a general learning task has two advantages. First, it avoids tacit commitments to particular learning goals: the admissibility predicate specifies precisely the obligation of methods that perform the task. Second, it provides a unifying framework in which the set of learning goals becomes an accessible, manipulatable, independent variable. This framework defines a space of learning tasks and a way of comparing existing machine learning tasks. The task description presented in Figure B.1 serves as a schema for more specific learning tasks since each distinct admissibility predicate defines a distinct learning task.

Traditional machine learning tasks may assume a narrowly-focused admissibility predicate because they constrain the knowledge to be learned. Consequently, the learning methods that perform these tasks can be equally narrowly-focused. They define strategies for only determining *how to learn* rather than also determining *what to learn*. A less focused admissibility predicate, one that does not tightly restrict the learning goals, requires learning methods to include strategies for both.

Increasing the scope of the admissibility predicate increases the difficulty of establishing it. For example, maintaining complete consistency among all statements in a knowledge base (the traditional goal of the machine learning task of *knowledge revision*) is a stronger learning goal than maintaining con-

sistency among some subset of the knowledge base, such as concept definitions and classified concept instances (the traditional goal of the machine learning task of *concept acquisition*). Enforcing the stonger learning goal is more difficult since any learning method that fails to satisfy the more specific learning goal also fails the general goal. Similarly, satisfying a given learning goal (e.g., maintaining the consistency of the knowledge base) is more difficult when the representation language is more expressive (e.g., first-order logic vs. propositional logic). In general, increasing the scope of the learning task increases the difficulty of performing it.

The task specification of Figure B.1 differs from traditional machine learning tasks by not committing to learning goals dedicated to a specific application task. The next section discusses learning goals that transcend narrow use expectations.

B.2 General learning goals

In general, it cannot be assumed that either the learner or the knowledge source (e.g., a teacher, a textbook) can predict the precise application tasks that the learner will eventually encounter (see Section A.1). This situation raises a fundamental question: in the absence of strong use expectations, why should the learner expend any cognitive energy trying to learn from new information? The learning behavior could simply add new information to the existing knowledge base. Such behavior would constitute rote learning in the extreme: no attention is given to the consequences of the new information, and there is no learning gain (as defined in Appendix A.3). However, the learning behavior depicted in the scenario of Figure 1.1 is anything but rote: the learner actively considers the consequences of the new information to find in-

consistencies with prior knowledge and suggest generalizations. In the absence of a specified application task, what can serve as learning goals to stimulate the learner to exhibit this behavior? What conditions should be enforced by the admissibility criteria? What principles trigger and guide learning?

B.2.1 Generic learning goals

In the absence of particular application tasks the learner relies on *generic learning goals:* consistency, completeness, economy, and conviction. Each of these goals promotes some aspect of the system's competence. The first two goals address the issue of correctness, minimizing both false positive and false negative beliefs about the target domain. The last two address meta issues, minimizing the cost of computing responses (e.g., establishing implicit beliefs) and maximizing confidence in the content of responses.

Consistency: The goal of consistency is to minimize false positive beliefs. These occur when the system establishes propositions which are not true in (the conceptualization of) the target domain. Promoting internal consistency involves minimizing the extent to which the learner's beliefs are contradictory. Promoting external consistency, or correspondence, involves maximizing the fidelity by which the learner's beliefs accurately represent the real world target domain. Improving external consistency bolsters the user's commitment to the system. The goal of consistency incites the learner to detect and resolve inconsistencies. Adjudicating among incompatible beliefs sometimes requires additional knowledge.

Completeness: The goal of completeness is to minimize false negative beliefs. These occur when the learner fails to establish a criterial proposition that is

true in the target domain. A proposition is *criterial* to a request when the content of the system's response is affected by whether or not the proposition is established. Promoting completeness involves identifying and filling gaps in the learner's knowledge, such as:

1. missing terms: collections or individuals in the target domain that have no corresponding object in the universe of discourse and consequently cannot be denoted in the theory.

2. missing predicates: relations, such as *isa, color, age, location,* etc., that are true of objects in the target domain but have no corresponding representation in the theory.

3. missing facts: ground beliefs, such as $isa(MousePad_6\ MousePad)$, $color(MousePad_6\ RoyalBlue)$, $age(Fred\ Years\ 29)$, $location(Fred\ AustinTX)$, that are true in the target domain but are not beliefs in the theory. [1]

4. missing rules: principles that are useful to describe sets of facts in the target domain but are not explicitly represented as beliefs in the theory, such as $[\forall\ (xyz)\ contains(x\ y)\ \&\ contains(y\ z) \Rightarrow\ contains(x\ z)]$.

Missing terms and relations restrict the set of expressible propositions and so typically entail large numbers of missing beliefs. Propositions that are expressible and true in the target domain but cannot be established also constitute missing beliefs. One reason such propositions cannot be established is they are not in the inferential closure of the theory because of other missing beliefs (e.g.,

[1] By convention, throughout this document, a term name comprising a concept prefix and an integer subscript denotes a particular instance of the concept; e.g., $MousePad_6$ denotes a particular mouse pad and $Plant_8$ denotes a particular plant.

rules and facts that support their derivation). Another reason is that they are inaccessible: their derivation cannot be completed (e.g., due to limitations on the computational resources consumed during inference).

Economy: The goal of economy is to organize knowledge in such a way as to minimize cost, where cost is the consumption of computational resources (e.g., memory requirements, response time, CPU cycles, page faults, etc.). A primary concern of economy is *compactness* – the accessibility of implicit beliefs. Economy is promoted when the accessibility of useful implicit beliefs is increased by compiling the results of inference (e.g., proofs, derivations) into explicit knowledge fragments (e.g., theorems, macro operators). Chunking [And83, RN86] and explanation-based learning [MKKC86, DM86] are techniques for increasing the accessibility of implicit knowledge. Compilation can (although not necessarily) improve the system's response time [Min88] and completeness.

Conviction: The goal of conviction is to maximize commitment to the system's beliefs. Even if the system's response to a given request is correct, in the sense that the representation law (Section A.1.3) holds for the contents of the request and response, the validity of the response might not be apparent to the user. Internal conviction is the extent to which the system attributes truth to its beliefs (e.g., a proposition may be held to be a monotonic or a nonmonotonic belief, that is, a default assumption, depending on its support). External conviction is the extent to which users are committed to the system's responses. Conviction is improved by justifying beliefs, that is, by determining how a belief follows from a set of other beliefs. Improving the system's explanatory competence – the ability to explain why a belief is held – bolsters the user's commitment to the system [Swa83, Mur90].

Discussion: These four learning goals are interrelated. In a system that permits nonmonotonic inference, consistency depends on completeness. For example, with the default rule [$\forall$ (x) $p(x)$ *unless* $q(x) \Rightarrow r(x)$] [2] the system might fail to conclude the true proposition $q(Thing_1)$ and conclude the false proposition $r(Thing_1)$. Thus, a false negative belief begets a false positive belief.

In a system that includes computational resource bounds, completeness depends on compactness. For example, let R denote the set of criterial beliefs computed with infinite computational resources in response to an arbitrary request, let R_b denote the set of criterial beliefs computed under the computational resource bound b in response to the same request (thus, every proposition in R_b that has a monotonic derivation is included in R), and let p denote an arbitrary element of R. Whether p is included in R_b is determined, in part, by the compactness of p's derivation. If the computational resources required to establish p (e.g., the the number of variable bindings that must be attempted before p is established) are too high, then p is omitted from R_b. If p is true in the domain then its omission from R_b constitutes a false negative belief.

In a system that includes derivation compilations, compactness depends on consistency. A compilation is only as valid as the rules in the derivation that it summarizes. Compilations over incorrect rules will often reflect the fallacies of those rules. As incorrect beliefs are detected and revised, compilations that assumed those beliefs are no longer valid, and the benefits for compaction afforded by those compilations are lost. Therefore, an evolving

[2] The operator *unless*(p) is satisfied when either $\neg p$ is established or p cannot be established; that is, it permits negation as failure when trying to establish $\neg p$ in this particular antecedent.

system with poor consistency cannot reliably improve its compactness.

These generic learning goals can, in fact, be seen alternatively as generic teaching goals (Appendix A.2.4). Each is simply a generic goal of any representation. These goals are equally relevant whether a knowledge base is being developed manually or with machine learning tools. In the absence of a specific application task, the generic goals stimulate learning behavior beyond simple rote learning.

B.2.2 Other learning goals

Generic learning goals and learning goals afforded by a specific application task are two extremes. In between them is a spectrum of learning goals associated with various aspects of each learning situation.

Domain-specific goals: Learners often acquire general learning goals for particular domains. For example, students are taught to value some types of knowledge within a specific domain over other types of knowledge. In most academic domains, general principles that apply to a multitude of particular situations are valued. The biological sciences stress the importance of survival of both individuals and species; the relative significance of anatomy, physiology, and reproduction are due partially to the extent to which they are essential to survival.

Students are taught to value some aspects of a domain over others because mentors (e.g., teachers and textbook authors) cannot provide complete accounts of the domain and so must choose which specific material will be presented. The more essential, fundamental, and influential a property is within a domain, the more value is placed on knowledge of the property. The choice

of material and the emphasis placed on it by mentors introduce and reinforce biases, causing students to value some aspects of the domain over others. These biases support the domain-specific learning goals:

- acquire valued domain knowledge
- determine how new information affects (e.g., explains) valued domain knowledge

These learning goals develop from choices made by mentors who do not have complete knowledge of the specific tasks each learner will encounter that might require knowledge of the domain. Instead, the choices rely on general expectations of the types of tasks for which knowledge of the domain is most useful.

Source-specific goals: Learners often acquire general learning goals for particular sources of new information. For example, it is reasonable to expect a student to respond differently to information disseminated by a teacher, a textbook, a parent, a priest, a friend, an expert in the domain, a non-fictional book, movie or television show, or a fictional book, movie or television show. Different knowledge sources give rise to differing expectations about the types of interactions with the source that are possible, about the veracity of the information, and about subsequent application tasks. These different expectations can cause the student to exhibit different learning behaviors.

Idiosyncratic and egocentric goals: Individual learning goals arise out of an agent's innate or acquired interests, or what Kahneman calls *enduring dispositions* [Kah73]. Most people are naturally interested in determining the consequences of new information about topics they care about, such as their

health, occupation, employer, hobbies, family members, friends, enemies, and investments. New information that references such topics may undergo much greater scrutiny than information that mentions nothing of special interest to the learner.

Spontaneous use expectations: While a learner may not have predetermined expectations about the use of acquired knowledge, new information may spontaneously trigger use expectations. These expectations have many sources:

1. One source of spontaneous use expectations is the learner's outstanding tasks. For example, a learner is informed by coworkers about traffic problems ensuing from some new construction on a nearby freeway. The learner recognizes that this information is directly applicable to the twice-daily task of planning a route between work and home. The learner may respond by collecting additional information to better perform this application, such as the precise location and expected duration of the construction, the condition of alternative roads, etc.

2. A second source of spontaneous use expectations is scripted or cliche uses of new information. For example, a learner is informed that an acquaintance had a particularly wonderful meal at a local restaurant. For most restaurant patrons such information is typically used for planning dining outings. The ensuing learning behavior may be guided by the learner's attempt to determine under what circumstances this new restaurant would be a better choice than other restaurants. Consequently, the learner may respond by seeking additional information about the new restaurant, such as the menu, typical dining cost, location, ambiance, suitable dress, need for reservations, etc.

3. A third source of spontaneous use expectations is tasks tacitly suggested by the new information. For example, a learner may be informed that an acquaintance is looking for a job. Even though the learner has no immediate plans to make use of this information, it does suggest a possible future use: that of pairing the acquaintance with appropriate job openings. Consequently, the learner may respond by soliciting additional information about the acquaintance (such as the acquaintance's interests and qualifications) to better determine what type of job openings might be appropriate. The new information suggests the task of pairing the acquaintance with job openings, and learning behavior includes attaining additional information required to perform that task.

In the first case above, new information is directly applicable to a specific outstanding task; in the second case, it is applicable to a standard task that the learner is likely to encounter in the future; and in the third case, the new information itself suggests a task that the learner may decide, if opportunities present themselves, to perform. In each case, learning is guided by use expectations. However, the commitment to these use expectations was not predetermined but rises spontaneously as the new information is encountered.

B.2.3 Discussion

General learning goals identify biases that guide learning in the absence of predetermined use expectations. They illustrate many alternative ways in which learning can be an active, goal-driven process. However, the goals that guide specific learning episodes cannot be detached from the content of the new information and cannot be assumed in advance; it is the content of the new information that suggests what learning goals are appropriate. Consequently,

narrow learning goals cannot be scripted into the learning task specification without also constraining the content of the new information in the task specification, which, in turn, constrains the applicability of the learning methods that perform the task.

Appendix C

Interpreting Semantic Networks

During interpretation, KI translates information, which is expressed in the semantic network formalism of the input language, into the knowledge base's internal representation, which herein is characterized as axioms in first-order logic. [1] Figure C.1 presents a formal specification of interpretation as an information-processing task.

Interpretation involves parsing training graphs (encoded in a nested-list notation) into tuples, translating the tuples into axioms in the representation language, and finally adapting these axioms, as well as relevant existing knowledge in the knowledge base, to promote their mutually compatibility. Traditionally, interpretation tasks confront the issue of *representational adequacy:* the internal representation language must be sufficiently expressive to encode the information provided [McC58]. When the input language is very expressive (e.g., natural language), interpretation becomes very difficult. Ironically, KI's interpretation task is difficult for a complimentary reason: the representation language is much more expressive than the input language; consequently, ambiguities within input expressions must be resolved heuristically. This appendix discusses how KI interprets expressions in the input language. The first section

[1] The knowledge base's internal representation is actually CycL [LG90]. However, for convenience, internal expressions are represented as sentences in logic.

Given: IL, an input language
TL, a target language
TR, a set of translation rules that map expressions from the input language into expressions represented in the target language
IE, an expression represented in the input language
KB, a set of expressions represented in the target language
AC, admissibility criteria, a predicate on sets of beliefs expressed in TR
Find: KB', a nonmonotonic extension of KB that includes a translation of IE into TL and that satisfies AC.

Figure C.1: The task of interpretation

describes the graphical input language, and the following sections discuss the three subtasks of parsing, translating, and adapting.

C.1 The input language: specifying training with semantic networks

Knowledge is presented to KI as semantic networks encoded as nested lists. For example, the initial training for the cuticle scenario is:

```
(LeafEpidermis (coveringPart (LeafCuticle (composedOf (Cutin)))))
```

The supplemental training (i.e., the selected revision) is:

```
(& (LeafEpidermis (portal (Stomata)))
   (¬ (LeafCuticle (covers (Stomata)))))
```

Figure C.2 presents the grammar of the input language. Classes of input expressions for which special handling is provided include logical networks and disablement subgraphs.

network	::=	<nonterminal> | <logical-network>
graph	::=	<terminal> | <nonterminal>
nonterminal	::=	(<node> <subgraph>$^+$)
terminal	::=	(<node>)
subgraph	::=	(<arc> <graph>$^+$)
logical-network	::=	<implication-network> | <conjunction-network> | <negation-network>
implication-network	::=	(⇒ <network> <network>)
conjunction-network	::=	(& <network>$^+$)
negation-network	::=	(¬ <network>)

Arcs are symbols denoting binary predicates, and nodes are symbols denoting constants (e.g., collections, individuals, attribute values, or predicates) or literal data permitted in the knowledge base (e.g., strings, numbers, ...).

Figure C.2: The input-language grammar

C.1.1 Logical networks

Conjunction networks allow multiple networks to be entered as training for a single learning event. For example:

```
(&  (Seed  (contains  (Embryo)))
    (Plant  (hasPart  (Root)  (Stem))))
```

denotes: *Seeds contain embryos, and plants have as parts roots and stems.*

KI assumes that knowledge-base assertions resulting from interpretation will be asserted with a positive truth value (i.e., denoting they are true of the domain). Negation networks override this assumption and allow the specifyication of beliefs are thought to be untrue. Each assertion produced by interpreting a network within the scope of the negation symbol ¬ is asserted with an inverted truth value. For example the network:

```
(¬  (Seed  (developmentalStageOf  (Plant)))
    (¬  (Seed  (contains  (PlantEmbryo  (developmentalStageOf  (Plant)))))
        (¬  (PlantEmbryo  (hasPart  (Fruit))))))
```

denotes: *While seeds are not developmental stages of plants, they contain embryos that are developmental stages of plants, and plant embryos do not have fruit.*

Implication networks enable conditional beliefs (e.g., rules) to be entered as training. For example the network:

```
(⇒  (Plant  (physicalPart  (Flower)))
    (Plant  (physicalPart  (Stem))))
```

denotes: *Plants that have flowers also have stems.*

C.1.2 Disablement subgraphs

It is sometimes convenient to specify that, in a particular context, an object does not exist or a process does not occur, when, in a broader context, that object or process is expected. For example, almost all plants are photosynthetic; however, a few are not. It is therefore reasonable to specify in the knowledge base that all plants, by default, engage in photosynthesis, and then override this expectation for those plants that are not photosynthetic. A very convenient way to specify this is:

```
(Plant  (performs  (Photosynthesis))
        (specs  (Mushroom  (performs  (Photosynthesis  (status  (Disabled)))))))
```

However, the intent of the subgraph (Photosynthesis (status (Disabled))) is certainly not to assert that photosynthesis, in general (i.e., for all plants), fails to occur. Therefore, KI adopts the convention that all such specifications of disablement (or enablement) are inherently context sensitive and includes special translation rules that are required to support this convention. For example, the above network is taken to denote that plants in general engage in photosynthesis, but mushrooms, in particular, do not. Note that this network is equivalent to:

```
(&  (Plant  (performs  (Photosynthesis))
            (specs  (Mushroom)))
    (⇒  (Mushroom  (performs  (Photosynthesis)))
        (Photosynthesis  (status  (Disabled)))))
```

C.2 Parsing

Parsing involves performing an in-order traversal of the input network. Each nonterminal appearing in the input network generates one or more tuples during parsing. For each node N_i appearing in the nonterminal, there exists up to one incoming arc and zero or more outgoing arcs:

$$(N_{i-1}\ (p_1\ (N_i\ \ (p_2\ (N_{i+1}))\\ (p_3\ (N_{i+2}))\\ (p_4\ (N_{i+3}))\\ \ldots)))$$

KI collects the tuples involving N_i:

$$p_1(\ N_{i-1}\ N_i)\\ p_2(\ N_i\ N_{i+1})\\ p_3(\ N_i\ N_{i+2})\\ p_4(\ N_i\ N_{i+3})\\ \ldots$$

and their inverses:

$$p_1^{-1}(N_i\ N_{i-1})\\ p_2^{-1}(N_{i+1}\ N_i)\\ p_3^{-1}(N_{i+1}\ N_i)\\ p_4^{-1}(N_{i+1}\ N_i)\\ \ldots$$

Because the rules resulting from translating inverse tuples are sometimes fallacious, KI enables the user to inhibit the inclusion of inverse tuples. [2]

[2]Unless otherwise noted, all examples (e.g., those discussed in Chapter 6) include generating the inverse tuples during translation.

For the cuticle example, the tuples resulting from parsing the initial training are:

coveringPart(LeafEpidermis LeafCuticle)
composedOf(LeafCuticle Cutin)

and their inverses are:

coveringPartOf(LeafCuticle LeafEpidermis)
inCompositionOf(Cutin LeafCuticle)

After parsing, each tuple is translated into one or more statements in the target language.

C.3 Translating tuples

The greatest challenge during translation is coping with the ambiguities that result from a lack of explicit quantification in the input language. Translations for each reference to a constant that denotes a collection may assume either universal quantification, existential quantification, or neither (i.e., a literal translation). Those taken to be either universally or existentially quantified are said to be *figurative*: the reference to the collection is not literal; rather, it denotes a variable that ranges over elements of the collection. Further ambiguity remains for tuples containing figurative references since they could be taken to define any of a variety of rule types (e.g., argument-typing constraints, inheritance rules, relation-type rules).

As Figure C.3 illustrates, at least 17 possible translations exist for each tuple that references two collections, 7 exist for tuples that reference one collection, only one exists for tuples that reference no collection. Since each translation may or may not be included in an interpretation, a set of N possible

(a) literal translation

1. physicalPart(Plant Leaf)
   The (thing denoted by the) constant Plant has as a physical part the constant Leaf.

(b) translations that disambiguate figurative references

2. ∀ (x) isa(x Plant) ⇒ physicalPart(x Leaf)
   Every plant has as a physical part the constant Leaf (i.e., an inheritance rule).

3. ∀ (x) isa(x Leaf) ⇒ physicalPart(Plant x)
   The constant Plant has as physical parts every leaf.

4. ∃ (x) isa(x Plant) & physicalPart(x Leaf)
   Some plant has a physical part the constant Leaf.

5. ∃ (x) isa(x Leaf) & physicalPart(Plant x)
   The constant Plant has as a physical part some leaf.

6. ∀ (x y) isa(x Plant) & isa(y Leaf) ⇒ physicalPart(x y)
   Every plant has as physical parts every leaf.

7. ∃ (x y) isa(x Plant) & isa(y Leaf) & physicalPart(x y)
   Some plant has a physical part which is a leaf.

8. ∀ (x) isa(x Plant) ⇒ [∃ (y) isa(y Leaf) & physicalPart(x y)]
   Every plant has a physical part which is a leaf (i.e., a relation-type rule).

9. ∀ (x) isa(x Leaf) ⇒ [∃ (y) isa(y Plant) & physicalPart(y x)]
   Every leaf is a physical part of some plant.

10. ∃ (x) isa(x Plant) & [∀ (y) isa(y Leaf) ⇒ physicalPart(x y)]
    Some plant has as physical parts every leaf.

11. ∃ (x) isa(x Leaf) & [∀ (y) isa(y Plant) ⇒ physicalPart(y x)]
    Some leaf is a physical part of every plant.

(c) translations that define typing constraints

12. ∀ (x y) isa(x Plant) & physicalPart(x y) ⇒ isa(y Leaf)
    Only leaves are physical parts of plants.

13. ∀ (x y) physicalPart(x y) & isa(y Leaf) ⇒ isa(x Plant)
    Leaves are physical parts of only plants.

14. ∃ (x) isa(x Plant) & [∀ (y) physicalPart(x y) ⇒ isa(y Leaf)]
    Some plant has as physical parts only leaves.

15. ∃ (x) isa(x Leaf) & [∀ (y) physicalPart(y x) ⇒ isa(y Plant)]
    Some leaf is the physical part of only plants.

16. ∀ (x) physicalPart(Plant x) ⇒ isa(x Leaf)
    Only leaves are physical parts of the constant Plant.

17. ∀ (x) physicalPart(x Leaf) ⇒ isa(x Plant)
    The constant Leaf is a physical part of only plants.

Figure C.3: Candidate translations for *physicalPart(Plant Leaf)*

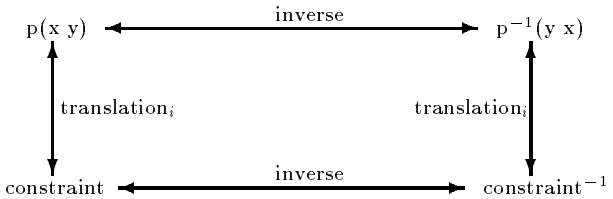


Figure C.4: Inverse tuples and inverse constraints

translations can suggest 2^N possible interpretations. Heuristics are therefore required to select among the candidate translations.

Note that symmetry exists between the following pairs of translations in Figure C.3: 2 and 3; 4 and 5; 8 and 9; 10 and 11; 12 and 13; 14 and 15; 16 and 17. Each element of a pair defines the *inverse constraint* of the pair's other element. For example, by the Inverse Rule (see Rule 10, Figure 3.7), the inverse tuple of *physicalPart*(*Plant Leaf*) is *physicalPartOf*(*Leaf Plant*). Translation 3 for this inverse tuple is: (∀ (x) isa(x Plant) ⇒ physicalPartOf(Leaf x)), which, by the Inverse Rule, can be rewritten as translation 2: (∀ (x) isa(x Plant) ⇒ physicalPart(x Leaf)). Figure C.4 illustrates this relationship.

Since inverse tuples are, by default, generated for independent translation, translations 3, 5, 9, 11, 13, 15, and 17 may be disregarded. Furthermore, translations 4, 6, 7, 10, 14, and 16 fail to capture common and useful relations among concepts in domains such as Botany (e.g., such translations would not be appropriate for any of the specifications in [PLM+88]); they are also disregarded. Thus, for each tuple, KI must select from among translations 1, 2, 8, and 12; making this selection involves identifying figurative references.

C.3.1 Identifying figurative references

Translating tuples extracted from a semantic network into first-order axioms involves determining which translation rules are appropriate for each tuple. Making this determination requires identifying figurative references, which KI does using existing knowledge (e.g., argument-typing constraints) relevant to each tuple. Specifically, for each tuple $p(x\ y)$, KI collects the most specific argument-typing constraints applicable to x and y and all existing constants already related to x by p.

In the example, one of the tuples parsed from the input network is $coveringPart(LeafEpidermis\ LeafCuticle)$. The domain (i.e., the argument-typing constraints imposed on the first argument) of predicate $coveringPart$ is $TangibleObject$, which is a generalization (i.e., a superset) of $LeafEpidermis$. Thus, the relation $physicalPart$ does not admit as a first argument the collection $LeafEpidermis$ but rather rather the individual elements of this collection; so this is a figurative reference.

Figurative references enable knowledge engineers to conceptualize domain knowledge in terms of prototypical entities without having to fuss with the details and rigor of first-order expressions (e.g., the syntax of quantification). While this abstraction has proved to be useful and appropriate for sketching out fragments of the domain theory by knowledge engineers, it provides only a partial specification of the intended internal knowledge structures (as demonstrated in Figure C.3). Therefore, during interpretation, KI must determine not only whether references to a collections should be taken literally or figuratively, but also the appropriate quantification and type of rule for each figurative reference.

Translation Rule 1: Identifying figurative references to known constants

rule a: If the first argument in a tuple is a subset of the predicate's domain, then the argument is used figuratively.

rule b: If the second argument is a subset of the predicate's range (i.e., the type constraint imposed on its second argument), then the second is used figuratively.

Translation Rule 2: Identifying literal references to known constants

rule a: If the first argument in a tuple is an element of the predicate's domain, then the argument is (necessarily) used literally.

rule b: When the second argument is an element of the predicate's range, then the argument is (necessarily) used literally.

As noted earlier, the domain of *coveringPart* is TangibleObject, which is a superset of LeafEpidermis; LeafEpidermis is thus used figuratively in the tuple *coveringPart*(*LeafEpidermis LeafCuticle*). The range of *composedOf* is *TangibleStuffType*, which has Cutin as an element; Cutin is therefore used literally in the tuple *composedOf* (*LeafCuticle Cutin*). [3] However, these translation rules apply only when the argument is a constant already defined in the knowledge base; other, heuristic translation rules are required to handle new constants.

C.3.2 Handling new constants

One of the ubiquitous activities in knowledge-base construction is introducing new concepts; one of the important functions of KI is to facilitate

[3] In the very rare cases when both translation rules 1 and 2 apply (e.g., a constant is both an element and a subset of the applicable argument types), then Rule 2 takes precedence.

this task. It is natural to reference new constants while embellishing existing concepts. In fact, a knowledge engineer will often not even know that a particular constant being referenced in a specification has not been previously defined. Therefore, it is preferable to enable new constants to be referenced within specifications without requiring special notational distinctions. KI provides an interface that has this property (i.e., the input specification language includes no special syntax for new constants).

It is particularly critical to determine the taxonomic specifications of each new constant (e.g., since they determine what predicates can reference the new concept); however, doing so should not involve obtrusive interruptions to the user's knowledge-base editing. Therefore, KI attempts to infer the taxonomic specifications of new concepts as they are introduced, recommends the inferred specifications to the user, then allows the user to accept or correct KI's recommendations.

When new constants are encountered in an input specification, KI first requests confirmation that the user intends to introduce a new concept. This precaution allows the user to correct spelling errors or employ psuedonyms when the user intends to refer to an existing concept. Next, KI determines whether or not the new constant is a collection.

Translation Rule 3: Identifying new collection constants

rule a: When any tuple references a new constant as the second argument and imposes an argument-typing constraint that is a subset of *Collection*, then the new constant denotes a collection.

rule b: Otherwise, when any tuple references a new constant as the second argument and the first argument is used figuratively and the

range of the predicate is a collection of individuals, KI assumes (heuristically) that the new constant is also used figuratively and is therefore a collection.

rule c: Otherwise, KI assumes (heuristically) that the new constant is used literally as an element of the applicable argument-typing constraints.

In the example, the tuples that reference the new constant *LeafCuticle* as the second argument are:

coveringPart(LeafEpidermis LeafCuticle)
inCompositionOf(Cutin LeafCuticle)

The applicable argument-typing constraint imposed on *LeafCuticle* by these tuples are BotanicalOrganismComponent and TangibleStuff, respectively. Neither of these typing constraints subset *Collection*, so translation Rule 3a does not apply. KI infers that *LeafCuticle* is a collection by Rule 3b applied to the tuple *coveringPart*(*LeafEpidermis LeafCuticle*), since *BotanicalOrganismComponent* is a collection of individuals and *LeafEpidermis* is used figuratively as the tuple's first argument.

Next, KI infers taxonomic specifications for the new constant by performing a ceiling-floor analysis on the applicable argument-typing constraints for each tuple in which the new constant appears as a second argument. A ceiling-floor analysis involves identifying existing constraints that restrict the membership of a new collection. The ceiling identifies a horizon in the existing taxonomy below which a collection is indexed (i.e., a set of collections, each element of which is a superset of the new collection). The ceiling imposed on

the new collection constant y by the tuple $p(x\ y)$ is identified by simply collecting all the applicable argument-typing constraints (assuming x is figurative in $p(x\ y)$): [4]

$$\{c \mid ako(x\ a)\ \&\ akoSlot(p\ b)\ \&\ classArgTwoType(a\ b\ c)\}$$

Similarly, the floor identifies a horizon in the existing taxonomy above which a collection is indexed (i.e., a set of collections each element of which the is a subset of the new collection). The floor imposed on the new collection constant y by the tuple $p(x\ y)$ is computed as:

$$\{c \mid ako(a\ x)\ \&\ akoSlot(b\ p)\ \&\ relationType(a\ b\ c)\}$$

Translation Rule 4: Identifying generalizations of a new collection

rule a: The new collection constant is (heuristically) a subset of each applicable typing constraint in the ceiling imposed on the new constant by tuples in which it is used figuratively as the second argument.

rule b: The new collection constant is (heuristically) an immediate proper subset of each of its most specific proper supersets.

Translation Rule 5: Identifying specializations of a new collection

rule a: The new collection constant is (heuristically) a superset of each applicable typing constraint in the floor imposed by tuples in which the new constant is used figuratively as the second argument.

rule b: The collection constant is (heuristically) an immediate proper superset of each of its most general proper subsets.

---

[4] If x is literal, then this formula becomes $\{c \mid isa(x\ a)\ \&\ akoSlot(p\ b)\ \&\ classArgTwoType(a\ b\ c)\}$

Translation Rule 6: Identifying types of a new collection

rule a: The new collection constant is (heuristically) an element of each collection that has an element any of the constant's supersets.

rule b: The new collection constant is (heuristically) a direct element of each of the most specific collections of which it is an element.

Translation Rule 7: Identifying types of a new constant

rule a: The new constant is an element of each collection that is an applicable typing constraint imposed by tuples in which the new constant is used literally.

rule b: The new constant is (heuristically) a direct element of each of the most specific collections of which it is an element.

In the example, the ceiling imposed on *LeafCuticle* by tuple *covering-Part*(*LeafEpidermis LeafCuticle*) is *BotanicalOrganismComponent*. The ceiling imposed on *LeafCuticle* by tuple *inCompositionOf*(*Cutin LeafCuticle*) is *TangibleStuff*. Neither tuple imposes a floor. By translation Rule 4a, KI infers that *LeafCuticle* subsets both *BotanicalOrganismComponent* and *TangibleStuff*. Since *BotanicalOrganismComponent* is an element of *TangibleObjectType*, translation Rule 6a suggests that *LeafCuticle* is an element of *TangibleObjectType*. Similarly, since *TangibleStuff* is an element of *Collection*, Rule 6a suggests that *LeafCuticle* is also a *Collection*. Since *BotanicalOrganismComponent* subsets *TangibleStuff*, Rule 4b concludes that *LeafCuticle* is an immediate subset of *BotanicalOrganismComponent*. Similarly, Rule 6b concludes that *LeafCuticle* is a direct element of *TangibleObjectType* Thus, *LeafCuticle* is tentatively inserted into the existing taxonomy of the knowledge base with the assertions:

superset(LeafCuticle BotanicalOrganismComponent)
elementOf(LeafCuticle TangibleObjectType)

which accounts for Rule F and Fact G in Figure 3.2c.

C.3.3 Establishing quantification

As noted earlier, KI considers only translations 1, 2, 8, and 12 of Figure C.3. Thus, for figurative subjects (i.e., for figurative references appearing as first arguments), KI assumes universal quantification, for figurative entries (i.e., for figurative references appearing as second arguments), KI assumes either existential quantification (e.g., relation-type rules) or universal quantification (e.g., argument-typing constraints). By convention, KI assumes that all applicable translations are intended. Thus, translations 8 and 12 are appropriate when both arguments are figurative; translation 2, when only the first argument is figurative; and translation 1, when both arguments are literal. The adequacy of this convention can be evaluated empirically and is likely to be domain specific.

Translation Rule 7: Determining quantification for figurative references

rule a: A figurative first argument with a literal second argument (heuristically) denotes universal quantification over its elements in an inheritance rule.

rule b: A figurative first argument with a figurative second argument (heuristically) denotes universal quantification over the elements of the first argument and existential quantification over the elements of the second argument in a relation-type rule.

rule c: A figurative first argument with a figurative second argument (heuristically) denotes universal quantification over the predicate's second argument in a argument-typing constraint.

In the example, the tuple *coveringPart*(*LeafEpidermis LeafCuticle*) includes figurative references for both arguments; it translates by Rules 7b and 7c (i.e., selecting the candidate translations 8 and 12 of Figure C.3) into Rules A and B of Figure 3.2. Similarly, the tuple *coveringPartOf*(*LeafCuticle LeafEpidermis*) includes figurative references for both arguments; it translates into Rules C and D of Figure 3.2. The tuple *composedOf*(*LeafCuticle Cutin*) includes a figurative first argument and a literal second argument; it translates by Rule 7a (i.e, selecting the candidate translation 2) into Rule E of Figure 3.2. The tuple *inCompositionOf*(*Cutin LeafCuticle*), while helping to determining the ceiling of *LeafCuticle*, does not produce a translation.

C.4 Adaptation: integrating translations

As each tuple is processed, KI integrates the translation into the knowledge base by identifying relevant prior knowledge modifying the new or relevant prior knowledge as necessary to accommodate the addition. [5] Relevant prior knowledge includes the axioms that define the ceiling and floor for each second argument in the pre-translation tuples. These are the argument-typing constraints and the relation-type rules that are within the scope of, may interact with, the new axioms.

When the scope of new and old rules overlap, KI must determine

[5]Note that this analysis is superficial in comparison with the integration that follows interpretation and is described in Chapters 3 – 5.

which rules take precedence: new rules can conjoin or disjoin with prior rules; or, they can supersede or be superseded by prior rules. For example, new existential constraints resulting from translation may specialize existing existential constraints: a prior rule might denote that photosynthetic organs contain some photosynthetic pigment; a new rule might denote that leaves, a subset of photosynthetic organs, contain chlorophyll, a type of photosynthetic pigment. In such a situation, KI must determine whether the prior and new rules disjoin (in this case, the old rule supersedes) or conjoin (in this case, the new rule supersedes). Similarly, new typing constraints resulting from translation may conflict with or, more typically, specialize existing typing constraints. In such cases, multiple axioms exist that could apply in a particular context. Therefore, KI must adjudicate between overlapping new and prior constraints to ensure, as much as possible, that the resulting constraints are not inconsistent or incomplete. Specifically, KI determines whether the new constraints should disjoin with or conjoin with prior constraints, whether the new constraints must be modified to avoid conflicting with prior constraints, or whether prior constraints must be modified.

It is, in general, unrealistic for learning systems to assume that all prior knowledge is correct or that all training is perfectly accurate. Thus, adjudicating conflicts that arise between new and prior knowledge seems inherently problematic. KI identifies the conflicts between new and prior constraints for the user and exploits heuristics to suggest how such conflicts might be resolved. Generally, potential conflicts occur among typing and existential constraints when satisfying every subordinate constraint does not guarantee satisfying every superordinate typing constraint. Specifically, some of the conflicts that can involve typing and existential constraints are:

1. A new typing constraint differs from existing local typing constraints; e.g.,

   (a) old: $classArgTwoType(x\ p\ y_1)$

   (b) new: $classArgTwoType(x\ p\ y_2)$

   (c) such that: $y_1 \neq y_2$

2. A new typing constraint fails to specialize existing superordinate typing constraints; e.g.,

   (a) old: $classArgTwoType(x_1\ p_1\ y_1)$

   (b) new: $classArgTwoType(x_2\ p_2\ y_2)$

   (c) such that: $ako(x_2\ x_1)$ & $akoSlot(p_2\ p_1)$ & $\neg ako(y_2\ y_1)$

3. A new typing constraint fails to generalize existing subordinate typing constraints; e.g.,

   (a) old: $classArgTwoType(x_1\ p_1\ y_1)$

   (b) new: $classArgTwoType(x_2\ p_2\ y_2)$

   (c) such that: $ako(x_1\ x_2)$ & $akoSlot(p_1\ p_2)$ & $\neg ako(y_1\ y_2)$

4. A new typing constraint fails to generalize existing local, subordinate, or superordinate existential constraints; e.g.,

   (a) old: $relationType(x_1\ p_1\ y_1)$

   (b) new: $classArgTwoType(x_2\ p_2\ y_2)$

   (c) such that: $ako(x_1\ x_2)$ & $akoSlot(p_1\ p_2)$ & $\neg ako(y_1\ y_2)$

5. A new existential constraint fails to equal existing local existential constraints; e.g.,

   (a) old: $relationType(x\ p\ y_1)$

   (b) new: $relationType(x\ p\ y_2)$

   (c) such that: $y_1 \neq y_2$

6. A new existential constraint fails to specialize existing superordinate existential constraints; e.g.,

   (a) old: $relationType(x_1\ p_1\ y_1)$

   (b) new: $relationType(x_2\ p_2\ y_2)$

   (c) such that: $ako(x_2\ x_1)$ & $akoSlot(p_2\ p_1)$ & $\neg ako(y_2\ y_1)$

7. A new existential constraint fails to generalize existing subordinate existential constraints; e.g.,

   (a) old: $relationType(x_1\ p_1\ y_1)$

   (b) new: $relationType(x_2\ p_2\ y_2)$

   (c) such that: $ako(x_1\ x_2)$ & $akoSlot(p_1\ p_2)$ & $\neg ako(y_1\ y_2)$

8. A new existential constraint fails to specialize existing local, subordinate, or superordinate typing constraints; e.g.,

   (a) old: $classArgTwoType(x_1\ p_1\ y_1)$

   (b) new: $relationType(x_2\ p_2\ y_2)$

   (c) such that: $ako(x_1\ x_2)$ & $akoSlot(p_1\ p_2)$ & $\neg ako(y_1\ y_2)$

9. A new local literal may not be admitted by existing local or superordinate typing constraints; e.g.,

   (a) old: $relationType(x_1\ p_1\ y_1)$

   (b) new: $p_2(x_2\ y_2)$

   (c) such that: $isa(x_2\ x_1)\ \&\ akoSlot(p_2\ p_1)\ \&\ \neg isa(y_2\ y_1)$

10. A new local literal differs from existing local literals; e.g.,

   (a) old: $p(x\ y_1)$

   (b) new: $p(x\ y_2)$

   (c) such that: $y_2 \neq y_1$

11. A a new inheritence literal may not be admitted by existing local, subordinate, or superordinate typing constraints; e.g.,

   (a) old: $relationType(x_1\ p_1\ y_1)$

   (b) new: $inherits(x_2\ (element\ p_2)\ y_2)$

   (c) such that: $ako(x_2\ x_1)\ \&\ akoSlot(p_2\ p_1)\ \&\ \neg isa(y_2\ y_1)$

12. A new inheritance literal differs from existing local, subordinate, or superordinate inheritance literals; e.g.,

   (a) old: $inherits(x_1\ (p_1\ p_2)\ y_1)$

   (b) new: $inherits(x_2\ (p_1\ p_2)\ y_2)$

   (c) such that: $ako(x_2\ x_1)\ \&\ y_2 \neq y_1$

In response to potential conflicts between new and prior rules, KI implements the following conflict resolution strategies:

rule a: Assume a new local typing constraint disjoins with all prior local and subordinate typing constraints (conflicts 1, 3).

rule b: If a new local typing constraint fails to specialize prior superordinate typing constraints, then minimally generalize the superordinate typing constraints to admit the new local typing constraint (conflict 2).

rule c: If a new local typing constraint fails to generalize all prior local and subordinate existential constraints, then minimally generalize it to admit all local and subordinate existential constraints (conflict 4).

rule d: Assume a new local existential constraint disjoins with existing local and subordinate existential constraints (conflicts 5 and 7).

rule e: If a new local existential constraint fails to specialize prior superordinate existential constraints, then minimally generalize the superordinate existential constraints to admit the new local existential constraint (conflict 6).

rule f: If a new local existential constraint fails to specialize prior local and superordinate typing constraints, then minimally generalize the superordinate typing constraints to admit the new local typing constraint (conflict 8).

rule g: If a new local literal is not admitted by existing local or superordinate typing constraints, then minimally generalize the typing constraint (conflict 9).

rule h: If a new inheritance literal is not admitted by existing local, subordinate, or superordinate typing constraints, then minimally generalize the typing constraint (conflict 11).

rule i: Assume a new inherited entry (i.e., second argument) of a single-entry predicate supersedes all prior entries inherited from superordinates (conflict 12).

rule j: Assume a new inherited entry of a multiple-entry predicate conjoins with all prior local and superordinate entries (conflict 11).

rule k: Assume all local, literal entries for a multiple-entry predicate conjoin (conflict 10).

rule l: If a single local typing constraint generalizes every other local typing constraint, then assume it conjoins with a disjunction of the other typing constraints (conflict 1).

rule m: If a single local existential constraint generalizes every existing local existential constraint, then assume it conjoins with a disjunction of the existing existential constraints (conflict 5).

rule n: If a single, common maximal specialization of every superset of the new collection constant exists, then assume the new collection is a proper subset of this common maximal specialization.

rule o: If a single, common maximal specialization of every collection of which the new collection constant is an element exists, assume the new collection is an element of this common maximal specialization.

Thus, KI assumes that new typing constraints disjoin with prior local and subordinate typing constraints but conjoin with (i.e., supersede) prior superordinate typing constraints. Why should local typing constraints disjoin? Consider examples involving the inverses of figurative references. Every time you drive

to the supermarket you get in your car, but getting in your car doesn't mean that you're driving to the supermarket; going to the supermarket is just one of many activities you could be performing. Furthermore, typing constraints must disjoin to permit multiple entries for a given property (e.g., a plant has as parts leaves and stems and flowers...). Therefore, it is prudent to assume that local typing constraints should disjoin.

Since KI, by default, assumes both existential and typing translations for figurative entries, it can be assumed that local existential constraints (that satisfy the local typing constraints) will exist whenever new local typing constraints fail to admit superordinate existential constraints. Since new local constraints conjoin with superordinate ones, and assuming new local existential constraints specialize the superordinate existential ones, new local typing constraints need only admit all the local and subordinate existential constraints (i.e., new local existential constraints will specialize any superordinate existential constraints that are not admitted).

Integrating translations of tuples into the knowledge base during interpretation is a special case of the general task of knowledge integration. Rather than relying on a general method to perform this task (e.g., see Chapters 2 through 5), KI implements special, narrowly-focused methods. For example, patterns of elaboration (e.g., computing the floor and ceiling applicable to the referenced constants) are directly coded in KI's implementation of interpretation. Furthermore, all inference involves meta-reasoning; the knowledge-base's inference engine is never used. Similarly, during adaptation, KI considers only a small, predetermined set of obvious conflict conditions and encodes, for each, a hard-wired conflict resolution strategy. Treating the integration of translated tuples during interpretation as a special case is warranted because the general

method for performing knowledge integration (e.g., hypothetical reasoning in the learning context) assumes that new information is (correctly) expressed in the representation language of the knowledge base. In situations where the learner must heuristically interpret the new information from an input language that differs from the representation language, it is reasonable to perform some amount of preliminary adaptation to correct obvious inconsistencies between the translation and relevant prior knowledge before committing to the correctness of the translation and invoking the general knowledge-integration method.

C.5 Summary

Interpretation translates new information expressed as semantic networks into first-order axioms and integrates them into the knowledge base. For each node appearing in the input network, KI

1. parses the node and its links into tuples
2. next translates the tuples into first-order axioms
3. modifies these and relevant existing axioms as necessary to accommodate each other
4. adds the resulting axioms to the knowledge base

The meaning of each new constant is determined from the context of its use as relevant argument-typing constraints and relation-type rules are identified and analyzed. For example, the constant $LeafCuticle$, when encountered during interpretation, is not defined in the knowledge base. KI exploits existing knowledge of the predicates $coveringPart$ and $composedOf$ to decide where

to place it in the taxonomy of the knowledge base. This procedure, however, has limitations. For example, when new constants appear only in tuples with other new constants (e.g., new predicate constants), it is unlikely that KI can correctly infer their taxonomic specifications or identify figurative uses. Also, while KI can accept new predicate constants, much of their taxonomic specifications (e.g., domain and range) must be provided explicitly in the input network.

KI's translation heuristics promote strong interpretations (e.g., universal quantification) for collection constants appearing as the source nodes of arcs. Thus, when told *Leaf epidermis is covered by leaf cuticle*, KI's interpretation assumes that it is reasonable to expect that any given leaf epidermis is covered by some leaf cuticle. However, it is not appropriate to assume that each leaf epidermis is covered by all leaf cuticles. Therefore, KI assumes a weaker interpretation (e.g., existential quantification) for collection nodes appearing as the destination nodes of arcs. [6] A consequence of KI's interpretation heuristics is that new information spawns pervasive, although relatively weak, expectations. The range of applicability of the expectations can be narrowed if and when subsequent experience violates them.

C.6 Lessons learned

At the onset, interpretation appeared to be a very significant part of knowledge integration. For example, determining where to index new constants in the taxonomic hierarchy is a ubiquitous and important task.

[6] These interpretation heuristics constitute an empirically-testable hypothesis re (1) the intention of (a) text book authors, (b) semantic net authors, and (2) the behavior of (a) text book readers, (b) semantic net readers.

Many of the test scenarios produced interesting results during interpretation. However, at least for this project, translation eventually turned to be less essential, less the true focus of knowledge integration. Existing knowledge was insufficient to resolve some of the ambiguity problems, such as determining whether figurative inverses should be asserted. Thus, some important aspects of translation relied upon conventions and weak heuristics.

Extending the syntax of the input networks to allow users to make explicit the currently ambiguous aspects of translation would remove the uncomfortable dependence on conventions and weak heuristics during translation. For example, it would be useful to extend the input language to enable the user to explicitly indicate:

1. the intended quantification over elements of referenced collections (e.g., see [Woo91])
2. whether or not a reference is figurative
3. whether or not the interpretation of the inverse tuples should be asserted

However, such an extension would also force the user to be much more adept with the target language (e.g., the internal representation language of the knowledge base) and force upon the user responsibilities for bridging the differences between specification language and the representation language. This draws into question the benefits of supporting a separate specification language and, consequently, the very need to perform interpretation.

Appendix D

A Model of Interestingness

The interestingness of a proposition is used by KI to estimate the interestingness of both concepts and views. It is an important factor in assessing the *activation level* (Section 2.1.2) while determining which candidate view will be used to extend the learning context. The following heuristics are used by KI to appraise how interesting a proposition is.

1. A proposition that references a term (i.e., non-predicate constant) appearing in the training is deemed extremely interesting. In the example, this would include any proposition referencing *LeafEpidermis*, *LeafCuticle*, or *Cutin*.

2. A proposition that participates in an explanation of some fact in the initial learning context (e.g., some fact that instantiates the new information) is deemed extremely interesting. For example, the fact $isa(LeafCuticle_1\ Cutin)$ participates in the explanation of $composedOf(LeafCuticle_1\ Cutin)$ (by Rule 5a, Figure 3.4).

3. A proposition that denotes a domain goal or function is deemed extremely interesting (e.g., the proposition establishes $health(x\ Facilitated)$ for some living thing bound to x).

4. A proposition that is identified as being anomalous is deemed extremely interesting. A proposition is considered anomalous if any of the following conditions are satisfied:

   (a) The proposition explicitly violates a constraint (e.g., the consequence $covers(LeafCuticle_1\ LeafEpidermis_1)$ violates the argument-typing constraints defined for *covers*; see Section 3.5.1).

   (b) Conflicting truth values of the proposition are established (e.g., $p(x\ y)$ and $\neg p(x\ y)$ are both established).

   (c) The proposition involves a predicate considered likely for one of its arguments, and that argument is disabled; e.g., $p(x\ y)$ & $likelyFor(p\ x)$ & $status(x\ Disabled)$.

   (d) The proposition establishes that an essential component or behavior becomes disabled. Currently, *essential component* is defined to include organs, and *essential behaviors* include development, the processing of essential assimilates (e.g., light, water, carbon dioxide, mineral nutrients, etc.), and goal behaviors.

   (e) The fact supports adopting a general rule denoting that an existing collection has no elements. This occurs when some explanation of the fact can be generalized into a rule of the form

   $$[\forall\ (x_i)\ isa(x_i\ X) \Rightarrow status(x_i\ Disabled)]$$

   for some collection X.

5. A proposition that is a consequence of the new information is deemed very interesting.

6. A proposition that references a hypothetical instance of a class appearing in the training is deemed very interesting. In the example, this would include any proposition referencing $LeafEpidermis_1$ or $LeafCuticle_1$.

7. A proposition that refutes an assumption (e.g., its negation is referenced within an *unless* clause) is deemed very interesting.

8. A proposition not yet established in the learning context but whose arguments are included in the learning context (i.e., the proposition establishes an as yet unconsidered relation between concepts already considered) is deemed moderately interesting.

9. A proposition that references a modulatory predicate is deemed moderately interesting. Modulatory predicates denote how events affect each other (e.g., *facilitates, controls, restricts,* etc.).

10. A proposition that references an attribution predicate (versus a relation predicate) is deemed moderately interesting.

11. A proposition that denotes a decomposing relation between something and its parts is deemed marginally interesting.

12. A proposition that denotes a relation between an event and the entities that participate in the event is deemed marginally interesting.

All other propositions are, by default, deemed to be negligibly interesting. Figure 4.10 summarizes these heuristics.

BIBLIOGRAPHY

[Ack92] L.E.H. Acker. Access methods for large, multifunctional knowledge bases. Technical Report AI92-183, Department of Computer Sciences, University of Texas at Austin, 1992.

[And83] J. Anderson. *The Architecture of Cognition.* Cambridge, MA: Harvard University Press, 1983.

[AP94] L. Acker and B. Porter. Extracting viewpoints from knowledge bases. In *Proceedings of the National Conference on Artificial Intelligence*, pages 547–552, 1994.

[Aus63] D.P. Ausubel. *The psychology of meaningful verbal learning.* New York: Grune & Stratton, 1963.

[BBB+83] B.G. Buchanan, D. Barstow, R. Bechtal, J. Bennett, W. Clancey, C. Kulikowski, T. Mitchell, and D.A. Waterman. Constructing an expert system. In F. Hayes-Roth, D.A. Waterman, and D.B. Lenat, editors, *Building Expert Systems.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1983.

[BD81] A. Barr and J. Davidsom. Knowledge representation. In A. Barr and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume Volume 1. Los Altos, CA: William Kaufmann, Inc., 1981.

[BM77] J.D. Bransford and A. McCarrell. A sketch of a cognitive approach to comprehension: some thoughts about understanding what it means to comprehend. In P.N. Johnson-Laird and P.C. Wason, editors, *Thinking: Readings in Cognitive Science*. Cambridge: Cambridge University Press, 1977.

[Boo85] J.H. Boose. A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man-Machine Studies*, 20:21–43, 1985.

[Bra79] J.D. Bransford. *Human Cognition*. Belmont, CA: Wadsworth, 1979.

[BW77] D. Bobrow and T. Winograd. An Overview of KRL, a Knowledge Representation Language. *Cognitive Science*, 1(1):3–46, 1977.

[Car86] J.G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.

[Cla90] P. Clark. Representing knowledge as arguments: Applying expert system technology to judgemental problem-solving. In T.R. Addis and R.M. Muir, editors, *Research and Development in Expert Systems*, volume Volume VII, pages 147–159. Cambridge University Press, 1990.

[Coh84] D. Cohen. A forward inference engine to aid in understanding specifications. In *Proceedings of the National Conference on Artificial*

Intelligence, pages 56–60, 1984.

[Coh85] P.R. Cohen. *Heuristic Reasoning About Uncertainty: An Artificial Intelligence Approach*. Boston: Pitman, 1985. (Based on PhD Dissertation, Computer Science Department, Stanford University).

[DeK85] J. DeKleer. How circuits work. In D.G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*, pages 205–280. Cambridge, MA: The MIT Press, 1985.

[DeK86] J. DeKleer. An assumption-based tms. *Artificial Intelligence*, 28:127–162, 1986.

[Der90] M.A. Derthick. An epistemological interface for cyc. Technical Report ACT-CYC-084-90, MCC, 1990.

[Die82] T.G. Dietterich. Learning and inductive inference. In P.R. Cohen and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume Volume 3. Los Altos, CA: William Kaufmann, Inc., 1982.

[Die86] T.G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1(3):287–315, 1986.

[Dij75] E. Dijkstra. Guarded commands, non-determinancy and formal derivation of programs. *Communications of the Association of Computing Machinery*, 18:453–457, 1975.

[DM86] G. DeJong and R. Mooney. Explanation–based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[Dow90] K. Downing. The qualitative criticism of circulatory models via bipartite teleological analysis. In *Proceedings of the 1990 Workshop on Qualitative Reasoning*, 1990.

[Doy79] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[DW93] S. Dixon and W. Wobcke. The implementation of a first-order logic agm belief revision system. In *Proceedings of the Fifth IEEE International Conference on Tools with AI*, pages 40–47, 1993.

[DW95] S. Dixon and W. Wobcke. A nonmonotic reasoning belief revision system. In *(submitted to) The Eighteenth Australian Computer Science Conference*, 1995.

[EL75] L.D. Erman and V.R. Lesser. A multi-level organization for problem solving using many diverse, cooperating sources of knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 483–490, 1975.

[FF91] B. Falkenhainer and K. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[Fis87] D.H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[FN88] S. Fickas and P. Nagarajan. Being suspicious: Critiquing problem specifications. In *Proceedings of the National Conference on Artificial Intelligence*, pages 19–24, 1988.

[Fra93] D.W. Franke. A theory of teleology. Technical Report AI93-201, Department of Computer Sciences, University of Texas at Austin, 1993.

[Gag78] E.D. Gagne. Long–term retention of information following learning from prose. *Review of Educational Research*, 48:629–665, 1978.

[Gag85] E.D. Gagne. *The Cognitive Psychology of School Learning.* Bostaon, MA: Little, Brown, and Company, 1985.

[Gal92] J.R. Galliers. Autonomous belief revision and communication. In P. Gardenfors, editor, *Belief Revision.* Cambridge University Press, 1992.

[Gar88] P. Gardenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* Cambridge, MA: The MIT Press, 1988.

[Gar92] P. Gardenfors. Belief revision: An introduction. In P. Gardenfors, editor, *Belief Revision.* Cambridge University Press, 1992.

[Gen83] D. Gentner. Structure-mapping: A theoretical framefork for analogy. *Cognitive Science*, pages 155–170, 1983.

[Gen89] D. Gentner. Mechanisms of analogical learning. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning.* London: Cambridge University Press, 1989.

[GH86] A.L. Glass and K.J. Holyoak. *Cognition.* Random House, 1986.

[Gil94] Y. Gil. Knowledge refinement in a reflective architecture. In *Proceedings of the National Conference on Artificial Intelligence*, pages 520–526, 1994.

[Gin87] M.L. Ginsberg. Introduction. In M.L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann Publishers, Inc., 1987.

[GN87] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1987.

[GP94] M. Goldszmidt and J. Pearl. Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 661–672, 1994.

[Gro86] B.J. Grosz. The representation and use of focus in a system for understanding dialogs. In B.J. Grosz, K. Sparck Jones, and B.L. Webber, editors, *Readings in Natural Language Processing*. Morgan Kaufmann Publishers, Inc., 1986.

[Gru91] T. Gruber. Learning why by being told what. *IEEE Expert*, 6(4):65–75, 1991.

[Guh91] R.V. Guha. Contexts: A formalization and applications. Technical Report ACT-CYC-423-91, MCC, 1991.

[Han92] S.O. Hansson. A dyadic representation of belief. In P. Gardenfors, editor, *Belief Revision*. Cambridge University Press, 1992.

[Har86] G. Harman. *Change in View: Principles of Reasoning*. Cambridge, MA: The MIT Press, 1986.

[Hay85] P.J. Hayes. The second naive physics manifesto. In R.J. Brachman and H.J. Levesque, editors, *Readings in Knowledge Representation*, pages 467–485. Morgan Kaufmann Publishers, Inc., 1985.

[HC74] S. Haviland and H. Clark. What's new? Acquiring new information as a process in comprehension. *Journal of Verbal Learning and Verbal Behavior*, 13:512–521, 1974.

[Hen75] G.G. Hendrix. Partitioned networks for the mathematical modeling of natural language semantics. Technical Report NL-28, Department of Cputer Sciences, University of Texas at Austin, 1975.

[IL94] Y. Iwasaki and A.Y. Levy. Automated model selection for simulation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1183–1190, 1994.

[JLB84] P.N. Johnson-Laird and B.G. Bara. Syllogistic inference. *Cognition*, 16:1–61, 1984.

[Kah73] D. Kahneman. *Attention and Effort.* Englewood Cliffs, NJ: Prentice–Hall, Inc., 1973.

[KBR91] R.M. Kaplan and G. Berry-Rogghe. Knowledge-based acquisition of causal relationships in text. *Knowledge Acquisition*, 3(3):317–337, 1991.

[KC85] S. Kedar-Cabelli. Purpose-directed analogy. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 150–159, 1985.

[Kel88] R.M. Keller. Defining operationality for explanation-based learning. *Artificial Intelligence*, 35:227–241, 1988.

[Kuh70] T.S. Kuhn. *The Structure of Scientific Revolutions.* Chicago, IL: University of Chicago Press, 1970.

[Kui85] B.J. Kuipers. Common sense reasoning about causality: Deriving behavior from structure. In D.G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*, pages 169–203. Cambridge, MA: The MIT Press, 1985.

[KvD78] W. Kintsch and T. van Dijk. Toward a model of text comprehension and production. *Psychological Review*, 85(5):363–394, 1978.

[Len76] D.B. Lenat. AM: An artificial intelligence approach to discovery in mathematics as heuristic search. Technical report, Ph.D. dissertation, Computer Science Department, Stanford University, 1976.

[LF87] D.B. Lenat and E.A. Feigenbaum. On the thresholds of knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1987.

[LG90] D. Lenat and R. Guha. *Building Large Knowledge–Based Systems.* Reading, MA:Addison-Wesley, 1990.

[LM90] S.L. Lytinen and C.E. Moon. A comparison of learning strategies in second language learning. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 377–383, 1990.

[Mar77] D. Marr. Artificial intelligence – a personal view. *Artificial Intelligence*, 9:37–48, 1977.

[May80] R. Mayer. Elaboration techniques that increase the meaningfulness of technical text: An experimental test of the learning strategy hypothesis. *Journal of Educational Psychology*, 72(6):770–784, 1980.

[MBF77] C. Morris, J. Bransford, and J. Franks. Levels of processing versus transfer appropriate processing. *Journal of Verbal Learning and Verbal Behavior*, 16:519–533, 1977.

[McA80] D. McAllester. An outlook on truth maintenance. Technical Report AI Memo No. 551, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1980.

[McC58] J. McCarthy. Programs with common sense. In *Proceedings of the Symposium on the Mechanization of Thought Processes*, pages 77–84, 1958. (Reprinted in M. L. Minsky (Ed.), Semantic Information Processing, Cambridge, MA: MIT Press, pp. 403-409, 1968.).

[McC77] J. McCarthy. Epistemological problems in artificial intelligence. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1038–1044, 1977.

[McC85] K.F. McCoy. The role of perspective in responding to property misconceptions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 791–793, 1985.

[McC89a] J. McCarthy. Artificial intelligence and logic. In R. Thomason, editor, *Philosophical Logic and Artificial Intelligence*. Dordrecht: Kluwer Academic, 1989.

[McC89b] K.F. McCoy. Generating context-sensitive responses to object-related misconceptions. *Artificial Intelligence*, 41:157–195, 1989.

[McD81] D. McDermott. Artificial intelligence meets natural stupidity. In J. Haugeland, editor, *Mind Design*, pages 143–160. Cambridge, MA: The MIT Press, 1981.

[MH69] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[Mic86] R.S. Michalski. Understanding the nature of learning: Issues and research directions. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume Volume II, pages 3–25. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.

[Mic94] R.S. Michalski. Inferential theory of learning: Developing foundations for multistrategy learning. In R.S. Michalski and G. Tecuci, editors, *Machine Learning: An Multistrategy Approach*, volume Volume IV, pages 3–61. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1994.

[Min81] M Minsly. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*, pages 95–128. Cambridge, MA: The MIT Press, 1981.

[Min88] S. Minton. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 564–569, 1988.

[MKKC86] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[MMM85] K. R. McKeown, W. Myron, and K. Matthews. Tailoring explanations for the user. In *Proceedings of the International Joint Con-*

ference on Artificial Intelligence, pages 794–798, 1985.

[Moo92] R. J. Mooney. Batch versus incremental theory refinement. In *Proceedings of the AAAI Symposium on Knowledge Assimilation*, pages 104–113, 1992.

[Mor89] K. Morik. Sloppy modeling. In K. Morik, editor, *Knowledge Representation and Organization in Machine Learning*, pages 107–134. Springer-Verlag, 1989.

[Mor91] K. Morik. Underlying assumptions of knowledge acquisition and machine learning. *Knowledge Acquisition*, 3:137–156, 1991.

[Mos83] J. Mostow. Machine translation of advice into a heuristic search procedure. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 367–404. Tioga Publishing, 1983.

[MP89] K.S. Murray and B.W. Porter. Controlling search for the consequences of new information during knowledge integration. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 290–295, 1989.

[MS86] J. Mostow and W. Swartout. Towards explicit integration of knowledge in expert systems: An analysis of Mycin's therapy selection algorithm. In *Proceedings of the National Conference on Artificial Intelligence*, pages 928–935, 1986.

[Mur88] K.S. Murray. KI: An experiment in automating knowledge integration. Technical Report AI88-90, Department of Computer Sciences, University of Texas at Austin, 1988.

[Mur90] K.S. Murray. Improving explanatory competence. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 309–316, 1990.

[New90] A. Newell. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press, 1990.

[Nil91] N. J. Nilsson. Logic and artifical intelligence. *Artificial Intelligence*, 47:31–56, 1991.

[OM90] D. Ourston and R.J. Mooney. Changing the rules: A comprehensive approach to theory refniement. In *Proceedings of the National Conference on Artificial Intelligence*, pages 815–820, 1990.

[PB91] M.J. Pazzani and C.A. Brunk. Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition*, 3:157–173, 1991.

[PBH90] B.W. Porter, R.E. Bareiss, and R.C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45, 1990.

[Pia46] J. Piaget. Piaget's theory. In P. Mussen, editor, *Carmichael's Manual of Child Psychology*, pages 703–732. John Wiley and Sons, Inc., 1946.

[PLM+88] B. Porter, J. Lester, K. Murray, K. Pittman, A. Souther, L. Acker, and T. Jones. AI research in the context of a multifunctional knowledge base. Technical Report AI88-88, Department of Computer Sciences, University of Texas at Austin, 1988.

[Qui90] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[Reb83] R. Reboh. Extracting useful advice from conflicting expertise. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 145–150, 1983.

[Red79] L. Reder. The role of elaborations in memory for prose. *Cognitive Psychology*, 11:221–234, 1979.

[Red82a] L.M. Reder. Elaborations: When do they help and when do they hurt? *Text*, 2:211–224, 1982.

[Red82b] L.M. Reder. Plausibility judgements versus fact retrieval: Alternative strategies for sentence verification. *Psychological Review*, 89:250–280, 1982.

[RM92] B.L. Richards and R.J. Mooney. Learning relations by pathfinding. In *Proceedings of the National Conference on Artificial Intelligence*, pages 50–55, 1992.

[RN86] P.S. Rosenbloom and A. Newell. The chunking of goal hierarchies: A generalized model of practice. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume Volume II, pages 247–288. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.

[RP94] J. Rickel and B. Porter. Automated modeling for answering prediction questions: Selecting the time scale and system boundary. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1191–1198, 1994.

[RS83] N. Roser and D.L. Schallert. Reading research: What it says to the school psychologist. In T.R. Kratochwill, editor, *Advances in school psychology*, volume Volume 3, pages 237–268. Hillsdale, N.J.: Lawrence Erlbaum, 1983.

[RSMH86] D.E. Rumelhart, P. Smolensky, J.L. McClelland, and G.E. Hinton. Schemata and sequential thought processes in pdp models. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, volume Volume II: Physchological and Biological Models, pages 7–57. Cambridge, MA: MIT Press, 1986.

[Sch76] D.L. Schallert. Improving memory from prose: The relationship between depth of processing and context. *Journal of Verbal Learning and Verbal Behavior*, 15:621–632, 1976.

[Sch82] D.L. Schallert. The significance of knowledge: A synthesis of research related to schema theory. In W. Otto and S. White, editors, *Reading Expository Material*, pages 13–48. New York: Academic Press, 1982.

[Sch87] D.L. Schallert. Thought and language, content and structure. In J. Squire, editor, *The dynamics of language learning: Research in reading and English*, pages 65–89. Urbana, IL: NCTE, 1987.

[Sco83] P.D Scott. Learning: The construction of a posteriori knowledge structures. In *Proceedings of the National Conference on Artificial Intelligence*, pages 359–363, 1983.

[SG87] D. Subramanian and M.R. Genesereth. The relevance of irrelevance. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 416–422, 1987.

[Sim81] H.A. Simon. *The Sciences of the Artificial.* Cambridge, MA: The MIT Press, 1981.

[Sim83] H.A. Simon. Why study machine learning? In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 25–38. Tioga Publishing, 1983.

[SS77] R. Stallman and G. Sussman. Forward reasoning and dependency directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–196, 1977.

[SS89] W.R. Swartout and S.W. Smoliar. Explanation: A source of guidance for knowledge representation. In K. Morik, editor, *Knowledge Representation and Organization in Machine Learning*, pages 1–16. New York: Springer-Verlag, 1989.

[Sut88] D. Suthers. Providing multiple views of reasoning for explanation. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 435–442, 1988.

[SV83] P.D Scott and R.C. Vogt. Knowledge oriented learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 432–435, 1983.

[Swa83] W. Swartout. XPLAIN: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21:285–325, 1983.

[SWMB85] R. Smith, H. Winston, T. Mitchell, and B. Buchanan. Representation and use of explicit justification for knowledge base refinement. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 673–680, 1985.

[Tap80] A. Tappel. Some algorithm design methods. In *Proceedings of the National Conference on Artificial Intelligence*, pages 64–67, 1980.

[Wat70] D.A. Waterman. Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence*, 1:121–170, 1970.

[Wei78] C. Weinstein. Elaboration skills as a learning strategy. In H.F. Jr. O'Neil, editor, *Learning Strategies*, pages 157–209. Academic Press, 1978.

[Wil88] D.C. Wilkins. Knowledge base refinement using apprenticeship learning techniques. In *Proceedings of the National Conference on Artificial Intelligence*, pages 646–651, 1988.

[Wob94] W. Wobcke. Algorithms for iterative belief revision. In *Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence*, 1994.

[Woo91] W.A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 45–94. San Mateo, CA: Morgan Kaufmann, 1991.

[Wro89] S. Wrobel. Demand-driven concept formation. In K. Morik, editor, *Knowledge Representation and Organization in Machine Learning*, pages 289–319. New York: Springer-Verlag, 1989.

VITA

Kenneth S Murray was born in Iowa City, Iowa, on October 31, 1956, the son of James Nigel Murray and Patricia Gilliam Murray. He was graduated from West High School, Iowa City, Iowa, in 1974. In December 1983 he received the degree of Bachelor of Science from the University of Iowa and, in September 1984, entered the Graduate School of The University of Texas.

Permanent address: 8524 Burnet Road, #1232
Austin, Texas 78757

This dissertation was typeset[1] with LaTeX by the author.

[1]LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for The University of Texas at Austin dissertation format was written by Khe-Sing The.