

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

STNC
1071

~~CALNET~~

~~CALNET~~

~~CALNET~~

Learning by Understanding Analogies

by

~~CALNET~~

Russell Greiner

COMPUTER SCIENCE DEPT.
TECHNICAL REPORT 88-05

UNIVERSITY LIBRARIES
CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

Department of Computer Science

Stanford University
Stanford, CA 94305

~~CALNET~~



ROOM USE ONLY

LEARNING BY UNDERSTANDING ANALOGIES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

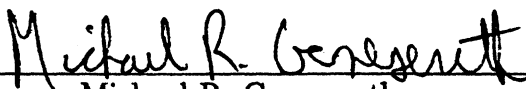
By
Russell Greiner
September 1985

© Copyright 1985

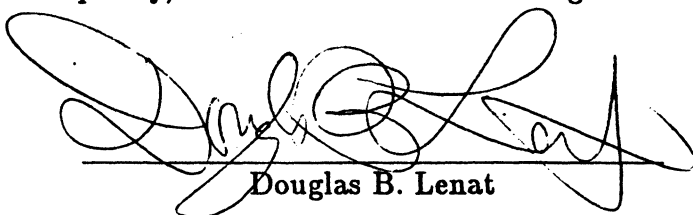
by

Russell Greiner


I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.


Michael R. Genesereth
(Principal Adviser)

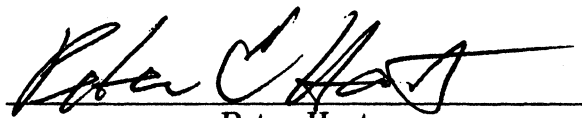
I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.


Douglas B. Lenat

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.


Bruce G. Buchanan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.


Peter Hart
(President, Syntelligence)

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies & Research

Acknowledgements

*This is one small step for mankind;
One giant step for me.*

— SHS

Both this dissertation and I owe a great deal to a good many people. By far our biggest intellectual debt is to my principal advisor, Professor Michael R. Genesereth, and his annoying habit of always being right. Essentially all of the ideas presented in this dissertation stem from his ideas and insights. He was also the source of much needed support and encouragement, offered during those many times when I ~~know~~ felt this research would never be finished.

The other members of my reading committee also provided major contributions. Let me first thank Dr. Peter Hart, for his services above-and-beyond the call of duty. Despite the inevitable hassles of starting up a new company, he still, somehow, managed to find large chunks of time in which to offer his respected expertise and assistance. His crisp appraisals often helped me prune and order my research paths; more than once his comments kept me from wandering down fruitless paths.

I owe a lot to Professor Douglas Lenat as well. As my mentor during most of my “early graduate student hood”, he introduced me to the general field of Artificial Intelligence, and vice versa. I still find him a constant source of creative new ideas, and inspiration.

Professor Bruce Buchanan completes my committee. His precision kept me honest, forcing me to think through murky ideas. He also proposed a great many suggestions and extensions, when that was appropriate (during my thesis proposal days) and, appropriately, many pruning heuristics to help me finish this research in a finite amount of time.

I have also had very high quality help from many of my colleagues. Let me especially thank Jock Mackinlay, who suffered through more thesis drafts and more preliminary presentations than anyone should have to endure. I also appreciate the useful advise, as well as comradeship, offered from many of my MUGS cohorts, including Tom Dietterich, Jeff Finger, Matt \mathbb{R}_{\wedge}^L Ginsberg, Mike Lowry, Jeff Rosen-schein, Stuart Russell, David Smith and Devika Subramanian. I also thank Steve Tappel, Tom Pressburger, Professor Lindley Darden, Dr. Frederick Hayes-Roth and Dr. Johan deKleer for their earlier contributions to the development of these ideas.

Dr. Shmuel Shaffer provided the electronics expertise needed to implement the specific examples which appeared throughout the thesis. His willingness to help during my times of need was yet even more important. Steve Katzman helped me through a jungle of philosophical and mathematical arguments, as well as serving as a general, and effective, sounding board:

Both Drs. Leslie Lamport and Howard Trickey provided a great deal of needed — and greatly appreciated — assistance in formatting this report in the neo-natal \LaTeX language. This research was done at the Knowledge Systems Laboratory (formerly the Heuristic Programming Project) at Stanford University, and supported by the Office of Naval Research contract # N00014-81-K-0004.

Let me close the string of thanks by acknowledging the important people of my life, friends who stood by me over the years, each helping me as much as possible whenever possible: Claire Coire, Devora Fishman, Lynne Gonda, Jock Mackinlay and Shmuel Shaffer, and, of course, my family. A special thanks goes to my parents: To my father, for volunteering to proofread this *magnus opus*, and to my mother, for being a constant source of love and caring. It's times of stress, such as writing this dissertation, when I realized who my friends really are... I could not have finished without your help; my sincere thanks to all of you.

Contents

| | |
|---|-----------|
| Acknowledgements | iv |
| Abstract | 1 |
| 1 Introduction | 1 |
| 1.1 Motivation | 1 |
| 1.2 Outline | 4 |
| 1.3 Example | 7 |
| 1.4 Objectives | 10 |
| 2 General Analogical Inference | 12 |
| 2.1 Definition of Analogical Inference | 13 |
| 2.2 Example of Analogical Inference | 17 |
| 2.3 Definition of Analogy Relationship | 22 |
| 2.4 Triviality and Aboutness | 23 |
| 2.5 Complexities of Analogical Inference | 28 |
| 3 Useful Analogical Inference | 36 |
| 3.1 Analogical Inference for a Purpose | 37 |
| 3.2 Definition of Useful Analogical Inference | 38 |
| 3.3 Example of Useful Analogical Inference | 40 |
| 3.4 Intuitions about Useful Analogies | 43 |
| 4 Use of Abstractions | 46 |

| | | |
|----------|---|------------|
| 4.1 | Informal Discussion of Abstractions | 47 |
| 4.2 | Use of Abstractions in Analogies | 54 |
| 4.3 | Why Abstractions <i>Should</i> be Used for Analogical Inference | 59 |
| 4.4 | Why Abstractions <i>Can</i> be used for Analogical Inference | 65 |
| 4.5 | Comparison of Abstractions | 70 |
| 5 | Ranking Analogical Inferences | 72 |
| 5.1 | Framework for Ranking Analogies | 72 |
| 5.2 | Abstraction-Based Heuristics | 74 |
| 5.2.1 | H_{JK} : Justification Kernel | 74 |
| 5.2.2 | H_{CC} : Common context | 78 |
| 5.3 | Least Constraining Analogies | 80 |
| 5.3.1 | Motivation for the Least Constraint Maxim | 80 |
| 5.3.2 | Syntactic Criterion: Additions to the Theory | 83 |
| 5.3.3 | Semantic Criterion: Constraints on Possible Worlds | 84 |
| 5.3.4 | H_{MGA} : Most General Abstraction | 92 |
| 5.3.5 | H_{FC} : Fewest Conjectures | 94 |
| 5.3.6 | H_{FT} : Findable Terms | 99 |
| 5.4 | Summary of Heuristics | 103 |
| 5.4.1 | Classification and Synopsis of Heuristics | 103 |
| 5.4.2 | Interactions of Heuristics | 105 |
| 6 | Specification of the NLAG Process | 108 |
| 6.1 | Components of NLAG System | 109 |
| 6.1.1 | <i>Verify</i> Module | 111 |
| 6.1.2 | <i>ComAbs</i> Module | 112 |
| 6.1.3 | 1: <i>Find-Kernel</i> Module | 113 |
| 6.1.4 | 2: <i>Inst-Source</i> Module | 114 |
| 6.1.5 | 3: <i>Inst-Target</i> Module | 115 |
| 6.2 | Underlying System | 116 |
| 6.2.1 | Why Use MRS? | 116 |
| 6.2.2 | <i>FC-Find</i> Module | 118 |

| | | |
|----------|---|------------|
| 6.2.3 | <i>C</i> Truep Module | 119 |
| 6.2.4 | Summary of MRS Additions | 128 |
| 6.3 | Requirements of the <i>NLAG</i> System | 134 |
| 7 | Experimental Results | 138 |
| 7.1 | Initial Set Up | 139 |
| 7.2 | <i>NLAG</i> 's Functionality — Runs#1 | 142 |
| 7.2.1 | Using Generator#1 | 144 |
| 7.2.2 | Using Generator#2 | 148 |
| 7.3 | <i>NLAG</i> 's Sensitivity to Initial Theory— Runs#2a | 150 |
| 7.4 | <i>NLAG</i> 's Sensitivity to Abstractions — Runs#2b | 152 |
| 7.5 | Ablation of Analogy — Runs#3 | 154 |
| 7.6 | Analysis | 157 |
| 7.6.1 | Utility of Analogies | 158 |
| 7.6.2 | Utility of Maximal Generality | 160 |
| 7.6.3 | Questionable Utility of Abstraction Label | 161 |
| 7.6.4 | Summary | 162 |
| 8 | Semantic Definition of Analogy | 164 |
| 8.1 | Motivation and Intuitions | 164 |
| 8.2 | Partial Interpretations | 166 |
| 8.2.1 | Why Not Use Tarskian Semantics | 166 |
| 8.2.2 | Partial Interpretation Semantics | 166 |
| 8.2.3 | Using Partial Interpretations to Describe Learning | 174 |
| 8.3 | Semantic Definition of Analogy | 177 |
| 8.4 | Semantic Definition of Analogical Inference | 179 |
| 9 | Standard Definitions of Analogies | 184 |
| 9.1 | Common Views of Analogy | 185 |
| 9.2 | How $\ \sim_{PT}$ Differs from Other Versions of Analogical Inference | 187 |
| 9.2.1 | View that Analogies must be Constructed | 187 |
| 9.2.2 | Definition of Maximal Commonality | 190 |

| | | |
|-----------|---|-------------|
| 9.2.3 | Limitations of the H_{MSA} Rule | 194 |
| 9.3 | How <i>Analogy_F</i> Differs from Other Versions of Analogy | 200 |
| 9.3.1 | Exact Formulation is Important | 203 |
| 9.3.2 | Need for Reformulation | 206 |
| 9.4 | Literature Survey | 210 |
| 10 | Conclusion | 220 |
| 10.1 | Placement | 220 |
| 10.2 | Future Work | 221 |
| 10.3 | Contributions | 226 |
| A | Notes from the Text | A-1 |
| A.1 | Notes from Chapter 1 | A-1 |
| A.2 | Notes from Chapter 2 | A-5 |
| A.3 | Notes from Chapter 3 | A-22 |
| A.4 | Notes from Chapter 4 | A-25 |
| A.5 | Notes from Chapter 5 | A-33 |
| A.6 | Notes from Chapter 6 | A-36 |
| A.7 | Notes from Chapter 7 | A-38 |
| A.8 | Notes from Chapter 8 | A-45 |
| A.9 | Notes from Chapter 9 | A-48 |
| A.10 | Notes from Chapter 10 | A-57 |
| B | Data for the Experiments | A-60 |
| B.1 | Actual <i>NLAG</i> Code | A-60 |
| B.2 | Initial Theory | A-66 |
| B.2.1 | General Information | A-66 |
| B.2.2 | Initial Electric Facts, <i>EC</i> | A-72 |
| B.2.3 | Initial Hydraulics Facts, <i>FS</i> | A-75 |
| B.2.4 | Facts about the Target Problem, <i>PT</i> | A-77 |
| B.3 | Run#1-1 Data | A-78 |
| B.3.1 | Trace of <i>NLAG</i> 's Actual Run | A-78 |

| | | |
|----------|---|--------------|
| B.3.2 | Source Kernel | A-97 |
| B.3.3 | Conjectures Considered | A-98 |
| B.3.4 | All Analogical Inferences Found — Run#1-1 | A-103 |
| B.4 | Data from Other Runs | A-104 |
| B.4.1 | All Analogical Inferences Found — Run#1-2 | A-104 |
| B.4.2 | Relevant Data Added — Runs#2a- <i>x</i> | A-106 |
| B.4.3 | Irrelevant Abstractions Added — Run#2b-2 | A-107 |
| B.4.4 | Relevant Abstractions Added — Runs#2b-3 and #3- <i>xb</i> | A-112 |
| B.4.5 | Results Found, Using all Eight Relevant Abstractions | A-113 |
| C | Glossary | A-118 |
| C.1 | Notation and Conventions | A-118 |
| C.2 | Glossary of Terms | A-120 |

List of Tables

| | | |
|-----|--|-------|
| 4-1 | Other Examples of Lumped Element Linear Systems | 67 |
| 4-2 | Abstractions used for Useful Analogies | 69 |
| 5-1 | Synopsis of Heuristics | 104 |
| 7-1 | Overview of Tests Run | 139 |
| 7-2 | Generated and Accepted Values for Variables | 145 |
| 7-3 | Data for Run#1-1 (using Generator #1) | 146 |
| 7-4 | Data for Run#1-2 (using Generator #2) | 149 |
| 7-5 | Data for Run#2b-3 | 155 |
| 7-6 | Ablation of Analogy Data | 157 |
| 7-7 | Summary of Findings | 163 |
| 9-1 | Tension between Commonality and Applicability | 199 |
| A-1 | Classification of Unary Formulae | A-20 |
| B-1 | Deductions for <i>Resolve-Exists</i> and <i>Verify</i> , for the rkk Abstraction . . | A-95 |
| B-2 | Deductions for <i>Resolve-Exists</i> and <i>Verify</i> , for the other Abstractions | A-96 |
| B-3 | Deductions for <i>Resolve-Exists</i> and <i>Verify</i> , using Generator #2 | A-106 |
| B-4 | Number of Deductions for Run#2a-3 and #2a-4 | A-107 |
| B-5 | Answers Returned for Run#2a-4 | A-108 |

List of Figures

| | | |
|-----|---|-----|
| 1-1 | Simple Fluid Circuit | 8 |
| 1-2 | Solution to the “Find the flowrate” Problem | 9 |
| 2-1 | Definition of Analogical Inference | 16 |
| 2-2 | Contents of Initial Theory, Th_{CF} | 18 |
| 2-3 | Desired Analogy Formula and Instantiations | 19 |
| 2-4 | Analogy Questions | 29 |
| 2-5 | Number of Possible Analogies | 32 |
| 3-1 | Definition of <u>Useful</u> Analogical Inference | 39 |
| 3-2 | Over-Extended Analogy | 44 |
| 4-1 | Example of a Perspective: Electric Domain | 49 |
| 4-2 | Definition of the RKK-EC Perspective | 50 |
| 4-3 | Definition of the RKK Abstraction | 52 |
| 4-4 | Formulae, Relations and Abstractions | 55 |
| 4-5 | Abstraction-Based (Useful) Analogical Inference | 57 |
| 4-6 | Different Types of Analogical Inference | 58 |
| 4-7 | Components of Analogical Inference — Hydraulics Example | 61 |
| 4-8 | Example of Abstraction-Based Analogical Inference | 62 |
| 5-1 | Possible Worlds | 85 |
| 5-2 | NonOverlapping Possible Worlds | 89 |
| 5-3 | Constrained Possible Worlds | 91 |
| 5-4 | Possible Worlds: H_{FC} Demonstration | 95 |
| 5-5 | RKK’s Material Implication | 101 |
| 6-1 | Overall Behavior | 109 |

| | | |
|-----|--|------|
| 6-2 | Detailed Behavior of NLAG | 110 |
| 6-3 | Detailed Behavior of ComAbs | 113 |
| 8-1 | Semantic Definition of (General) Analogical Inference: \approx | 181 |
| 9-1 | Why $\text{Duke} \in \text{Dogs}$ is more specific than $\text{Duke} \in \text{Animals}$ | 191 |
| 9-2 | Commonality Intuition | 191 |
| 9-3 | How are Dams like Transistors? | 204 |
| 9-4 | Simple Pressure Regulator | 205 |
| A-1 | Desired Analogy Mapping, $\mathcal{M} = \langle S_{CF} P_C P_F \rangle$ | A-11 |
| A-2 | Number of Analogies | A-19 |
| A-3 | Definition of the RKK-Junc Abstraction | A-28 |
| A-4 | Rule Used to Solve A Group Problem | A-41 |
| A-5 | Definition of PlusAll | A-54 |
| A-6 | Faulty Definition of TimesAll | A-55 |
| A-7 | Definition of PlusAll-add1 | A-55 |
| A-8 | Definition of Plus-2-All | A-56 |

Abstract

The phenomenon of learning has intrigued scholars for ages; this fascination is reflected in Artificial Intelligence, which has always considered learning to be one of its major challenges. This dissertation provides a formal account of one mode of learning, learning by analogy. In particular, it defines the *useful analogical inference* process (UAI), which uses a given analogical hint of the form “A is like B” and a particular target problem to map known facts about B onto proposed conjectures about A. UAI only considers conjectures which are *useful* to the target problem; that is, the conjectures must lead to a plausible solution to that problem.

To construct a procedure which can effectively find these useful analogies, we use two sets of heuristics to refine the general UAI process. The first set is based on the intuition that useful analogies often correspond to “coherent” clusters of facts. This suggests that UAI seeks only the analogies which correspond to *common abstractions*, where abstractions are relations which encode solution methods to past problems. The other set of rules embody the claim that “better analogies impose fewer constraints on the world”. Basically, these rules prefer the analogies which require the fewest additional conjectures.

This dissertation also describes a running program, *NLAG*, which implements this model of analogy. It is then used in a battery of tests, designed to empirically validate our claim that UAI is an effective technique for acquiring new facts. This data also demonstrates that the heuristics are effective, and suggests why.

In summary, the primary contributions of this research are [1] a formal definition

of UAI, described semantically (using a new variant of Tarskian semantics), syntactically and operationally; [2] a collection of heuristics which efficiently guide this process towards useful analogies; and [3] various empirical results, which illustrate the source of power underlying this approach.

Chapter 1

Introduction

“Analogy pervades all our thinking, our everyday speech and our trivial conclusions as well as artistic ways of expression and the highest scientific achievements.”

How to Solve It, Polya (1957)

1.1 Motivation

Designing a system capable of learning new facts has long been a central theme for the general field of Artificial Intelligence; it has also been one of its greatest challenges [Fei63]. In the Expert Systems sub-area, Knowledge Acquisition — the task of expanding a system’s knowledge base — is considered the major problem of the 1980’s [FM83]. This dissertation discusses the task of learning by understanding an analogy: using information about some well understood source analogue as a framework for proposing new conjectures about a target concept.

Why use analogies? A common method for learning new facts is by explicit instruction from some teacher. However, acquiring new facts about a novel concept can be time consuming and laborious to both teacher and student. This is especially true when the new material is unfamiliar to the learner; this is always the case when the pupil is a young Expert System [BCEM78].

Fortunately, however, certain ideas pertain to more than one domain. This

means that knowledge can sometimes be re-used, that facts known about one concept can be used, *mutatis mutandis*, to understand another. The examples below illustrate this transference. In each case, facts known about the source analogue are used to suggest new conjectures about the target analogue.

- Electricity is often taught by showing the similarity of electrons flowing in resistors to water flowing in pipes (see Section 1.3).
- Matrix operations and abstract function spaces are often understood by relating them to more familiar concepts, like numbers (see Note:7-2 in SubAppendix A.7).
- The underlying structure of one computer subroutine can often be used to guide the construction of another. For example, a program which computes the product of the entries in an array can be used as a guide when designing a program to compute the sum of the members of a vector, even if the first is in PASCAL and the latter in LISP. (See Note:7-3 in SubAppendix A.7 and Note:9-6 in SubAppendix A.9, as well as [Der83] and [Ric81].)
- Computers are often anthropomorphized and people “computer-morphized”: e.g., “SUMEX doesn’t want to print my file” or “I was swapped out”. This understanding allows us to accurately predict the further behavior of the information processing system, be it human or machine; see [McC79] and [NS72].
- We think of happiness as “up”, as in “He was *up* in the clouds” or “I was *down* in the dumps.” This connection is sufficiently systematic that we can understand these and other related comments; see [LJ80], [CM83], [Hob83a] and [Red79].
- Many great scientists have used analogies to guide their work: Darwin understood natural selection by analogy to artificial selection [Dar58,Dar83]. Edison, too, was able to re-use results from one field in another — in particular, his first plans for the kinetoscope followed from his design of a phonograph system [Bro85].

- The similarities between the findings of cardiomyopathy and hypercalcemia can be used to predict the latter's anticipated drug toxicity with respect to the Digitalis treatment. ([Swa81] refers to this as the common *Anticipate Drug toxicity* domain principle.)
- Introductory computer programming texts often portray variables as boxes in which values can be stored (see [Bur84,AFB78]). Similarly, many people learn how to use a text-editor by thinking of it as a typewriter (see [Dou83]).
- Many frame-based representation systems allow the user to create a new unit by editing a copy of an existent unit. This is based on the assumption that many properties associated with the source unit will carry over to the target. (*Cf.*, [GL80,Len84]).
- Concepts with similar names often have analogous functions. For example, the observed symptoms associated with viral meningitis are quite similar to those of bacterial meningitis [Kun82, 273-74]. This is even more prominent in designed domains: I was able to use my knowledge of the Scribe text-formatting system [RW80] to correctly guess the names of many L^AT_EX operations [Lam84]. (*E.g.*, I guessed that L^AT_EX's `itemize` command is "`\begin{itemize}`", as Scribe's command is "`@begin{itemize}`".) The general observation is that many designed artifacts share obvious common properties with other objects built from the same plan; naming conventions are but one example of this phenomenon.

This list includes both interfield and intrafield connections (see [DM77]) and deals with both natural and artificial domains. Each entry is an analogy, or more precisely, an instance of learning by understanding an analogy. In each case, the instructor is exploiting the student's initial knowledge; using that body of facts, insights and intuitions pertaining to a source analogue to suggest plausible conjectures about the target.¹

¹Note:1-1 in SubAppendix A.1 lists yet other examples of analogy.

Roughly speaking, *analogical inference* refers to the process commonly known as learning by analogy. Stated more precisely, it is the process of proposing new facts about A based on existing facts about B and an analogical hint, “A is like B”. This dissertation defines and validates the thesis that:

Analogical inference, guided by common abstractions, is an effective mechanism for acquiring additional useful facts.

1.2 Outline

This section provides an outline for this dissertation, a guide for the casual reader and a brief description of the intended reader. Section 1.3 presents an example to make the notion of analogical inference more concrete. It implicitly describes the basic learning-by-analogy task and indicates the desired behavior of an analogical inference. Section 1.4 describes our goals and points out how it differs from much of the other work on analogy and learning.

The rest of this dissertation motivates, presents and validates a specific mechanism for performing analogical inference:

Chapter 2 provides the framework, by defining the terms “analogy” and “analogical inference”. It also illustrates the vast space of possible analogies. The rest of this dissertation can be viewed as a search through this space, one which seeks the “best” analogies first.

Chapter 3 focuses the search by imposing a *usefulness* constraint: *i.e.*, we want to consider only those analogical inferences whose conjectures help to solve specific problems. This leads to the *useful analogical inference process*. Section 3.4 informally discusses some intuitions about what should qualify as a good useful analogy. (Subsequent chapters convert these insights into heuristics which order the space of analogies. This ordering allows efficient generation of the analogies that suggest useful new facts.)

Chapter 4 continues the progression of refinements, defining the class of useful

analogies which correspond to “common abstractions” and describing the underlying process of *abstraction-based (useful) analogical inference*. These *abstractions* — abstract relations which each encode solutions to previous problems — realize one of Section 3.4’s intuitions: that an analogy should deal with a *closed* collection of facts. Section 4.2 states the fundamental claim of this research: that most useful analogical inferences reduce to finding a common abstraction which links source and target analogues. This chapter also shows *how* the abstraction-based analogical inference process works and motivates both why this process *should* be used (by showing that it can efficiently find useful analogies) and why it *can* be used (by showing that these needed abstractions do exist in a usable form). The rest of this dissertation deals with this refined model of analogy.

There can still be a large number of legal abstraction-based analogies. Chapter 5 discusses ways of ordering this search space, to rapidly find “better” analogies. In particular, Section 5.3 provides a cluster of related heuristics to guide the search to the “closest” analogy, that is, to the analogy which requires the fewest additional assumptions. (This “nearness” measure stems from another of Section 3.4’s intuitions.)

The next two chapters put these abstract ideas to work. Chapter 6 discusses how I implemented this model of analogy, describing the *NLAG*² system; Chapter 7 describes the experiments and demonstrations run using this system. This is the crux of my research, as it validates the thesis claim shown above. In addition to an empirical demonstration that this model of analogical inference is effective, it also includes an analysis which describes

- the underlying source of power of our abstraction-based approach — *i.e.*, why seeking common abstractions is effective. This analysis suggests the characteristics a domain and task must exhibit for this approach to be effective.
- why this “abstraction” label is not overly important. This means that our reliance on finding common *abstractions* does not constitute a major restriction.

²Pronounced en-el-a-gee, that is, “analogy”.

Chapter 8 presents a semantic account for analogies, providing semantic definitions which correspond to the syntactic ones given in Chapters 2 and 3. This framework is also used to formally explain and justify many of the heuristics presented in Chapters 4 and 5.

This closes the major new contributions of this research. The concluding chapters discuss the impact of this work in the larger picture. Chapter 9 discusses how my version of analogical inference differs from those of other researchers. Chapter 10 lists both future directions suggested by this research and its major contributions.

The various appendices cover tangential aspects of this research. Appendix A is a collection of long but indirectly relevant digressions, extending brief teasers which appear in the text. (Each pointer of the form “Note:i-j” refers to the “jth” note in SubAppendix A.i.) Appendix B elaborates Chapters 6 and 7, describing *NLAG*’s implementation in more detail and providing details of the actual experiments and demonstrations. Appendix C explicates my notational conventions and provides a quick glossary of the terms used in this dissertation.

As a final comment, this dissertation contains a fairly generous sprinkling of footnotes, each pursuing some theme considered tangential to the main point. Any focused reader who is annoyed by digressions is advised — indeed, urged — to ignore the footnotes whenever the main topic is understood.

Skimming Guide: The casual reader can acquire a good summary of this material by reading only the following portions:

- Chapter 1: describes the type of problem this research addresses;
- Sections 2.1 and 2.2: provide the general context by defining and demonstrating the basic analogical inference process;
- Section 3.4: suggests intuitive ways of focusing this search for useful analogies;
- Sections 4.1 and 4.4: motivate and justify our proposed solution, *viz.*, using abstraction-based analogical inference;
- Subsection 5.4.1: provides a quick synopsis of the heuristics which focus this analogical inference process;

- Section 7.2 and Subsection 7.6.4: demonstrate that this approach is effective and summarize the results of the various experiments; and
- Section 10.3: explicitly lists the major contributions of this research.

The other parts of this dissertation provide additional comments, justifications and extensions to this highlighted material. Most chapters, and many sections, begin with an intuitive description of its objective and explain how this objective pertains to our overall goals; these may help the reader decide whether to read the embellishment which follows. (Some of the more tangential and technical sections have been tagged accordingly, to allow the reader to skip those parts on the first reading.)

Our Model of the Reader: This dissertation in general, and Chapters 2 and 8 in particular, assumes that the reader is familiar with the basic concepts of logic and semantics. The unfamiliar reader is referred to the superb logic text [End72]. Much of Chapters 2 through 5 is worded in terms of search; this fundamental Artificial Intelligence paradigm is discussed at length in the Search chapter of “The Handbook of Artificial Intelligence” ([vG82]) and in [Nil80]. *N.b.*, while Appendix C does provide brief summaries of many of the terms, these definitions rely on the underlying concepts found in the texts.

1.3 Example

The Fluid Circuit diagram shown in Figure 1-1 describes a simple fluid dynamics problem: given the pipe characteristics C_a and C_b and total flowrate, Q_0 , determine the flowrate, Q_a , through the pipe P_a .

This problem is trivial to anyone who knows the first principles of fluid dynamics. But what if you didn't know what FlowRate was? ... and never thought to consider Ohm's or Kirchoff's Laws in this situation? That is, imagine all you knew about Fluid Systems were deductions you could make from the problem statement and Figure 1-1: e.g., that “pipes are objects”, and that “the FlowRate function assigns a number to each pipe”. You would clearly be unable to solve the problem.

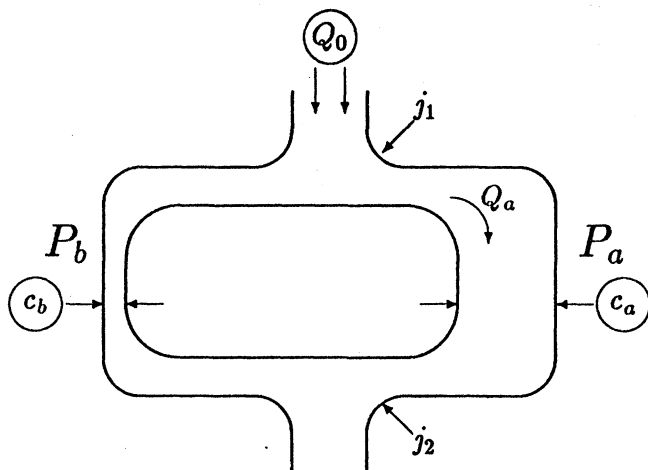


Figure 1-1: Simple Fluid Circuit

Now begins the learning process. You are told that

“FlowRate in a Fluid System is like Current in an Electric Circuit.” (1.1)

This hint tells the learner to apply his knowledge of Electric Circuits (EC) to help solve the problem. It suggests that a particular set of EC facts, known to the hearer and related to Current, can be used to generate a corresponding set of Fluid System propositions (FS) which, once incorporated, lead to a reasonable solution to this FS problem.

The intended interpretation of this hint is that Ohm’s and Kirchoff’s Laws do apply here, *mutatis mutandis*. That is, the learner should now conjecture that:

- the FlowRate into a junction must equal the FlowRate out, based on the knowledge that Current has that property (\approx Kirchoff’s Law #1);
- there is a VoltageDrop-like quantity associated with the junctions of a Fluid System whose algebraic sum around any closed loop is zero — which a fluid-dynamicist would call PressureDrop (\approx Kirchoff’s Law #2); and
- the VoltageDrop-like quantity [*viz.*, PressureDrop] across a Resistor-like devices, [*viz.*, Pipe] is proportional to the Current-like quantity [*viz.*, FlowRate]

entering it, using the device's Resistance-like term [*viz.*, PipeCharacter] as the constant of proportionality (\approx Ohm's Law).

| | |
|---|-------------------------------------|
| $\text{FlowRate}(j_1, P_a) + \text{FlowRate}(j_1, P_b) = Q_0$ | $\approx \text{Kirchoff's Law \#1}$ |
| $\text{PressureDrop}(j_1, j_2, [P_a]) = \text{PressureDrop}(j_1, j_2, [P_b])^3$ | $\approx \text{Kirchoff's Law \#2}$ |
| $\text{PressureDrop}(j_1, j_2, [P_a]) = \text{FlowRate}(j_1, P_a) * C_a$ | $\approx \text{Ohm's Law}$ |
| $\text{PressureDrop}(j_1, j_2, [P_b]) = \text{FlowRate}(j_1, P_b) * C_b$ | $\approx \text{Ohm's Law}$ |
| <hr/> | |
| $Q_a = \text{FlowRate}(j_1, P_a) = [C_b / (C_a + C_b)] * Q_0$ | |

Figure 1-2: Solution to the “Find the flowrate” Problem

Figure 1-2 shows that these proposed new facts do lead to an answer to this problem. This solution is reasonable, given the above hints — and it is an excellent approximation, at least in the case of laminar water flow (see [Coc80] and [Zie77]).⁴

We saw that the standard problem solver could not solve this “Find the flowrate” problem until the new FS facts had been added. *NLAG*'s mission is to propose this body of facts. However, the “FlowRate is like Current” hint may lead to many different sets of proposed conjectures. For example, given a sufficiently impoverished knowledge base, the *NLAG* system might incorrectly use each pipe's cross-section as the constant of proportionality in its hydraulics analogue to Ohm's Law; or even each pipe's cost. It might also try to map the capacity or inductance model from its original electricity context to this hydraulics domain.

In general, this hint allows us to infer that FlowRate satisfies some known property of Current. Hence, it might suggest that that FlowRate's material (corresponding to Current's charge) can be stored in a capacitor-like device, that FlowRate obeys Faraday's Law, that the greater the FlowRate through a device, the more heat is produced, or even that the modern study of FlowRate began with William Gilbert's early treatise [Rob85].

³The arguments to this ternary PressureDrop function are a pair of junctions and a path between them. Of course, *Kirchoff's Law #2* means that this third argument is superfluous. See Subsection 9.3.2.

⁴Note:1-2 both describes the “more correct” answer and discusses why I am not concerned that it differs from this solution.

Of course, these variants will suggest different answers to the original problem; and some would not lead to any solution. The interesting question — and main topic of this research — is how the “correct” conjectures can be found efficiently. Ideally, we would like *NLAG* to find that first set of conjectures (which included Ohm’s Law and PipeCharacter), and find it before any of the other possible conjectures — *i.e.*, before the ones which dealt with capacity or Faraday’s Law. Jumping up a level, my research goal is to determine how to use the analogical hint (Equation 1.1) and the target problem (“Find the flowrate”) to find the appropriate set of conjectures, first.

This hydraulics and electricity example plays a central rôle in this dissertation; it is rich enough to illustrate essentially all of the relevant situations and rules. In particular, more specific descriptions of the relevant analogical inferences appear in Sections 2.2 and 3.3. A detailed trace appears in Chapter 7, which is further elaborated in Appendix B.

Of course, the *NLAG* system is not limited only to these electricity-to-hydraulics analogies; it is a general domain-independent learning-by-analogy system. Various notes sketch examples of its operation in other fields: Note:7-2 describes how this analogy process can use facts about one algebraic operator to learn about another; and Note:7-3 and Note:9-6 describe how *NLAG* can be applied to the domain of programming.

1.4 Objectives

“Analogies prove nothing, that is quite true, but they can make one feel more at home.”

New Introductory Lectures on Psychoanalysis, Freud (1932)

Like many other learn-by-analogy systems, *NLAG* uses facts known about one analogue to propose new base-level facts about the other. Most other systems, however, try to find the *maximal similarity* which connects the pair of analogues (*cf.*, [HIM78], [Win82], [Gen80b], [Hob83a] and [DM77]). My focus, instead, is on finding *useful* new facts about the target analogue. This means that *NLAG* seeks analogies which

suggest new target domain facts when those new facts are likely to help solve a range of standard problems.

Using Section 1.3's example, many other analogy systems would try to find all the ways in which FlowRate is like Current. (*E.g.*, they might propose *all* of the suggested analogies shown above.) NLAG, on the other hand, is interested only in connections which (are likely to) lead to the new information needed to find a solution to this particular hydraulics problem. (Section 3.4 further describes this dichotomy; and Subsection 9.2.3 argues that this other "find the maximal commonality" approach can be both inefficient and counter-intuitive. Section 9.4 supplies specific pointers to such "maximal commonality" systems.)

NLAG applies a model-based approach to this task, using *known generalizations* (called "abstractions") to guide the search [Pol54]. While many other systems implicitly address the task of acquiring new generalizations or modifying existing ones, my interest is focused on demonstrating that such generalizations can be used effectively to propose new conjectures about the target analogue.

Finally, I want to emphasize that we are describing a method of *plausible reasoning*. While the new facts this NLAG system proposes are guaranteed to be reasonable, they may not be semantically correct. This is discussed further in Note:1-2.

Chapter 2

General Analogical Inference

This chapter lays the foundation for this dissertation by defining the *general analogical inference process*, commonly called “learning by analogy”. Section 2.1 presents a formal definition of this process, and Section 2.2, an example of its behavior.

To simplify the initial presentation, these first sections have suppressed certain tangential details. The final sections present three of these elaborations. Section 2.3 describes the analogy relation and ties this definition to analogical inference. Section 2.4 shows how this analogy formalism eliminates trivial analogies. Finally, Section 2.5 discusses the complexities inherent in the process of finding analogical inferences, providing first an informal synopsis followed by a quantitative description which demonstrates the exponential size of the analogy space. Other interesting digressions appear as notes in SubAppendix A.2. Readers may wish to skip these points on their first readings.

Various subsequent chapters extend and complement these ideas. Chapter 3 refines the general definition of analogical inference to consider only the analogies which are useful to some given problem. Chapter 8 provides a semantic account of this analogical inference process, corresponding to these syntactic definitions. Finally, this model of analogy is slightly different from many of the more standard descriptions. A comparison appears in Chapter 9, which also provides a justification for my approach.

2.1 Definition of Analogical Inference

a·nal·o·gy ə·nal'ə·jē ... 3. *Logic* [An] inference that items showing some resemblances will show others [as well].

Funk & Wagnalls Standard Dictionary (1980)

Intuitively, an analogical inference postulates *new* facts about a target concept based on existing facts known about a source concept. In particular, it uses an analogical hint “A is like B” and known facts about B to postulate plausible but underivable facts about A. For example, we used the hint “FlowRate is like Current” and the known fact, “Kirchoff1(Current)”, to postulate “Kirchoff1(FlowRate)”.¹

This section provides a formal definition of general analogical inference. For pedagogical reasons, it begins with the simplified version shown in Equation 2.1 below and works up to the full definition shown in Figure 2-1.

$$\begin{array}{l}
 Th, A \sim B \vdash \varphi(A) \\
 \text{where Common: } Th \models \varphi(B) \\
 \text{Unknown: } Th \not\models \varphi(A) \\
 \text{Consistent: } Th \not\models \neg\varphi(A)
 \end{array} \tag{2.1}$$

The *analogical inference* operator, \vdash , uses a theory² Th and a statement of the form $A \sim B$ to *analogically infer* a new fact about the target analogue, A. Here φ is a formula which is known to hold for the source analogue B; this is encoded by the **Common** condition. The other two requirements (**Unknown** and **Consistent**) mean that an analogical inference only postulates those $\varphi(A)$ which are independent of the starting theory, Th ; hence \vdash does not accept any sentence which is already derivable nor any which would lead to an inconsistent theory.

(These latter two conditions, **Unknown** and **Consistent**, alone provide a good definition of learning in general. This reflects the view that analogical inference is a learning process, constrained only by the requirement that the result be an analogy.)

¹By convention, base level facts --- i.e., symbols the user can type and see --- are in this fixed-width font. A complete summary of notation appears in SubAppendix C.1.

²A theory is a deductively closed, consistent collection of axioms.

Putting all three conditions together, we see that *an analogical inference is an inference which consistently extends a theory by adding new (that is, undeducible) facts about a target analogue, based on provable facts referring to the source analogue.*

While this notion of analogical inference is general, it is not vacuous. The two conditions

$$\text{Common: } Th \models \varphi(B) \tag{2.2}$$

$$\text{Unknown: } Th \not\models \varphi(A)$$

alone eliminate many undesirable formulae, including

$$\varphi(x) = x \neq x \tag{2.3}$$

or

$$\varphi(x) = x = x \tag{2.4}$$

or

$$\varphi(x) = \text{Tall}(\text{Fred}) \tag{2.5}$$

or

$$\varphi(x) = \exists y x \in y \tag{2.6}$$

Case 2.3 would lead to the obviously “incorrect” fact “ $A \neq A$ ”; and the other three, to trivial analogies: Case 2.4 is a tautology, and the formula in case 2.5 does not lexically include its parameter. For case 2.6, we assume that every concept is a member of some class — i.e., $Th \models \forall x \exists y x \in y$. This means that the facts “ $\text{Fido} \in \text{Dogs}$ ” and “ $\text{Democracy} \in \text{Ideas}$ ” do not establish an analogical connection linking Fido to Democracy.

There are also $\langle A B Th \rangle$ triples from which no analogical inferences can be made. An extreme example is the empty theory, $Th = \{\}$. Here, no formula can satisfy the three conditions specified in Equation 2.1. (In fact, no formula can satisfy even Equation 2.2’s two requirements.) More generally, no analogy formula is possible if none of Th ’s sentences lexically include B. As a different example, imagine that everything known about A is directly contradicted by facts initially known about B, e.g., $Th = \{\text{foo}(A), \neg\text{foo}(B)\}$. Here, any formula φ which passes Equation 2.1’s first condition (Common) will fail its third (Consistent).

As stated above, this formalization of analogical inference is not comprehensive; we now motivate and supply two extensions. First, we need to deal with n -ary formulae, as opposed to the unary formulae show in Equation 2.1. Given that we want to infer

Ohms(FlowRate, PressureDrop, PipeCharacter, Pipes)
 from Ohms(Current, VoltageDrop, Resistance, Resistors),

we need to deal with “Skolem variables” — such as VoltageDrop’s dependency on Current. Equation 2.1’s unary φ formula is not enough for these cases; we need to deal with n -ary formulae. Figure 2-1 shows the revised definition. Notice it uses two sets of constant terms $\{a_j\}$ and $\{?b_j\}$.

(Before delving into the meaning of this definition, we want to clarify issues about the form of the statement. First, there is an implicit universal quantifier around the definition; i.e., it should read “ $\forall A, B, Th, \{a_j\}. \dots$ ”. The ? prefix of the $?b_j$ terms indicates that they are existential variables. They are embedded within the outer universal quantifier; in fact, another way of reading the Common condition is

$$\text{Common: } Th \models \varphi|_i(B).$$

(This uses the projection operator, $|_i$, to isolate the formula’s i^{th} argument; i.e.,

$$\varphi|_i(y) \iff \exists x_{j \neq i} \varphi(x_1, \dots, y, \dots, x_n) \tag{2.7}$$

refers to the projection of the i^{th} argument of the n -ary formula φ .) Figure 2-1 shows the actual existential variable, $?b_j$, to explicate the connection between $a_j \mapsto ?b_j$. To allay a third possible confusion: each of the terms used to instantiate the \vdash equation — e.g., the term used for A — is a constant. In our context, some of these constants are relation symbols, e.g., Current. Now, back to the plot...)

This embellishment forces the second change: the inclusion of an explicit Non-Trivial condition, already included in Figure 2-1. To understand its purpose, imagine we are told that Democracy \sim Fido” and knew that Fido \in Dogs. Should we be allowed to analogically infer that Democracy \in Ideas, based on the common formula, $\varphi_e(x, y) = x \in y$? While this is clearly a meaningless and unmotivated conjecture, none of \vdash ’s other requirements prevent it.

| |
|---|
| $Th, A \sim B \vdash \varphi(a_1, \dots, A, \dots, a_n)$ <p style="margin-left: 20px;"> where Common: $Th \models \varphi(?b_1, \dots, B, \dots, ?b_n)$ Unknown: $Th \not\models \varphi(a_1, \dots, A, \dots, a_n)$ Consistent: $Th \not\models \neg\varphi(a_1, \dots, A, \dots, a_n)$ NonTrivial: $\neg Trivial(\varphi _i, Th)$ </p> |
|---|

Figure 2-1: Definition of Analogical Inference

How can we thwart such inferences? Consider the domain of φ_e 's first argument: this is the set of values which could instantiate x in the formula " $x \in y$ " for some value of y . As every concept is a member of some set, this domain is universal. The observation that both Fido and Democracy qualify is meaningless, since any other concept could have qualified as well. We consider a formula to be "trivial" if its domain is universal; and any analogy based on a trivial formula, uninteresting, as it says nothing about the target analogue.

The **NonTrivial** requirement prevents such analogies. Repeating the above arguments, a formula is trivial in its i^{th} argument if any value could legally appear there. The formal definition is

Definition 1 $Trivial(\varphi|_i, Th) \iff \forall s \exists x_{j \neq i} Th \models \varphi(x_1, \dots, s, \dots, x_n)$.

Section 2.4 further justifies this definition.

This concludes the definition of general analogical inference. We end this section with some final notes. The next section provides an example of this inference process, based on the hydraulics and electricity situation shown in Section 1.3. The following chapter defines a restricted sense of analogical inference, one whose goal is finding only useful analogies.

Defn1. For notion, we refer to this φ as the "analogy formula", and to $\varphi(b_1, \dots, B, \dots, b_n)$ as the "source analogy sentence" or the "instantiation (of φ) in the source domain". Likewise, $\varphi(a_1, \dots, A, \dots, a_n)$ represents the "target analogy sentence" or

the “instantiation (of φ) in the target domain”. We occasionally speak of the target analogy sentence as the “analogy”.

Defn2. This \vdash formulation allows the common formula to be any arbitrary combination of clauses. Note:2-1 discusses why this \vdash is not limited to simple conjuncts of positive literals, but allows disjunctions and negations as well.

Defn3. Page 13 suggested that “ $\text{Analogy}_x = \text{Learning}_x + \text{Common}$ ”. This idea also holds for the other versions of analogy presented in this dissertation; *cf.*, Figures 2-1, 3-1 and 4-5.

Defn4. The observant reader may suggest that we use the unary $\varphi|_i$ formula throughout, rather than the full n -ary φ formula. This would eliminate the need for Figure 2-1’s extended definition, and allow us to use the simpler Equation 2.1. Note:2-3 explains why this is inadequate, why analogical inference has to contend with these full n -ary formulae. That note also illustrates why the **Unknown** condition does not subsume the **NonTrivial** condition.

2.2 Example of Analogical Inference

The previous section presented a formal definition of analogical inference. To make that definition more concrete, this section presents an example of its behavior, using the example of Section 1.3.

As input, we are given the analogical hint “**FlowRate** \sim **Current**” and the target problem, “Find the flowrate”. We also have access to the theory Th_{CF} , which includes the initial collection of facts known about **Current** and **FlowRate**. Figure 2-2 shows its relevant contents, using ellipses to avoid enumerating the huge collection of irrelevant facts. (Notice it does not include either $\text{Kirchoff2}(\text{PressureDrop})$ nor $\text{Ohms}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes})$; these omissions will be important later.)

Our task is to find a formula and instantiation which satisfies the constraints

```

Kirchoff1(Current),
Kirchoff2(VoltageDrop),
ConservedThru(Current, Resistors),
Ohms(Current, VoltageDrop, Resistance, Resistors), ...
CostPerHour(Current, $2.43),
Units(Current, Amperes),
Cost(Resistor1, $3.50),
Color(Wire1, Red),
ModernTreatise(Current, wGilbert, 1600),
InDiscipline(Current, Electric), ...

Kirchoff1(FlowRate),
ConservedThru(FlowRate, Pipes),
Units(FlowRate, Meter3PerSec), ...
Cost(Pipe1, $2.73),
Color(Pipe1, Red),
InDiscipline(FlowRate, Hydraulics), ...

Domain(Kirchoff1, 1, Functions),
Domain(Kirchoff2, 1, Functions),
Domain(Ohms, 1, Functions),
Domain(Ohms, 4, Classes), ...

 $\forall t$  Kirchoff1(t)  $\Leftrightarrow \forall j \sum_{p:Conn(p,j)} t(j,p) = 0$ ,
 $\forall c$  Kirchoff2(c)  $\Leftrightarrow \forall loop \sum_{\langle i,j \rangle \in loop} c(i,j,[x]) = 0$ ,
 $\forall c, l$ . ConservedThru(t, l)  $\Leftrightarrow \forall d \ 1(d) \Rightarrow [t(j_d^1, d) + t(j_d^2, d) = 0]$ ,
 $\forall t, c, r, l$ . Ohms(t, c, r, l)  $\Leftrightarrow [\forall d \ 1(d) \Rightarrow [c(j_d^1, j_d^2, [d]) = t(j_d^1, d) * r(d)]]$ 
...

```

Figure 2-2: Contents of Initial Theory, Th_{CF}

(The notation " j_r^1 " refers to the first junction associated with the resistor r . In general, " j_d^i " refers to the i^{th} junction associated with the device d . Also, "[d]" refers to the path which goes through device d .)

$$\varphi_{RKK}(t, c, r, l) = \left\{ \begin{array}{l} \text{Kirchoff1}(t) \\ \& \text{Kirchoff2}(c) \\ \& \text{ConservedThru}(t, l) \\ \& \text{Ohms}(t, c, r, l) \end{array} \right\}$$

Target Instantiation = [FlowRate, PressureDrop, PipeCharacter, Pipes]

Source Instantiation = [Current, VoltageDrop, Resistance, Resistors]

Figure 2-3: Desired Analogy Formula and Instantiations

shown in Figure 2-1. The “correct” answer is the formula φ_{RKK} ³ and target instantiation [FlowRate, PressureDrop, PipeCharacter, Pipes], both shown in Figure 2-3. (That figure also shows the source instantiation, [Current, VoltageDrop, Resistance, Resistors], which is an intermediate, internal result.)

To verify that this qualifies as a legal analogical inference, we have to confirm the following conditions:

$$\begin{array}{ll} \text{Common:} & Th_{CF} \models \varphi_{RKK}(\text{Current, VoltageDrop, Resistance, Resistors}) \\ \text{Unknown:} & Th_{CF} \not\models \varphi_{RKK}(\text{FlowRate, PressureDrop, PipeCharacter, Pipes}) \\ \text{Consistent:} & Th_{CF} \not\models \neg\varphi_{RKK}(\text{FlowRate, PressureDrop, PipeCharacter, Pipes}) \\ \text{NonTrivial:} & \neg Trivial(\varphi_{RKK}|_1, Th_{CF}) \end{array} \tag{2.8}$$

The first condition is easy to confirm. The second and third follow from Th_{CF} 's ignorance, that it knows sufficiently little about FlowRate that this φ_{RKK} instantiation is independent of that starting theory. (This involves the claim that Figure 2-2's “...”s do not hide anything important.) The NonTriviality condition follows from the Domain(Kirchoff1, 1, Functions) statement,⁴ which implies that only

³Chapter 4 explains why we chose this particular φ_{RKK} formula. In particular, Figure 4-3 in Section 4.1 shows this formula reified as the RKK relation.

⁴This is a second-order fact -- i.e., a fact about a relation. While this could have been worded in

functions can satisfy $\varphi_{\mathbf{RKK}}|_1$. (Left implicit is the statement that the full universe contains objects besides functions.)

This demonstrates that the particular formula and target instantiation shown in Figure 2-3 form a legal analogical inference.

Of course, this answer is not unique: there are many both other target instantiations for this formula, and other legal analogy formulae. Using this same $\varphi_{\mathbf{RKK}}$ formula, the target instantiation [FlowRate, PressureDrop, CrossSection, Pipes] also qualifies, as does [FlowRate, VoltageDrop, PipeCharacter, Pipes]. While neither of these is correct, none of \sim 's conditions prevents either conjecture from being analogically inferred. (In particular, each constitutes a consistent extension to the theory Th_{CF} .)

There are many other target instantiations which violate some of Figure 2-1's criteria. For example, [FlowRate, PressureDrop, PipeCharacter, Cost] is illegal, as it violates the Consistent condition. (This uses Th_{CF} 's fact, Domain(Ohms, 4, Classes), together with the implicit $Cost \in Functions$ assertion.)

There are also a vast range of other legal analogical formulae. For example, we can consider a formula which is more specific⁵ than $\varphi_{\mathbf{RKK}}$, such as

$$\varphi_{\text{More}}(t, c, r, l, m) \equiv \varphi_{\mathbf{RKK}}(t, c, r, l) \ \& \ \text{CostPerHour}(t, m), \quad (2.9)$$

or less specific, such as

$$\varphi_{\text{Min}}(t, c, r, l) \equiv \text{Kirchoff2}(c) \ \& \ \text{Ohms}(t, c, r, l) \quad (2.10)$$

$$\varphi_{\text{Ohm}}(t, c, r, l) \equiv \text{Ohms}(t, c, r, l). \quad (2.11)$$

Other possible analogy formulae are less related to $\varphi_{\mathbf{RKK}}$, e.g.,

$$\varphi_{\text{Cost}}(t, m) \equiv \text{CostPerHour}(t, m). \quad (2.12)$$

On the previous page, we saw that only some instantiations of an analogy formula lead to legal analogies. Similarly, only some formulae qualify as analogy formulae. As before, the facts in the initial knowledge base eliminate many candidates.

a first-order framework (i.e., $\forall x \text{Kirchoff1}(x) \Rightarrow x \in Functions$), this seemed pointless as the arguments themselves (e.g., FlowRate) are already relations.

⁵The formula $\varphi_1(x)$ is *more specific than* $\varphi_2(x)$ if $\forall x \varphi_1(x) \Rightarrow \varphi_2(x)$. Here, we also say that $\varphi_2(x)$ is *more general than* $\varphi_1(x)$.

None of the following formulae are legal analogy formulae:

$$\varphi_{K_2}(x) \equiv \text{Kirchoff2}(x) \quad (2.13)$$

$$\varphi_G(x) \equiv \text{Group}(\mathcal{R}, x, 0) \quad (2.14)$$

$$\varphi_{\text{Units}}(x) \equiv \text{Units}(x, \text{Amps}) \quad (2.15)$$

$$\varphi_{\text{Taut}}(x) \equiv x = x \quad (2.16)$$

$$\varphi_{\text{Triv}}(x, y) \equiv x \in y \quad (2.17)$$

The first two cases violate the **Common** condition,

$$Th_{CF} \not\models \varphi_i(\text{Current}).$$

The third case violates the **Consistent** criterion,

$$Th_{CF} \models \neg \varphi_{\text{Units}}(\text{FlowRate}).$$

The fourth case, φ_{Taut} , violates both the **Unknown** and **NonTrivial** conditions; the final φ_{Triv} , only the **NonTrivial** condition.

This verification process suggests an algorithm for implementing Figure 2-1's analogical inference process. Namely, it is sufficient to generate every possible formula and every instantiation, and then test each against these requirements. These examples, however, suggest the vast number of legal analogies; Section 2.5 demonstrates it is exponential in size. Thus, while this approach may be epistemologically adequate, it is heuristically implausible as this task is inherently intractable [MH69]. Section 3.3 presents a stronger argument against this blind generate-and-test approach, by first explaining that we are searching for *useful* analogies and then showing that few of the legal analogy formulae and instantiations qualify. The rest of this dissertation provides a better approach to this problem.

2.3 Definition of Analogy Relationship⁶

The previous sections defined the analogical inference process. This section describes the $Analogy_F$ relation, which defines what it means to claim that two concepts are analogous. It then discusses how $Analogy_F$ relates to the analogical inference process, \vdash .

There are two main reasons for defining the analogy relation. First, making explicit the distinction between the dynamic analogical inference process and the static analogy relation may clarify some of the existing confusions concerning the nature of analogy. Secondly, this definition facilitates the presentation of many of the subsequent definitions; cf. Definitions 12 and 27.

Analogy Relation: Loosely stated, two concepts are analogous if these each satisfies the same formula. This suggests the $Analogy_F$ definition of analogy, in which A is deemed analogous to B if each appears in some instantiation of the same non-trivial formula, φ :

Definition 2 $Analogy_F(A, B, Th, \langle \varphi [a_1, \dots, A, \dots, a_n] [b_1, \dots, B, \dots, b_n] \rangle)$
 $\iff Th \models \varphi(a_1, \dots, A, \dots, a_n) \ \& \ Th \models \varphi(b_1, \dots, B, \dots, b_n)$
 $\ \ \ \ \ \& \ \neg Trivial(\varphi|_i, Th)$

where the analogues $A = a_i$ and $B = b_i$ for the same i used in $\varphi|_i$. The $Analogy_F$ -encoding of the analogy given in Section 1.3 is shown in Figure 2-3. (This formulation differs from the one implied by Section 1.3's description. Note:2-4 describes that alternate version of analogy, $Analogy_T$.)

Connection between Analogy and Analogical Inference: How is the four-place $Analogy_F$ relation described above related to the \vdash rule of inference mentioned in Section 2.1? Intuitively, an analogical inference *completes an analogy* by adding the conjectures needed to establish an analogical connection between the given analogues. Hence, the $Analogy_F$ relation uses the result of the analogical inference,

⁶The rest of this chapter can be skipped on the first reading.

which is the embellished theory

$$Th' = Th + \varphi(a_1, \dots, A, \dots, a_n).^7$$

Excluding certain pathological cases,⁸ this φ establishes a legal analogical connection between A and B in this newly formed theory; i.e.,

$$Analogy_F(A, B, Th', \langle \varphi [a_1, \dots, B, \dots, a_n] [b_1, \dots, B, \dots, b_n] \rangle). \quad (2.18)$$

Summary: To recapitulate the definitions presented so far: This section has defined the static $Analogy_F$ relationship, which describes when two concepts are analogous. By contrast, Section 2.1 discussed the dynamic process of analogical inference, describing a learning process in which facts were added to a growing theory. The next chapters present various refinements of this analogical inference process.

2.4 Triviality and Aboutness

The purpose of the $\neg Trivial$ clause in Definition 2 is to prevent trivial analogies. The analogical inference process has similar reasons for using this same restriction — encoded as the $NonTrivial$ constraint in Figure 2-1. (There, this constraint augments \vdash 's other independence conditions, *viz.*, $Unknown$ and $Consistent$. While those restrictions do prevent many trivial would-be analogies, they are not sufficient.) This section formalizes this notion of triviality. In particular, it describes the $Trivial$ relation, which defines when a unary formula is trivial with respect to a theory.

If we remove this $\neg Trivial$ condition from Definition 2, nothing prevents $Analogy_F$ from using $\varphi(x) = \top^9$ and thereby finding any pair of concepts trivially analogous.

⁷The notation $Th + \sigma$ abbreviates the larger theory, $Th \cup \{\sigma\}$.

⁸See Points TR4 and TR5 in Section 2.4.

⁹This \top denotes “the true” [Thi68, p90]. Alternatively, we can view it as any ground tautology, e.g., “ $0 = 0$ ”. Also, this section uses the syntax $\rho(Z) = \sigma$ to mean that the formula ρ “expands” into the sentence σ when the term Z is substituted for ρ 's free variable. This means that

We avoid such embarrassments by insisting that $\varphi(b_1, \dots, B, \dots, b_n)$ be not just provable, but also *non-trivially about B*. This section leads up to the correct definition of triviality by presenting two intuitive but inadequate strawmen, *LexTrivial* and *TrivialAll*.

One way of avoiding the above $\varphi(\dots B \dots) = \varphi(\dots A \dots)$ degeneracy is to insist that φ deals with its arguments. That is, we want to forbid cases like $\varphi(x) = \text{Tall}(\text{Fred})$. The *LexTrivial* relation states this proposal more formally:

Definition 3 $\neg \text{LexTrivial}(\varphi, Th) \iff [\forall X \text{ LexInclusion}(X, \varphi(X))]$

where *LexInclusion*(X, σ) means that the symbol X is lexically included in the sentence σ . (For example, *LexInclusion*($X, \text{"Foo}(X)$ ") and $\neg \text{LexInclusion}(X, \text{"Foo}(B)$ "). Notice this means that the universal quantifier in Definition 3 is ranging over symbols; that is, the universe of discourse here is the language itself.

Unfortunately, Definition 3's criterion is too weak. While it does prevent some undesirable formulae (by insisting that $\varphi(B) \neq \varphi(A)$), it still allows other vacuous analogies to slip through. Consider any tautology with free variables, e.g., $\phi(x) = x = x$. While $\neg \text{LexTrivial}(\phi, Th)$, no analogy based on this ϕ will be very useful.

As a second suggestion, perhaps we should use some proof-based notion of triviality. We might consider a sentence, $\sigma = \varphi(y_1, \dots, x, \dots, y_n)$, to be trivial with respect to the concept x if that same fact could pertain to *any* other concept as well. That is, σ is non-trivial for some concept x if it does not hold for some other concept, *mutatis mutandis*.

Definition 4 $\neg \text{TrivialAll}(x, \varphi(y_1, \dots, x, \dots, y_n), Th) \iff \exists s Th \not\models \varphi(y_1, \dots, s, \dots, y_n)^{10}$

While this is better than *LexTrivial*, it is still not enough. The problem is Skolem constants. Recall again the formula $\varphi_\epsilon(x, y) = x \in y$, and assume we knew

$\rho(Z) = \text{Tall}(Z) \ \& \ \text{Green}(Z)$ if the formula ρ was defined as $\forall x \rho(x) \equiv \text{Tall}(x) \ \& \ \text{Green}(x)$. This use of "=" is distinguished from the more general use of " \equiv ", which denotes logical equivalence. Hence, using the same ρ defined above, we would have $\rho(Z) \equiv \text{Green}(Z) \ \& \ \text{Tall}(Z)$ but $\rho(Z) \neq \text{Green}(Z) \ \& \ \text{Tall}(Z)$. This distinction is only important for this section.

¹⁰This *TrivialAll* relation does use a different set of arguments than the eventual *Trivial* relation employs.

that $\forall x \exists y x \in y$. Could this formula serve as the common analogy formula? For example, are the facts $\text{Fido} \in \text{Dogs}$ and $\text{Democracy} \in \text{Ideas}$ sufficient to consider Fido analogous to Democracy ?¹¹ This issue reduces to the related question, is $\text{Fido} \in \text{Dogs}$ about Fido ?

The answer to both questions is “no”: Fido ’s occurrence in $\text{Fido} \in \text{Dogs}$ is trivial. Why? We could have used *any* other constant in Fido ’s place, knowing that the needed second argument (here, Dogs) is guaranteed to exist. Stated another way, $\text{Fido} \in \text{Dogs}$ does not say anything about Fido as any value could be used in Fido ’s place. This observation fits our expectations for analogies: the fact that both $\text{Fido} \in \text{Dogs}$ and $\text{Democracy} \in \text{Ideas}$ hold does not express any commonality joining Fido and Democracy . This is why it should not constitute a legal analogy.

Unfortunately, even though we argued above that $\text{Fido} \in \text{Dogs}$ should be considered trivial in this situation, *TrivialAll* reports otherwise; i.e., *TrivialAll*(Fido , $\text{Fido} \in \text{Dogs}$, Th) does *not* hold.

The fault does not lie with the \in relation itself. Consider, for example, the pair of assertions “ $\text{Fido} \in \text{Dogs}$ ” and “ $\text{Duke} \in \text{Dogs}$ ”, based on the common formula $\varphi_D(x) \equiv \varphi_\epsilon(x, \text{Dogs}) \equiv x \in \text{Dogs}$. These two assertions, together, do seem significant; i.e., φ_{Dogs} is not trivial in its first argument, even though φ_ϵ is. The difference is in terms of what is allowed to vary from one analogue to the other. When using the φ_ϵ relation, ϵ ’s second argument could vary and can therefore be set to different values depending on the first argument — it was Dogs for Fido but Ideas for Democracy . On the other hand, this “virtual” second argument is fixed when considering the φ_{Dogs} analogy formula: it is forced to be Dogs independent of the first argument. This means that triviality must be with respect to the set of terms which are allowed to vary from one analogue to the other.

This leads to the correct definition of triviality: an n -ary formula is trivial in its i^{th} argument if we could prove that any concept could occupy that position.

¹¹Yes, ϵ ’s second argument did vary, from Dogs to Ideas . This does not render this question meaningless. For example, that the two facts $\text{Ohms}(\text{Current}, \text{VoltageDrop}, \text{Resistance}, \text{Resistors})$ and $\text{Ohms}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes})$ are sufficient to link FlowRate to Current .

Definition 5 $Trivial(\varphi|_i, Th)$

$$\iff \forall s \ Th \models \varphi|_i(s)$$

$$\iff \forall s \ Th \models \exists x_{j \neq i} \varphi(x_1, \dots, s, \dots, x_n)$$

As expected, this $Trivial(\varphi, Th)$ relation is equivalent to $TrivialAll(x, \varphi(x), Th)$ when φ is unary (i.e., when there are no other arguments to vary). Notice also that this definition is syntactic. Chapter 8 presents a corresponding semantic version of this triviality condition; see Definition 26.

We conclude this section with some final comments.

TR1. Limitations of Triviality:

While this *Trivial* relation does provide a technical sense of triviality, it does not capture the full intuitive sense of “uninteresting”. Consider, for example, the non-trivial, but not interesting $\varphi(x) = x \neq \text{Fred}$. Subsection 9.2.2 extends this binary triviality criterion to a *specificity measure* which gives this uninteresting formula an appropriately low score. (By *extending*, we mean that this *Trivial* criterion is the extreme point of the specificity continuum.)

TR2. Ties to Aboutness:

We can use this *Triviality* idea to define the related notion of aboutness. A sentence $\varphi(x)$ is *about* a concept x , with respect to a theory Th , if

Definition 6 $About(x, \varphi(x), Th) \iff [Th \models \varphi(x)] \ \& \ [\exists s \ Th \not\models \varphi(s)].$

The second conjunct on the right is the non-triviality condition, $\neg Trivial(\varphi, Th)$. The extension to n -ary formulae is obvious.

This aboutness condition ties in with several other ideas: First, it is related to Hempel’s definition of the essential use of a constant in a sentence, see [Hem65, p38]. (Glymour uses this in his implementation; see [Gly80, Gly83].) Secondly, realize the underlying purpose of this aboutness condition is to determine the

relevance of a given proposition. This seems similar to Carnap's distinction between the three types of sentences: object, pseudo-object and syntactic [Mar67, p50]. Third, Note:8-1 presents the related concept of novelty, first defined seen in [GG83]. (That paper also discusses the relation between aboutness and novelty.)

TR3. Applicability of *Triviality* condition:

Two comments: First, this *Triviality* condition holds in any situation where the domain of the i^{th} argument of some relation is the entire allowable space. Hence, it pertains to the arguments of any total function and to the range of any onto function.

Secondly, this triviality condition can selectively pertain to some arguments of a relation but not to others. That is, $Trivial(\varphi|_i, Th)$ can be independent of $Trivial(\varphi|_j, Th)$. For example, the φ_ϵ relation can still express a non-trivial fact: even though $D \in S_D$ is trivial in its first argument, D , it is non-trivial in its second. This is because this second argument can include only certain concepts, *viz.*, those which denote some class of objects.

As another way to think about this: consider how much the single statement $D \in S_D$ tells us about its arguments. While we still know absolutely nothing about its first argument, D , we now know that S_D must denote some class. The realization that $D \in S_D$ places a constraint on its second argument (here, that it denotes some class), means that this relation, φ_ϵ , is not trivial in its argument.) This is why $Trivial(\varphi_\epsilon|_1, Th)$ and $\neg Trivial(\varphi_\epsilon|_2, Th)$.

TR4. $\neg Trivial(\phi, Th) \not\Rightarrow \neg Trivial(\phi, Th')$:

As suggested on page 23, the result of analogical inference might not be a legal analogy. As an example, imagine that the initial theory Th included a pathological clause of the form $\phi(A) \Rightarrow [\forall x \phi(x)]$, together with $\phi(B)$. Once we add (the analogically inferred) $\phi(A)$ to form Th' , the formula $\phi(x)$ becomes trivial; *i.e.*, $Th' \models \forall x \phi(x)$, meaning that $Trivial(\phi, Th')$. This is despite the fact that this formula is not trivial with respect to the original theory, $\neg Trivial(\phi, Th)$.

TR5. *Analogy's* Need for *Triviality*:

The point above indicated a situation where Equation 2.18 does not hold; where the result of an analogical inference does not qualify as an analogy. One way of avoiding the problem is to replace the $\neg Trivial$ condition with a slightly stronger condition in Figure 2-1 (and arguably in Definition 2 as well). We could insist that φ be strongly non-trivial with respect to A's position (the i^{th} argument) and the starting theory, i.e., $StronglyNonTrivial(\varphi|_i, Th)$, where

Definition 7 $StronglyNonTrivial(\phi, Th) \iff \exists c Th \models \neg\phi(c)$

By monotonicity of consistent additions, this guarantees that $StronglyNonTrivial(\varphi|_i, Th')$, as desired.

I choose, however, not to impose this criterion, even though this means that Equation 2.18 may fail. Why?

[1] In practice, the special instances of the learning by analogy task often impose other constraints, which in turn, render this triviality check unnecessary. For example, even the $StronglyNonTrivial$ constraint is subsumed by the **Abstraction** constraint presented in Section 4.2.

[2] At a theoretical level, I still feel justified in imposing a weaker criterion for establishing an analogy than for exhibiting an analogy. In the former case, we see the theory "in transition", and observe the "novelty" of this new $\varphi(A)$ fact. I feel that this additional information is enough even for these pathological conjectures to qualify as legitimate (if strained) analogical inferences.

2.5 Complexities of Analogical Inference

*"O Nature, and O soul of man! how far beyond all utterance
are your linked analogies! not the smallest atom stirs or lives
on matter, but has its cunning duplicate in mind."*

Moby Dick, Melville (1851)

This chapter has described what it means to learn by analogy. Based on Section 2.1's description, one might assume that analogical inference amounts to simple instantiation, that it is enough to merely instantiate a formula with new terms. Section 2.2, however, has already indicated that the world is not so simple.

There are two reasons why analogical inference is a difficult task. One complication arises if new terms are generated; that is, when the starting information is reformulated. (This dissertation only touches on this issue; *cf.* Section 9.3.) This section discusses only second issue: the tremendous size of the search space, even ignoring the issue of reformulation.

To repeat the situation: we are given a pair of analogues, A and B, and an initial theory, Th . Our goal is to determine the relevant source fact, $\varphi(b_1, \dots, B, \dots, b_n)$, and the new target instantiation, $[a_1, \dots, A, \dots, a_n]$. The key questions are:

- What source information is relevant?
i.e., what is $\varphi(b_1, \dots, B, \dots, b_n)$, the *source* of the analogy?
- How do we use this to derive the new target fact, $\varphi(a_1, \dots, A, \dots, a_n)$?
i.e., what is the target instantiation, $[a_1, \dots, A, \dots, a_n]$?

Figure 2-4: Analogy Questions

Section 2.2 indirectly suggested some complexities of analogical inference. The rest of this section states the issues more precisely, presenting an informal description of the vast size of this space followed by some quantitative measures.

Informal: Consider the first question posed in Figure 2-4: what is the source of the analogy? Imagine we are presented with a flat list containing all EC facts,¹² and asked to select the facts which are relevant for this particular task. How did we know that the analogy sentence should be the conjunction of the particular EC facts shown in Figure 2-3, which describe Current, VoltageDrop, Resistors, *etc.*? Why didn't we consider the cost of electricity, the history of electricity, the owner of the wires, or the timing characteristics of such circuits?

This unconstrained search through EC might be appropriate if all we know is that

Fluid Systems are like Electric Circuits.

¹²Recall EC is a collection of electric circuit facts. It is a subset of our initial theory, $EC Th_{CF}$. Similarly, FS is a collection of facts about fluid systems, $FS Th_{CF}$.

Here, however, we are told more. In particular, we are told that

FlowRate is like Current.

(We are also given a specific problem to solve. Chapter 3 shows how we can use this objective as a further constraint.)

This leads to an obvious — but faulty — proposal: Perhaps the source analogy fact is composed of precisely those EC facts which lexically mention Current? Unfortunately, this “syntactic inclusion” criterion is neither necessary nor sufficient:

Not Necessary: There may be EC sentences which mention Current but are irrelevant or, worse, lead to a contradictory theory. It is easy to find sentences which we would not even consider: like facts about devices to measure Current or equations which relate Current to the heat dissipated by associated devices. While one may find possible analogues for these statements in the FS World, they are not relevant to the “Find the flowrate” task.

Not Sufficient: Recall that we need the conservation law for PressureDrop to solve this FS problem, and realize that it could not have been derived without the fact that

The sum of all VoltageDrops around any closed loop is zero.

It seems clear that the above EC fact must be included, even though it did not lexically contain Current.

Further examination of the above “Sufficiency Argument” points to another complication: Just where does VoltageDrop (and its analogue PressureDrop) come from? Critical as it is to solving the problem, it is neither mentioned in the problem statement nor given in the hint. If we are allowed to add other EC terms, why not add capacitance, or timing delays, or resistor color codes? (The range of possibilities becomes unbounded if we are allowed to define new terms.)

Formal: The preceding informal arguments show that Definition 2’s criteria is incredibly general and woefully underspecified: it is trivial to generate an arbitrarily

large number of (uninteresting) analogies. Note:2-6 presents some quick, back-of-the-envelope calculations based on lexical constraints. The rest of this section, instead, uses some syntactic criteria for estimating the size of the space.

Begin by considering just the first of Figure 2-4's questions: how many source facts, $\varphi(b_1, \dots, B, \dots, b_n)$, are *a priori* possible. As this fact is derivable from Th , we can consider its finite support, $ST_{Th}(\varphi(b_1, \dots, B, \dots, b_n))$, where $ST_{Th}(\sigma)$ is the smallest subset of Th from which σ can be derived:

Definition 8 $ST_{Th}(\sigma) = \{\rho_i\} \iff$

$$\begin{aligned} & \{\rho_i\} \subseteq Th \quad \& \\ & \{\rho_i\} \models \sigma \quad \& \\ & \{\rho_i\}_{i \neq j} \not\models \sigma. \end{aligned}$$

(See [End72].) The $ST_{Th}(\varphi(b_1, \dots, B, \dots, b_n))$ set is precisely the subset of Th 's sentences which need to be considered — i.e., the analogy would hold even if we eliminated the other elements, $Th - ST_{Th}(\varphi(b_1, \dots, B, \dots, b_n))$. As different analogies may use arbitrary $\varphi(b_1, \dots, B, \dots, b_n)$ s, this $ST_{Th}(\varphi(b_1, \dots, B, \dots, b_n))$ could be any subset of Th . This means that the subtask of simply finding the relevant source facts is itself exponential, as it could involve searching through the entire 2^{Th} power set.

This support set addresses only the first of Figure 2-4's questions; determining the analogy involves two other considerations. (Hence, this exponential estimate is only a lower bound on the number of possible analogically inferred conjectures.) Figure 2-5 illustrates this, showing the various steps required to go from a given theory to a possible analogy conjecture. The above discussion suggests that each of the 2^{Th} different subsets of the theory Th may lead to a distinct analogy; i.e., the $\overset{1}{\rightsquigarrow}$ mapping, from theory to support set, branches exponentially.

The other two mappings are also non-trivial. Subsection 9.3.1 demonstrates that different analogies can arise from semantically identical source facts. This means that a given support set can lead to different analogies, i.e., the $\overset{2}{\rightsquigarrow}$ link is a one to many map. (As another way to look at this, the actual formulation of the $\varphi(b_1, \dots, B, \dots, b_n)$ fact needs to be considered as well.)

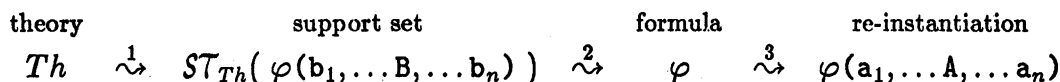


Figure 2-5: Number of Possible Analogies

A particular analogy formula can lead to many distinct analogies as well, each corresponding to a different target instantiation. That is, there may be many different sets $\{a^k_i\}_i$ s for which $\varphi(a^k_1, \dots, A, \dots, a^k_n)$ is a legal analogical inference. Note:2-6 addresses this consideration.

This matches our intuitions: the more Th includes about B , the more meaningful support sets can be considered, and this leads to more analogies. The number also depends on our knowledge of the target analogue, A . In particular, the more Th includes about A , the fewer analogical inferences can be drawn. This is because every fact about A serves as a constraint, reducing both the number of possible analogy formulae, φ^k , and then the number of $\{a^k_i\}_i$ sets which can be used to re-instantiate any such formula.

The rest of this section makes this argument more precise, showing the the exponential number of possible analogies. This follows from the observation that analogies can be combined. Given any two legal analogical inferences,

$$\begin{aligned} Th, A \sim B &\vdash \varphi_1(A) \\ Th, A \sim B &\vdash \varphi_2(A), \end{aligned}$$

we can define $\psi(x) \equiv \varphi_1(x) \& \varphi_2(x)$, and, barring pathological cases, find this $\psi(A)$ to be a legal analogical inference; $Th, A \sim B \vdash \psi(A)$.¹³

Actually only one of the formulae must be a legal analogy; it is sufficient that the other (without loss of generality φ_2) only satisfy $Th \models \varphi_2(B)$ and $Th \not\models \neg \varphi_2(A)$:

$$\begin{array}{r} Th, A \sim B \vdash \varphi_1(A) \\ Th \models \varphi_2(B) \\ Th \not\models \neg \varphi_2(A) \\ \hline Th, A \sim B \vdash \varphi_1(A) \& \varphi_2(A) \end{array} \quad (2.19)$$

¹³This "proof" would involve demonstrating that this ψ formula satisfies the four conditions shown in Figure 2-1. Unfortunately, only three of those requirements are met. Note:2-7 describes some pathological cases where the **Unknown** condition does not hold.

We can use Equation 2.19 to construct a subset of all legal conjunctive analogy formulae. First, we need to define the class of “base unary existential formulae”, *buefs*:

Definition 9 *A base unary existential formula is an atomic formula whose relation has all but one of its formal arguments existentially bound.*

Hence, a *buef* is a formula with one free variable, as the name implies. So, for example, $\phi_1(t) = \text{Kirchoff1}(t)$ and $\phi_2(x_2) = \exists ?_1 ?_3 \text{Group} (?_1, x_2, ?_3)$ would each qualify, but $\phi_3() = \exists ?_1 \text{Kirchoff2} (?_1)$ and $\phi_4(x_1, x_2) = \exists ?_3 \text{Monoid}(x_1, x_2, ?_3)$ would not. (We later complicate this situation by considering the actual values of these existentially bound variables.)

Let Ψ_1 refer to the set of all *buefs* which qualify as legal analogical inference formulae, and Ψ_2 , to the *buefs* which qualify for the φ_2 rôle shown in Equation 2.19 (but are not legal analogies). That is,

$$\begin{aligned} \Psi_1 &= \{ \varphi_1 \mid \text{buef}(\varphi_1) \ \& \ [Th \models \varphi_1(B)] \ \& \ [Th \not\models \neg \varphi_1(A)] \ \& \ [Th \not\models \varphi_1(A)] \} \\ \Psi_2 &= \{ \varphi_2 \mid \text{buef}(\varphi_2) \ \& \ [Th \models \varphi_2(B)] \ \& \ [Th \not\models \neg \varphi_2(A)] \ \& \ [Th \models \varphi_2(A)] \} \end{aligned}$$

Now consider the conjunction of any non-empty subset of Ψ_1 , *anded* with the conjunction of any subset of Ψ_2 :

$$\varphi_{\Pi}(x) \equiv \left[\bigwedge_{\psi_1 \in \Pi_1} \psi_1(x) \right] \ \& \ \left[\bigwedge_{\psi_2 \in \Pi_2} \psi_2(x) \right] \quad (2.20)$$

where $\{\} \subset \Pi_1 \subseteq \Psi_1$ and $\Pi_2 \subseteq \Psi_2$. By construction, the resulting formula φ_{Π} is a legal analogy.

This leads to an exponential number of lexically different analogies:

$$(2^{\|\Psi_1\|} - 1) * 2^{\|\Psi_2\|} \quad (2.21)$$

As expected, this increases exponentially with the number of facts known about B, especially if the corresponding A fact is not known. (This follows from the observation that the memberships of Ψ_1 and Ψ_2 each increase with additional knowledge about B, and decrease as more is learned about A.)

This estimate is conservative. A more realistic value needs to consider the actual values of the existential values included in each *buef*. This suggest we deal with *bufs*, where

Definition 10 *A base unary formula is an atomic formula whose relation has all but one of its formal arguments instantiated to a ground term.*

Each *buef* leads to at least one, and usually many, distinct *bufs*. Furthermore, a particular analogical inference may use a pair of related but different *bufs*, rather than a single *buef*.

To illustrate this point, consider the Ohms Law relation:

$$\text{Ohms}(t, c, r, l) \Leftrightarrow [\forall d \text{ l}(d) \Rightarrow [c(j_d^1, j_d^2, [d]) = t(j_d^1, d) * r(d)]]$$

We can use it to form the *buef* $\phi_O(x) = \exists c r l \text{ Ohms}(x, c, r, l)$. This single *buef* corresponds to several distinct *bufs*, including

$$\begin{aligned} \phi_1(x) &= \text{Ohms}(x, \text{VoltageDrop}, \text{Resistance}, \text{Resistors}) \\ \phi_2(x) &= \text{Ohms}(x, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \end{aligned}$$

among many others. In a language with $\|\mathcal{L}\|$ distinct symbols, there are $\|\mathcal{L}\|^3$ different Ohms-induced *bufs* (including the two ϕ_i s shown above) all derived from the single *buef* ϕ_O . There are many more if we allow arbitrary terms as well as pre-defined constants.

Recall again the analogy discussed in Section 1.3. This single analogical inference paired $\phi_1(\text{Current})$ with $\phi_2(\text{FlowRate})$: i.e., it used one ϕ_i for the source analogue and another ϕ_j for the target. Hence, $Th \models \phi_1(\text{Current})$, $Th \not\models \phi_2(\text{FlowRate})$ and $Th \not\models \neg\phi_2(\text{FlowRate})$ are the necessary pre-conditions for a legal analogy.

We can now consider how many more analogies can be formed using these *bufs* instead of the earlier *buefs*. First, let Ψ'_1 and Ψ'_2 resemble Ψ_1 and Ψ_2 , but contain *bufs* as well as *buefs*. Not only is each Ψ'_i much larger than its counterpart, but we can construct an analogical inference using any pair of single-relation spawned *bufs*. This means we must now consider the number of ways of selecting pairs of subsets, rather than a single subset. For example, the Ohms relation alone could contribute

up to $(\|\mathcal{L}\|^3)^2 = \|\mathcal{L}\|^6$ different *buf* pairs, rather than the single *buf* entry of the earlier situation.

This analysis only considers conjunctions of *bufs*. If we add in other connectives, like negation and disjunction, we get an even larger space. Note:2-7 further describes this space, showing it is almost the full power-set of all possible literals!

Chapter 3

Useful Analogical Inference

The previous chapter defined analogical inference in its full generality and discussed its use as a mechanism for proposing new information. Our goal is more specific. We want to consider only certain analogical inferences, namely, only those analogies which suggest *useful* conjectures. (We define *useful* in terms of expected utility in solving standard problems.)

This chapter describes one way of using a particular problem to focus the search for new facts. This leads to our notion of *useful analogical inference*, an important refinement of the general analogical inference process. Section 3.1 motivates this special case by describing other research with similar goals and demonstrating how this objective fits within the underlying goals of my research. Section 3.2 formally defines what we mean by a *useful analogical inference*. Section 3.3 extends the example began in Section 2.2 to illustrate how this additional requirement constrains the search. Here we see that there may still be more than a single possible useful analogy; Section 3.4 presents some intuitions which suggest which of these should be produced first — *i.e.*, it begins to characterize *good* useful analogies. (Chapters 4 and 5 use these insights to propose heuristics which order and prune this space of legal analogies, towards efficiently finding the useful analogies.)

3.1 Analogical Inference for a Purpose

This section discusses how the objective of solving a specific problem can facilitate a learning process. After describing the general approach, it focuses on how this idea applies to the current analogy situation.

Almost all learning — both human and machine — is done within a context. The canonical context is solving a particular problem. (Indeed, this problem-solving context seems sufficiently encompassing that it is hard to find any situation which cannot be worded in terms of solving some problem.) This is the basis of the case method approach, where a series of specific problems are presented, each as an impetus to learn certain new facts. (The [BvL81,vB79,vL85] research deals extensively with this paradigm, to cite just one example from cognitive science.)

Much of the current research in machine learning deals with *learning for a purpose*. Mitchell coined this phrase and demonstrated its effectiveness within the LEX project (see [Mit83,Utg84]). This theme is repeated in the more recent LEAP work [MMS85a], which is explicitly based on the claim that a typical learning episode is learning to solve some problem. This leads to explanation-based learning, a hot research topic in current machine learning research; see [Mit85], especially [DeJ85,Leb85,Min85,MMS85b,Sch85].

Historically, analogy was almost always used for a specific purpose, often to explain some unfamiliar phenomenon. This dates back (at least) to Plato, who, for example, explained the Idea of the Good by the similarity analogy: “the Idea of the Good makes knowledge possible in the intelligible world just as the sun makes vision possible in the perceptual world” [Emm85]. More recently, Nagel discussed the distinction between “useful analogies”, which help to solve a problem (where this “problem” may be to explain some phenomenon), and the others [Nag61].

Following this history, essentially all of the early analogy systems have been constructed to work on a specific task. Evan’s program tried to solve “geometric-analogy intelligence-test questions”, of the form $A:B :: C:D_i$, for one of the five presented D_i s [Eva68]. Kling’s Zorba program similarly had a specific goal: to effectively find a proof for a given theorem.

This idea is even more prominent in recent analogy research. Most have a specific problem to solve; *cf.*, [WBKL83], [AFS83,And83] and [EA81]. Kedar-Cabelli's current work, as one example, discusses how the particular problem can be used to determine which of the collection of available facts are relevant: *i.e.*, which should be considered part of the analogy [Ked84,Ked85]. Similarly, both Burstein [Bur84] and Carbonell [Car81a,Car83a] employ a particular problem to help determine which facts should be considered important.

In general, these analogies programs map the target problem into a corresponding source problem, and use this analogous problem (or, usually, a symbolic solution to this derived problem) to suggest how to solve the original target problem. For example, Zorba used a particular problem in the source domain to suggest which target domain facts are likely to be used in the proof. (Section 9.4 provides a more complete description of these systems; see especially Point Lit3.)

The underlying purpose of each of these analogy programs, like mine, is to solve some specific problem. The next section describes a model of analogical inference which incorporates this additional criterion. The next chapters show how we can move this constraint into the generator, by using abstractions.

3.2 Definition of Useful Analogical Inference

The previous chapter claimed that the result of a general analogical inference is a new (*i.e.*, independent) conjecture, which is then incorporated into the initial knowledge base. That definition permits arbitrary new facts, which leads to the tremendous number of legal analogical inferences which \sim allows.

The previous section discussed our more specific objective. Rather than consider arbitrary new conjectures, we want to consider only the ones which allow us to solve a specific problem. This leads to the notion of a *useful analogical inference*. Its input includes a particular problem to solve (*e.g.*, "Find the flowrate") in addition to the analogical hint (*e.g.*, "FlowRate is like Current") and the initial knowledge base (*e.g.*, the Th_{CF} shown in Figure 2-2). It assumes, furthermore, that finding the appropriate analogy leads to a solution to this target problem. Stated

| | |
|---|---|
| $Th, A \sim B \vdash_{PT} \varphi(a_1, \dots, A, \dots, a_n)$ | |
| where Common: | $Th \models \varphi(?b_1, \dots, B, \dots, ?b_n)$ |
| Unknown: | $Th \not\models \varphi(a_1, \dots, A, \dots, a_n)$ |
| Consistent: | $Th \not\models \neg\varphi(a_1, \dots, A, \dots, a_n)$ |
| NonTrivial: | $\neg Trivial(\varphi _i, Th)$ |
| Useful: | $Th + \varphi(a_1, \dots, A, \dots, a_n) \models PT$ |

Figure 3-1: Definition of Useful Analogical Inference

formally, the objective of a useful analogical inference is to find a new conjecture, $\varphi(a_1, \dots, A, \dots, a_n)$, such that $Th' = Th + \varphi(a_1, \dots, A, \dots, a_n)$ produces an answer to the target domain problem. For notation, we use PT to refer to this target problem; our canonical PT is “Find the flowrate”.

This is the Usefulness condition:

$$\text{Useful: } Th + \varphi(a_1, \dots, A, \dots, a_n) \models PT. \quad (3.1)$$

When added to the definition of general analogical inference (shown in Figure 2-1), we obtain the definition of *useful analogical inference* shown in Figure 3-1. This is indicated using the \vdash_{PT} operator, which is the general analogical inference operator, \vdash , adorned with a PT subscript to indicate this additional Usefulness constraint. (To repeat Point Defn3 on page 17, if we omit the **Common** criterion, we are again left with a good general definition of learning for a purpose — *i.e.*, the remaining conditions describe a way of acquiring an unknown fact which can be used to solve a particular problem.)

This model of learning by analogy is honed to consider the task of learning the new facts necessary to solve a given problem. While the more general \vdash would consider any way that FlowRate is like Current, \vdash_{PT} is more specific. It considers only certain ways in which they are similar: namely, only the ways which suggest some solution to the given problem. (Here, \vdash_{PT} seeks some FlowRate to Current similarity which suggests a solution to this particular “Find the flowrate” problem.)

To summarize \sim_{PT} 's desired behavior: a useful analogical inference process is given a pair of analogues A and B, a problem PT and an initial theory Th . Its goal is to expand this Th as necessary to form a new theory which exhibits an analogy between A and B, and in which PT is solvable. This means it seeks a formula φ and target instantiation $[a_1, \dots, A, \dots, a_n]$ such that

$$Analogy_F(A, B, Th + \varphi(a_1, \dots, A, \dots, a_n), \langle \varphi [a_1, \dots, A, \dots, a_n] [b_1, \dots, B, \dots, b_n] \rangle)$$

and

$$Th + \varphi(a_1, \dots, A, \dots, a_n) \models PT.$$

3.3 Example of Useful Analogical Inference

The previous section defined the Useful clause; this section describes how it is used. The two main points are: [1] it does prune the space of \sim analogies, and [2] it still leaves many remaining \sim_{PT} analogies.

Recall again the φ_{RKK} formula of Section 2.2. Equation 2.8 has already demonstrated that it qualifies as a legal \sim general analogical inference. To show that it qualifies as a \sim_{PT} useful analogical inference, we must show that it also satisfies the Useful constraint. That is,

$$\begin{aligned} \text{Useful: } Th_{CF} + \varphi_{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \\ \models \text{"Find the flowrate"} \end{aligned} \tag{3.2}$$

must hold. This proof uses the definitions of φ_{RKK} 's four component clauses, all shown in Figure 2-2, and requires the claim that they are sufficient to solve the given hydraulics problem. (The details are cumbersome to show, but can be inferred from Figure 1-2. In particular, observe that the hydraulics versions of both Kirchoff's and Ohms laws are required.)

We now see that any formula more specific than φ_{RKK} would also qualify as a legal \sim_{PT} analogy formula, provided it satisfies \sim 's requirements. In fact, any analogy formula more specific than φ_{Min} (shown in Equation 2.10) can lead to a

useful analogy,¹ even if it is more general than φ_{RKK} . Consider, however, the yet more general formula, φ_{Ohm} shown in Equation 2.11. While it does qualify as a \sim general analogical inference, it is not a legal *useful* analogical inference for this *PT* problem. This is because the hydraulic's Ohm's Law, by itself, is not sufficient to solve this "Find the flowrate" problem; we also need to know that $\text{Kirchoff2}(\text{PressureDrop})$ holds.

What about less related formulae? Some of these might still qualify. For example, assume the knowledge base included Faraday's Law, and had general rules which use it to describe the behavior of alternating current through a circuit. This might suggest various conjectures which describe "alternating flowrate", and these might lead to an answer to the initial "Find the flowrate" problem. (While this answer is, no doubt, very different from the earlier "correct" answer, \sim_{PT} has no means to ascertain it to be wrong: in particular, this conclusion is consistent with all of its known facts.)

All of these examples involve facts which a knowledgeable linear system engineer would consider reasonable. Of course, the initial Th_{CF} knowledge base may not be that smart. It might, for example, consider the Cost of each pipe to be the Resistance-like constant of proportionality (which relates the PressureDrop through a pipe to its FlowRate) and use this to determine an answer to the target problem (albeit a strange one).

What about yet other possible general analogy formulae, which deal with, for example, history of cost? It is unlikely that any of them would lead to useful analogies. This is because there is no connection between the target "Find the flowrate" problem and the conjectures those analogies would suggest.

To help understand this, compare the earlier φ_{RKK} -based analogy with an analogy which suggests some fact about the history of FlowRate. The φ_{RKK} analogy suggests new conjectures like $\text{Kirchoff1}(\text{FlowRate})$. Such assertions provide ways of computing the FlowRate through a pipe, and hence suggest some solution to the target problem. By contrast, knowing more about the history of FlowRate does not

¹The phrase *useful analogy* abbreviates "target instantiation of analogy formula of a useful analogical inference". It refers to just the analogies which are useful to the particular problem. Of course, almost any possible analogy is useful to some problem or other.

lead to any rule which tells how to compute the `FlowRate` through a pipe. This independence is also true in the source domain: there is no connection between the history of current and the current through a wire either.

We likewise have no general rule which tells us how to use x 's `CostPerHour` to compute the value of x through a device. However, we do have a rule which uses the assertion that x satisfies `Kirchoff1` to compute the x through a device.

These examples show that relatively few of the general analogies would qualify as useful analogies. We also see that there are still a sizable number of useful analogies. This exposes two other problems with Section 2.2's strawman proposal that we use a blind generate-and-test process to find useful analogies. That section discussed the problems of simply finding a qualifying (general) analogy formula and its target instantiation. One problem is: As relatively few of the legal \vdash analogies qualify as \vdash_{PT} analogies, a yet smaller percentage of the generated assertions would survive both pruning steps.

The second objection stems from the observation that there are still quite a few remaining useful analogies. These analogies, however, are not all alike. Rather than return these legal useful analogies in random order, they should be returned in a meaningful order, with the more likely ones returned first.

This reflects one important objective of this research: to find (and return) the better analogies first. Given the sparseness of useful analogies within space of legal sentences, and the number of possible useful analogies, this is a difficult task for a blind generate-and-test process.

All is not lost, though. Section 3.4 presents some intuitions which address these problems; Chapters 4 and 5 expand these into operational heuristics. Chapter 7 demonstrates that these rules are effective, by enumerating the relatively few answers the *NLAG* system actually produced in this specific situation.

3.4 Intuitions about Useful Analogies

The previous chapter defined analogical inference as a process for adding new conjectures which pertain to the target analogue. This chapter proposed the focusing constraint that we consider only conjectures which are useful to a given problem. The next goal of this research is to characterize the space of these desirable analogically-inferred useful conjectures, both by restricting which conjectures should be postulated and by determining an effective ordering.

As one proposal, perhaps *NLAG* should produce only *semantically correct facts*. Note:1-2 discusses why this is both impossible and undesirable. Another possible objective is that the analogy be easy to find: Note:3-1 analyzes this goal.

This section focuses on one aspect of this problem, addressing the issue of how much (i.e., how many conjectures) should be added. It presents three “obvious” intuitions, labeled I_{Most} , I_{Least} and I_{Close} , and demonstrates the tension among them. Much of this dissertation can be viewed as an attempt to resolve this conflict. In particular, this issue motivates both the use of abstractions (Chapter 4) and the least constraining heuristics (Section 5.3); it also surfaces in Subsection 9.2.2.

The I_{Most} intuition asserts that we should add as much as we can, insisting only that these additions be “reasonable”. (Its strongest argument is in terms of efficiency: Given how simple it is to merely state one fact, it is inefficient to use this elaborate analogy mechanism unless you are communicating a great deal of information.) The opposing view, I_{Least} , asserts that we should not expect too much of the analogy. It is easy to find anecdotes where the hearer has *overextended* the analogy with disastrous effects. (See Figure 3-2, as well as [HM82] and [Bur83b].)

These two desires pull in opposite directions: I_{Most} wants to tease as much information from the analogical hint as possible: it advocates conjecturing *every possible connection* between *FlowRate* and *Current*. I_{Least} , on the other hand, favors a more conservative approach, of making as few additional assumptions as possible. Taken to the extreme, it would prefer adding in *nothing*, using only the facts initially known about the analogues.. Stated another way, I_{Most} seeks the *most* constraining legal extension of the known facts, while I_{Least} seeks the *least*



Figure 3-2: Over-Extended Analogy
(from Ros Chast, [Cha84])

constraining extension.

One way of resolving this tension involves a third intuition, I_{Close} : we often think of an analogical inference as "completing a whole", as adding in the other facts which are "causally connected" or interrelated (see [WBKL83], [Ked85] and [Dav85]), which follow from some selective inference (as in [Hob83a]), or which fulfill the n -ary relation (as in [Gen80b, Car81b]), etc. This reflects a compromise between I_{Most} and I_{Least} : Rather than require the learner to add as many facts as possible, I_{Close} claims it is sufficient to conjecture only those facts which are required to complete some meaningful, complete chunk. ²

In our current problem-solving context, it is easy to define these "chunks" *a posteriori*. Each meaningful chunk corresponds to the information required to solve some problem. In particular, we are happy to acquire just the facts needed to solve the particular problem posed. Our challenge is to determine these chunks *a priori*.

Chapter 4 ties this notion of chunks to abstractions, arguing that an abstraction

²This is related to the notion of natural kinds: each chunk reflects a good (read "useful") way of organizing the world. See [Ros73] and [Boy79].

instance qualifies as a complete chunk, at least in this problem solving context. This suggests that a good analogy is one which suggests just the facts required to complete an abstraction instance, and no more. Section 5.3 then elaborates this “and no more” point, arguing that the I_{Least} position (favoring the minimal extension) is appropriate within this *abstraction-based view of analogy*.

Three final notes:

- This discussion here has been quite informal. Subsequent sections (in particular, Subsections 5.3.3 and 9.2.2) provide semantic formalizations of these intuitions.
- The standard lore still agrees with I_{Most} , claiming that the maximally specific analogy (the one which expresses the greatest similarity connecting the analogues) should always be used. Section 9.2 argues against this position, demonstrating that such maximal analogies are not necessarily the best by showing that they do not efficiently lead to the conjectures which are likely to help solve problems.
- These three intuitions (I_{Most} , I_{Least} and I_{Close}) pertain to any learning process, not just learning by analogy. Chapter 4 argues that their resolution, in terms of abstractions, is as general.

Chapter 4

Use of Abstractions

The previous chapters defined the useful analogical inference process and discussed its use in conjecturing useful new information. This chapter proposes a particular method for finding these useful conjectures efficiently, based on *abstractions*.

Section 4.1 presents the intuitions behind our sense of abstractions, expressing the view that each abstraction encapsulates some previous problem solving experience and arguing that this abstracted information can be used again to solve new problems in new domains.

Section 4.2 formalizes this claim, stating my position that these *abstraction-based analogical inferences* are sufficient to cover a large class of situations. This section includes formal definitions of both the abstraction-based analogy relation, $Analogy_{CA}$, and the related abstraction-based analogical inference process, \Vdash_{PT} .

Section 4.3 explains why it is desirable to use abstractions when learning by analogy. It demonstrates how this approach constrains the search to useful analogies and thereby side-steps many of the complexities associated with the general analogical inference process. This section includes a behavioral description of the abstraction-based analogical-inference engine, *NLAG*.

Having established that abstraction-based analogical inferences are desirable, Section 4.4 demonstrates that they are possible. This requires the claim that these abstractions are both ubiquitous and re-usable.

Section 4.5 concludes this chapter by comparing this notion of abstractions with

related ideas from both artificial intelligence and other fields.

The rest of this dissertation is oriented around this model of analogy. Chapter 5 discusses various heuristics which make this search for useful abstraction-based analogies more efficient; Chapter 6 describes how I operationalized these rules in my *NLAG* implementation; and Chapter 7 demonstrates the effectiveness and utility of this implementation.

4.1 Informal Discussion of Abstractions¹

This section motivates a particular form of analogical inference, one based on abstractions. It discusses the intuitions which underlie our sense of abstractions, by describing what an abstraction is and illustrating how an abstraction can be formed in one domain and re-used in another. This requires addressing both of the questions posed in Figure 2-4 on page 29: first characterizing which sentences should be in the source of a useful analogy and then describing how to use this source information to produce the plausible propositions which should be added to the target theory.

Relevant Source Facts: Section 2.5 mentioned that we can consider the source of the analogy, $\varphi(b_1, \dots, B, \dots, b_n)$, to be a subset of the starting *Th* theory. The important observation here is that *only a small number of the $2^{\|Th\|}$ subsets of the $\|Th\|$ element theory make sense*. To understand this idea, forget about analogies for a moment and just consider the general task of solving a “Find the current” problem in the EC domain. The important realization is that only certain facts are relevant, namely, only those which deal with

Current, VoltageDrop, Resistors, Topological connections, *etc.*

Stating this in the more important direction, *all other facts are irrelevant*. That is, we do not have to even consider facts which pertain to the color of the wires, the owner of this circuit, or the history of electricity.

¹The reader should not be concerned by the hand-waviness of this section. Its objective is only to motivate the rest of this chapter; the actual validation of this “abstraction” concept is independent of these arguments. (It appears in Section 7.6.) See also the disclaimer in Section 4.4.

This simple observation — that sets of facts tend naturally to clump together — leads to our notion of a *perspective*:

Intuition 1 *A Perspective is a single cohesive cluster of facts.*

For example, the collection of facts we just described, which deals with **Current**, **VoltageDrop**, **Resistance** and **Resistors**, is one particular perspective. We refer to this set as the “resistance analogue” perspective of Electric Circuits, or **RKK-EC**. This collection of facts appears as the tagged subset of Th_{CF} in Figure 4-1; Figure 4-2 shows these facts in greater detail. (Notice that we derived this perspective *independently* of any analogy or any analogical processing. Note:4-1 elaborates this description, demonstrating how one can derive an abstraction *in toto* without appealing to any analogical process.)

There are other perspectives associated with Electric Circuits. In addition to this **RKK-EC** one, other subsets of EC facts² might deal with

- the history of electrical circuits — describing the design and construction of the first circuit ...
(What other events and people could have influenced this achievement?)
- bioelectricity — connection between electricity and biological effects ...
(What caused Galvani’s frog’s muscles to contract?)
- timing information — describing in race conditions, stability of flip-flops, ...
(Could this particular bug be caused by a timing situation?)
- physical devices — describing space requirements, physical shape, ...
(How much space does this circuit require?)
- commodity — describing who owns a particular circuit, the costs of its components, ...
(What type of equipment do rich people buy?)

²[Rob85] supplies a great deal of information about all of these perspectives.

Electric Facts ($EC \subset Th_{CF}$)

...

ModernTreatise(Current, W.Gilbert, 1600)

QuantifiedBy(Current, G.Ohms, 1827)

Kirchoff1(Current)

CostPerHour(Current, \$2.43)

Kirchoff2(VoltageDrop)

UnitsOf(Current, Amperes)

ConservedThru(Current, Resistors)

UnitsOf(Capacitance, Farads)

UnitsOf(Inductance, Henrys)

Ohms(Current, VoltageDrop,
Resistance, Resistors)

InDiscipline(Current, Electric)

...

$\varphi(\text{Current}, \dots)$

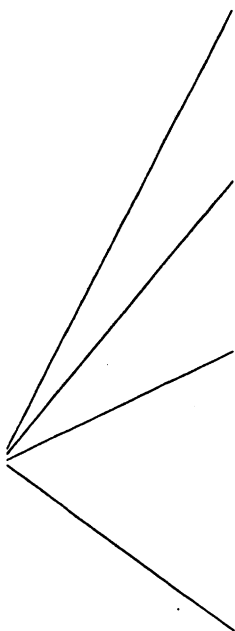


Figure 4-1: Example of a Perspective: Electric Domain

RKK-EC \Leftrightarrow

| |
|--|
| <p>Ohms(Current, VoltageDrop, Resistance, Resistors)</p> <p>$\forall d \text{ Resistors}(d) \Rightarrow$ $[\text{VoltageDrop}(j_d^1, j_d^2, [d]) = \text{Current}(j_d^1, d) * \text{Resistance}(d)]$ "VoltageDrop = Current * Resistance"</p> <p>ConservedThru(Current, Resistors)</p> <p>$\forall d \text{ Resistors}(d) \Rightarrow [\text{Current}(j_d^1, d) + \text{Current}(j_d^2, d) = 0]$ "Current is conserved through Resistors."</p> <p>Kirchoff1(Current)</p> <p>$\forall j \sum_{p:Conn(p,j)} \text{Current}(j, p) = 0$ "The sum of all Currents flowing into a junction is 0."</p> <p>Kirchoff2(VoltageDrop)</p> <p>$\forall loop \sum_{\langle i,j \rangle \in loop} \text{VoltageDrop}(i, j, [x]) = 0$ "The sum of all VoltageDrops around any closed loop is 0."</p> |
|--|

Figure 4-2: Definition of the **RKK-EC** Perspective

- the structure of the circuit — used for the processes of design, debugging, redesign, ...
 (Find the bug in a specific circuit.)
- other Lumped Element Linear System facts, describing inductance, transient behavior, *etc.* (Some of these are supersets of the **RKK-EC** collection.)
 (What is the impedance through a component of a specific circuit?)

By design, each perspective includes the facts needed to solve a certain class of problems. The parenthetical note following each above description provides a representative example of such a problem.

These perspectives play an important rôle in finding analogies: We claim that *the source of any useful analogy is such a perspective*. This means we need to search only a relatively small space to find these source sentences: rather than consider *all* $2^{|Th|}$ subsets of the collection of *Th* facts, we can just consider some of these subsets

— each corresponding to some perspective thought relevant to these analogues and to the problem at hand.

Corresponding Target Facts: Even given this source information, we still have to find the corresponding target sentences, *viz.*, the set of known facts to be selected from the target theory, appended to the new propositions which should now be conjectured. This task becomes relatively easy once the perspective is expressed in the appropriate *parameterized* form. That is, rather than think of **RKK-EC** as a subset of a theory, we can think of these resistance analogue facts as a *relation*, instantiated with various concepts — here **Current**, **VoltageDrop**, **Resistance** and **Resistors**. That is,

$$\mathbf{RKK}(\mathbf{Current}, \mathbf{VoltageDrop}, \mathbf{Resistance}, \mathbf{Resistors}) \equiv \text{facts in } \mathbf{RKK-EC} \text{ perspective.} \quad (4.1)$$

We call each such coherent relation an *abstraction*. Each abstraction specifies certain conditions which its arguments must fulfill — both individual predicates and various interrelations among the terms.³ Figure 4-3 provides **RKK**'s formal definition.

We call any ground clause which instantiates an abstraction (e.g., **RKK**(**Current**, **VoltageDrop**, **Resistance**, **Resistors**)) an *abstraction instance*. Each such clause is equivalent to a perspective; e.g., repeating Equation 4.1, **RKK**(**Current**, **VoltageDrop**, **Resistance**, **Resistors**) \equiv **RKK-EC**.

We see that the *source* of a useful analogy can be considered an instantiation of some abstraction in the source domain. (Here, the source of the analogy is an instantiation of the **RKK** abstraction in the Electric Circuit domain.) Similarly, the *target* of a useful analogy is just an *instantiation of the same abstraction* in the target domain. Hence, the target in this situation is an instantiation of the **RKK** abstraction in the Fluid System domain.

This suggests that the analogical inference process should look for target terms which satisfy the conditions of the **RKK** relation, conjecturing plausible new facts

³As many of these parameters are themselves functions and relations, these conditions are often second-order relations.

$$\begin{array}{c}
 \mathbf{RKK}(t, c, r, l) \iff \\
 \left. \begin{array}{l}
 \text{Ohms}(t, c, r, l) \\
 \quad \forall d \, l(d) \Rightarrow [c(j_d^1, j_d^2, [d]) = t(j_d^1, d) * r(d)] \\
 \\
 \text{ConservedThru}(t, l) \\
 \quad \forall d \, l(d) \Rightarrow [t(j_d^1, d) + t(j_d^2, d) = 0] \\
 \\
 \text{Kirchoff1}(t) \\
 \quad \forall j \quad \sum_{p: \text{Conn}(p, j)} t(j, p) = 0 \\
 \\
 \text{Kirchoff2}(c) \\
 \quad \forall \text{loop} \quad \sum_{\langle i, j \rangle \in \text{loop}} c(i, j, [x]) = 0
 \end{array} \right\}
 \end{array}$$

Figure 4-3: Definition of the **RKK** Abstraction

and proposing new terms, as necessary. As we saw above, this leads to an abstraction instance in the target domain, **RKK**(FlowRate, PressureDrop, PipeCharacter, Pipes).

We refer to this process as *abstraction-based (useful) analogical inference*, and represent it with the \Vdash_{PT} operator symbol. Once again, the $_{PT}$ subscript means that \Vdash_{PT} is used to suggest conjectures which help solve some problem. (The related *non-useful abstraction-based analogical inference* process is encoded by the unadorned \Vdash .)

The rest of this section provides a high-level description of how one could implement such a \Vdash_{PT} system, and closes with several preliminary comments on this process. The next sections flesh out this description.

\Vdash_{PT} 's Implied Process: The *NLAG* system implements this abstraction-based analogical inference process by finding an abstraction which has a *deducible* instantiation in the source domain and a *consistent* instantiation in the target domain. Any standard deduction process can handle the first part. Finding the target abstraction instance is more problematic. As the initial hint told us to regard FlowRate as Current, *NLAG* instantiates the **RKK** abstraction using FlowRate in Current's

position, that is, as **RKK**'s first argument. This means *NLAG* searches for an instantiation of the $?_i$ s which satisfies the proposition $\mathbf{RKK}(\text{FlowRate}, ?_2, ?_3, ?_4)$.

The satisfaction process used by the *NLAG* system postulates new conjectures, as necessary. For example, it would add $\text{Kirchoff1}(\text{FlowRate})$ to the theory if it is not already known. (I.e., *NLAG* would first check whether $Th \models \text{Kirchoff1}(\text{FlowRate})$; if not, it would propose adding $\text{Kirchoff1}(\text{FlowRate})$.)

NLAG also has to fill in the other arguments, the $?_i$'s of $\mathbf{RKK}(\text{FlowRate}, ?_2, ?_3, ?_4)$. In seeking an instantiation for $?_2$, for example, it would seek the FS function of a pair of junctions which is conserved around any closed loop. This means *NLAG* searches for an FS function which resembles the EC function VoltageDrop in this particular way. Here, this prompts *NLAG* to propose that PressureDrop has this property. (If this PressureDrop symbol was not in the language, an \Vdash_{PT} process might create a new symbol, $G0017$, to fill this position in the **RKK** relation — i.e., which would (by definition) satisfy both this and all other **RKK**-induced VoltageDrop -like properties. My particular *NLAG* system does not actually create new symbols. See the H_{FT} rule in Subsection 5.3.6.)

In summary, *NLAG* uses an instantiation process to find the source of the analogy followed by a satisfaction process to find the target. The latter process finds a set of target propositions, including both previously known facts and proposed conjectures. These two sets of target facts, together, correspond to the relevant EC facts, i.e., to the source of the analogy.

Comments: This section suggested that we seek only common abstractions when searching for analogies, rather than arbitrary common formulae. This is clearly more efficient, as there are an arbitrarily large number of possible formulae but only a relatively small number of possible abstractions. (I.e., relatively few of the possible \sim analogies correspond to common abstractions.) Fortunately, this smaller class of common-abstraction analogies still covers a large number of important cases. In particular, this research claims that they account for a large percentage of the useful analogical inferences, i.e., of the possible \Vdash_{PT} inferences. This is based on the tacit assumption that

| | |
|--|-------|
| <p><i>Most useful formulae have already been noted, and already exist as abstractions.</i></p> | (4.2) |
|--|-------|

This follows from the view that abstractions are compilations of solutions to past problems: *i.e.*, they are the “wisdom of the ages”, gleaned from solutions to earlier problems. In terms of our application, this means that each abstraction is a heuristic: each claims that, given certain facts, it is reasonable to expect that certain other facts — *viz.*, those associated with the same abstraction — will hold as well.

This notion of abstractions ties in with the intuition labeled I_{Close} in Section 3.4. There we claimed that a useful analogy should cover a “closed” collection of facts. This section has argued that *abstractions* capture this sense of closure; as each abstraction corresponds to the facts needed to solve a class of problems.

The rest of this chapter elaborates this insight. The next sections formally define this important subclass of analogies — those based on common abstractions — and elaborate many of the points suggested above, addressing why this model of analogy is both efficient and possible. Section 4.3 also shows *NLAG*’s behavior in more detail.

4.2 Use of Abstractions in Analogies

The previous section suggests that we consider only certain possible analogies, *viz.*, only those which correspond to common abstractions. This section describes how we realize this suggestion, in terms of the *Common Abstraction Assumption for Useful Analogical Inference*, hereafter called H_{Abst} .

The general definition of useful analogical inference (Figure 3-1’s \vdash_{PT}) imposed no restriction on the type of formula which could be used to link a pair of analogues. This allows arbitrary combinations of clauses, leading to a huge number of possible formulae. (See Section 2.5.)

This H_{Abst} rule prunes that space of possible analogies by legislating that only “abstraction formulae” be considered. Stated more precisely, H_{Abst} restricts which formulae can qualify as analogical formulae: It permits only atomic formulae, and

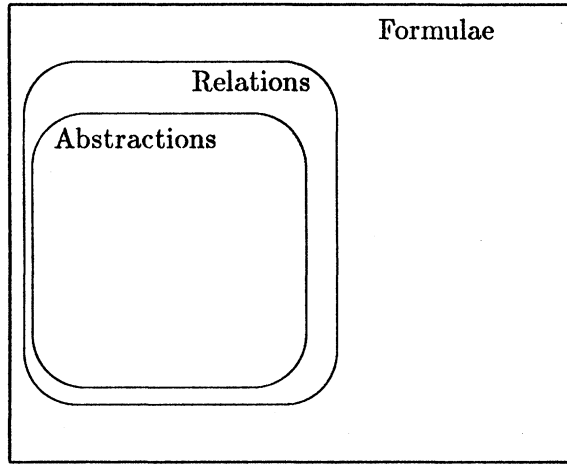


Figure 4-4: Formulae, Relations and Abstractions

of those, only the ones whose relation symbol has been “tagged” as an *abstraction*. Figure 4-4 shows the relevant containments, among formulae, relations and abstractions. The fact that the space of abstractions almost fills the space of relations is intentional. (See the discussions in both Section 7.6 and Note:10-1.)

This section illustrates this “abstraction formulae” idea with some examples, states the resulting definition formally and concludes with some relevant observations.

Examples: The formula

$$\varphi_1(d, op, id) = \mathbf{Group}(d, op, id)^4$$

qualifies as an abstraction-based analogy formula, since the second-order assertion $\mathbf{Abstraction}(\mathbf{Group})$ is included in the initial theory (i.e., as $Th \models \mathbf{Abstraction}(\mathbf{Group})$). On the other hand, the atomic formula

$$\varphi_2(x, y, z) = +(x, y, z)$$

does not qualify since $\mathbf{Abstraction}(+)$ is *not* in our knowledge base, nor could the

⁴We will continue to write the names of abstractions in this bold-face fixed-width font; e.g., **Group**.

compound formula

$$\varphi_3(d, op, id) = \text{Group}(d, op, id) \ \& \ \|d\| > 10.$$

An *abstraction formula* is formally defined as an atomic formula whose relation is an abstraction:

Definition 11 $AbstForm(\varphi) \iff$
 $AtomicFormula(\varphi) \ \& \ Abstraction(Relation(\varphi))$

Re-using the examples above, $AbstForm(\varphi_3)$ is false because φ_3 is not an atomic formula (i.e., as $\neg AtomicFormula(\varphi_3)$) and $AbstForm(\varphi_2)$ is false because φ_2 's relation, $Relation(\varphi_2) = +$, is not an abstraction. Notice that $AtomicFormula$ is a meta-level predicate, as it selects a subset of syntactically legal formulae; the $Relation$ function is both meta-level and second-order, as it maps formulae into relations; and $Abstraction$ is a second-order predicate, which selects a subset of the relations.

Typical examples of abstractions include **Group**, **Ring** and **Field** from algebra, and **RKK** (the “resistance analogue”) and **RCKK** (the “resistance and capacitance analogue”) from the lumped linear system domain. Section 4.4 lists other abstractions from diverse domains.

Definitions: We use this notion of abstractions to define the $Analogy_{CA}$ relation, which only accepts those analogies which are based on *common abstractions* (see [Pol54, Gen80a]):

Definition 12 $Analogy_{CA}(A, B, Th, \langle S [a_1, \dots, A, \dots, a_n] \cdot [b_1, \dots, B, \dots, b_n] \rangle)$
 $\iff Analogy_F(A, B, Th, \langle S [a_1, \dots, A, \dots, a_n] [b_1, \dots, B, \dots, b_n] \rangle)$
 $\ \& \ Th \models AbstForm(S)$

where the analogues $A = a_i$ and $B = b_i$ for the same i . We refer to the triple $\langle S [a_1, \dots, A, \dots, a_n] [b_1, \dots, B, \dots, b_n] \rangle$ as an *abstraction-based analogy*; or, in particular, as “a *common abstraction* (which links A and B)”.⁵

⁵As a simplifying shorthand, we will continue to equate the abstraction formula, S , with its

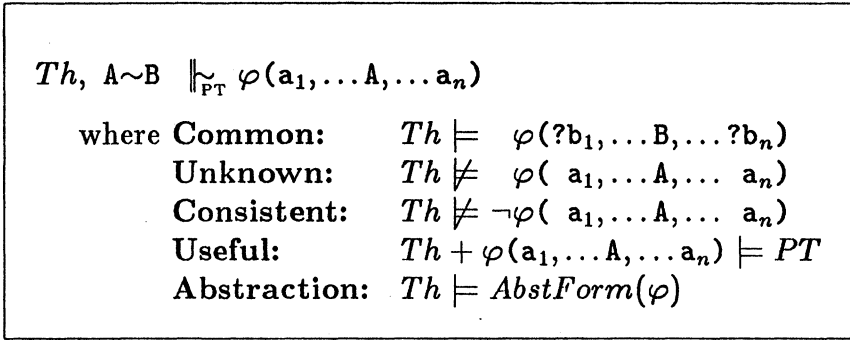


Figure 4-5: Abstraction-Based (Useful) Analogical Inference

Figure 4-5 presents the related inference process, \Vdash_{PT} , which incorporates H_{Abst} :

Heuristic 1 H_{Abst} : Use formula φ only if $AbstForm(\varphi)$

We conclude this section with various observations about this model of analogy.

CA1. Criterion for Abstraction Predicate:

The previous section suggested the essential *a posteriori* characteristic for an abstraction, in terms of its applicability to standard problems. We also impose an *a priori* requirement: an abstraction must be strongly non-trivial in all of its arguments — that is, $StronglyNonTrivial(S|_i, Th)$ must hold for all i . (See Definition 7 on page 28). This is why Figure 4-5's \Vdash_{PT} does not need an explicit **NonTrivial** condition. (The following comment, Point CA2, motivates another reason for this requirement.)

This is not a major limitation. In general, each argument has (at least) some type information: e.g., $RKK(?_1, c, ?_3, ?_4) \Rightarrow c \in Functions$. (See Figure 5-5 in Subsection 5.3.6.) As this implication eliminates many possible values, it is sufficient to guarantee that $StronglyNonTrivial(RKK|_1, Th)$ holds.

CA2. Analogy_{CA} \Rightarrow Analogy_F:

It is easy to see that this model of analogy is a refinement of the more general included abstraction, **S**. Hence, the abstraction **Group** can be used to refer to the formula **Group**(d, op, id), for variables d, op and id .

case given in Definition 2, i.e.,

$$\text{Analogous}_{CA}(A, B, Th) \Rightarrow \text{Analogous}_F(A, B, Th) \quad (4.3)$$

(where, by convention, Analogous_χ is the “exists-form” of the corresponding Analogy_χ — i.e., $\text{Analogous}_\chi(A, B, Th) \Leftrightarrow \exists \Sigma \text{Analogy}_\chi(A, B, Th, \Sigma)$).

Proof: Simply let $\varphi(x_1 \dots x_n)$ be $\mathbf{S}(x_1, \dots, x_n)$, and keep the same instantiations, $[a_1, \dots, A, \dots, a_n]$ and $[b_1, \dots, B, \dots, b_n]$. As $\mathbf{S}|_i$ is non-trivial, the \mathbf{S} abstraction formula is a legal analogy formula.

CA3. $\text{Analogy}_F \not\Rightarrow \text{Analogy}_{CA}$:

Equation 4.3, above, is intentionally a one-way implication. This is consistent with the comments first presented on page 53. Figure 4-6 succinctly restates that claim. There are two main salient features: For any given problem, PT , [1] few \vdash analogies are \vdash_{PT} analogies, and [2] most of *Useful Analogical Inference* (\vdash_{PT}) is covered by *Abstraction-Based Analogical Inferences* (\parallel). That is, most of the analogically derived new target facts which satisfy \vdash_{PT} correspond to abstractions — i.e., satisfy \parallel_{PT} .

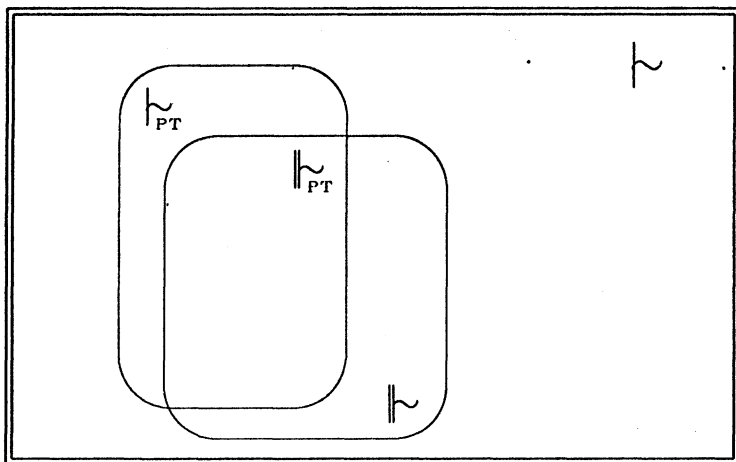


Figure 4-6: Different Types of Analogical Inference

CA4. Ties between \parallel_{PT} and Analogy_{CA} :

An abstraction-based analogical inference establishes a legal abstraction-based

4.3. WHY ABSTRACTIONS SHOULD BE USED FOR ANALOGICAL INFERENCE

analogy. (This mirrors the claim that a general analogical inference establishes a general *Analogy_F*; see page 22 in Section 2.3.) Furthermore, as an abstraction is *StronglyNonTrivial* in all of its arguments, there can be no counter-examples to this claim. (Thus, this claim is slightly stronger than the one for general analogical inference. See Point TR4 in Section 2.4.)

CA5. Ties between \Vdash_{PT} and \Vdash_{PT} :

This definition of an abstraction-based analogical inference differs from the general definition of useful analogical inference (in Figure 3-1) in only one respect: it imposes the single additional constraint that only abstraction formulae are allowed. Abstraction-based analogies differ from general analogies (in Definition 2) in exactly this same manner.

This ends the basic definitions, describing what an abstraction is and how they can be used. The following sections argue why this approach is both advantageous (in Section 4.3) and possible (in Section 4.4).

4.3 Why Abstractions *Should* be Used for Analogical Inference

This section discusses why abstractions are useful for this analogical inference task. Understanding this point requires a more detailed description of how abstractions are used. Below is a behavioral description of my abstraction-based analogical inference process, *NLAG*. It is followed by a discussion of the benefits and limitations of using this model of analogy.

Behavior: In a nutshell, the abstraction-based analogical inference process *NLAG* seeks a *common abstraction* which links the two analogues, where the source instantiation is deducible with respect to the source analogue and the target instantiation is consistent with respect to the target analogue. Its input includes

- a pair of source (B) and target (A) analogues
(here B = Current and A = FlowRate)

- a problem to solve, PT
(here $PT = \text{“Find the flowrate”}$)
- a theory Th which, by convention, includes
 - many facts about the source analogue, B ,
 - relatively few facts about the target analogue, A , and
 - a collection of tagged relations, the abstractions
(here, Th_{CF} includes **RKK** and its definition).

Its output is

- an abstraction, S ,
- its instantiation in the target domain, $[a_1, \dots A, \dots a_n]$, and
- a list of assumptions which had to be conjectured, $\{\delta_j\}$,
(Following [Fin85], we call this a *residue*.)

where S 's instantiation is

- Derivable with respect to the source domain
 $Th \models S(b_1, \dots B, \dots b_n)$
- Consistent with respect to the target domain
 $Th \not\models \neg S(a_1, \dots A, \dots a_n)$
 $Th \cup \{\delta_j\} \models S(a_1, \dots A, \dots a_n)$
 $Consistent(Th \cup \{\delta_j\})^6$

Figure 4-7 shows an example of *NLAG*'s output for our canonical electricity and hydraulics situation. (The above listing suggests *NLAG*'s inputs; one complete example appears explicitly in Figure 4-8.) In addition to the common abstraction, Figure 4-7 explicates the set of conjectures which *NLAG* postulates, labeled $\{\delta_j\}$, and the binding list used to instantiate the abstraction in the target domain, labeled

⁶The relation $Consistent(\Sigma)$ is true when the sentences in the set Σ are consistent.

4.3. WHY ABSTRACTIONS SHOULD BE USED FOR ANALOGICAL INFERENCE

| | | | |
|-----------------------------|-----------------------------|-------------|--|
| <i>Abstraction</i> | S | = | RKK |
| <i>Target Instantiation</i> | $[a_1, \dots A, \dots a_n]$ | = | [FlowRate, PressureDrop, PipeCharacter, Pipes] |
| <i>Conjectures</i> | $\{\delta_j\}$ | \subseteq | $\left\{ \begin{array}{l} \text{Kirchoff1(FlowRate)} \\ \text{Kirchoff2(PressureDrop)} \\ \text{ConservedThru(FlowRate, Pipes)} \\ \text{Ohms(FlowRate, PressureDrop, PipeCharacter, Pipes)} \end{array} \right\}$ |

Figure 4-7: Components of Analogical Inference — Hydraulics Example

$[a_1, \dots A, \dots a_n]$. The binding list is required because most abstractions are n -ary relations and we initially know the value of only one of its arguments, namely, the parameter occupied by the target analogue. (*NLAG* also finds an instantiation of this abstraction with respect to the source analogue — a binding set $[b_1, \dots B, \dots b_n]$ such that $Th \models S(b_1, \dots B, \dots b_n)$; here that binding set is [Current, Voltage-Drop, Resistance, Resistors]. This is an intermediate, internal calculation.)

We can use the $\Delta(Th, S(a_1, \dots A, \dots a_n), \{\delta_j\})$ relationship to link these various outputs together, where

Definition 13 $\Delta(Th, \sigma, \Sigma) \iff$
 $[Th \cup \Sigma \models \sigma] \ \& \ \text{Consistent}(Th \cup \Sigma).$

Notice this Δ relation is not a function: there may be many different sets Σ which each satisfy this criterion. In fact, this definition permits many undesirable conjecture sets. (For example, it allows $\delta_1 = S(a_1, \dots A, \dots a_n)$, trivializing the entire conjecturing process.) For the definition to be used meaningfully, we need to constrain what qualifies as a residue. Subsection 6.2.3 describes a syntactic requirement for the legal conjecturable propositions which eliminates many of these meaningless cases.

Given:

$AH = \text{"FlowRate} \sim \text{Current"}$

$PT = \text{"Find the flowrate"}$

$$Th_{CF} = \left\{ \begin{array}{l} \text{Kirchoff1(Current),} \\ \text{Kirchoff2(VoltageDrop),} \\ \text{ConservedThru(Current, Resistors),} \\ \text{Ohms(Current, VoltageDrop, Resistance, Resistors), ...} \\ \text{CostPerHour(Current, \$2.43),} \\ \text{Units(Current, Amperes),} \\ \text{Cost(Resistor1, \$3.50),} \\ \text{Color(Wire1, Red),} \\ \text{ModernTreatise(Current, WGilbert, 1600),} \\ \text{InDiscipline(Current, Electric), ...} \\ \\ \text{Kirchoff1(FlowRate),} \\ \text{ConservedThru(FlowRate, Pipes),} \\ \text{Units(FlowRate, Meter3PerSec), ...} \\ \text{Cost(Pipe1, \$2.73),} \\ \text{Color(Pipe1, Red),} \\ \text{InDiscipline(FlowRate, Hydraulics), ...} \\ \\ \text{Domain(Kirchoff1, 1, Functions),} \\ \text{Domain(Kirchoff2, 1, Functions),} \\ \text{Domain(Ohms, 1, Functions),} \\ \text{Domain(Ohms, 4, Classes), ...} \\ \\ \forall t \text{ Kirchoff1}(t) \Leftrightarrow \forall j \sum_{p:Conn(p,j)} t(j,p) = 0, \\ \forall c \text{ Kirchoff2}(c) \Leftrightarrow \forall loop \sum_{\langle i,j \rangle \in loop} c(i,j,[x]) = 0, \\ \forall c, l. \text{ ConservedThru}(t, l) \Leftrightarrow \forall d \text{ l}(d) \Rightarrow [t(j_d^1, d) + t(j_d^2, d) = 0], \\ \forall t, c, r, l. \text{ Ohms}(t, c, r, l) \Leftrightarrow [\forall d \text{ l}(d) \Rightarrow [c(j_d^1, j_d^2, [d]) = t(j_d^1, d) * r(d)] \\ \dots \\ \text{Abstraction(RKK), ...} \\ \text{RKK}(t, c, r, l) \Leftrightarrow \text{Kirchoff1}(t) \ \& \ \text{Kirchoff2}(c) \ \& \ \text{ConservedThru}(t, l) \\ \quad \quad \quad \& \ \text{Ohms}(t, c, r, l) \end{array} \right.$$

Find:

Abstraction: **RKK**

Target Instantiation: [FlowRate, PressureDrop, PipeCharacter, Pipes]

Set of Conjectures: $\left\{ \begin{array}{l} \text{Kirchoff2(PressureDrop)} \\ \text{Ohms(FlowRate, PressureDrop, PipeCharact, Pipes)} \end{array} \right\}$

Figure 4-8: Example of Abstraction-Based Analogical Inference

(These Th_{CF} facts repeat the sentences shown in Figure 2-2, on page 18. The only addition facts are the two final statements about **RKK**.)

4.3. WHY ABSTRACTIONS SHOULD BE USED FOR ANALOGICAL INFERENCE

Figure 4-8 elaborates this specific situation. Here, the initial theory already included the two `FlowRate`-facts, `Kirchoff1(FlowRate)` and `ConservedThru(FlowRate, Pipes)`. This means that *NLAG* needs to add in only the other two facts,

$$\{\delta_1, \delta_2\} = \left\{ \begin{array}{l} \text{Kirchoff2(PressureDrop)} \\ \text{Ohms(FlowRate, PressureDrop, PipeCharacter, Pipes)} \end{array} \right\} \quad (4.4)$$

Notice how the **RKK** abstraction guides the search for the needed conjectures: they are just the “missing parts” of **RKK**’s target abstraction instance. In particular, these $\{\delta_j\}$ are a subset of the “component propositions” of the **RKK** abstraction instantiation in the target domain; *i.e.*, the conjectures shown in Equation 4.4 are a subset of the four propositions shown in Figure 4-7. This means this resistance analogue (in this guise of this **RKK** abstraction) provides a *model*⁷ which suggests which new propositions should be postulated. (Subsection 6.2.3 describes what can qualify as a “component proposition”.)

Possible Benefits: Why are abstractions useful for analogies? As hinted above, they focus the search for new, analogically-inspired conjectures: Rather than search for an arbitrary collections of facts, *NLAG* can consider only some of those sets, only the ones which correspond to an instance of some abstraction. Hence, they represent what Mitchell calls a bias of our learning system, pushing it towards some sets of new facts over others [Mit83].

Why is this necessary? That [Mit83] article noted that every learning system must use some biases to constrain an otherwise infinite search.⁸ Section 2.5 reinforced this concern by demonstrating the immensity of our analogy space: showing that the number of possible analogies is exponential in the number of facts known,

⁷Here we use “model” in the sense of schemata or paradigm, not in the model theoretical sense. See [Hes67] and [Dav85] for a summary of the historical connection between analogy and such paradigms (from the Greek word *παράδειγμα*, meaning “pattern”).

⁸It has been argued that any epistemological system must begin with some biases. The best known argument of this kind is in Kant’s Critique of Pure Reason, wherein Kant deduces that humans have categories by which they interpret the world [Smi68]. This is also addressed in the Whorf-Sapir Hypothesis which states that we “are very much at the mercy of the particular language which has become the medium of expression for [our] society” [Man49, p162]; *i.e.*, the “real world” is built up on our language habits.

and becomes infinite if we are allowed to generate new terms.⁹ These abstractions are explicit biases, used to avoid searching the full space exhaustively. (Section 7.6 makes their “source of power” more concrete, by describing which parts of the search they help channel.)

Our use of abstractions makes this search for useful new facts more efficient. Without any guidance, the best we could do would be a “bottom up” search through an essentially infinite space. (Sections 2.2 and 3.3 discuss this approach, and the data described in Section 7.5 confirms its limitations.)

The use of abstractions turns this into a top-down search. This is a big improvement for two reasons: First, there are usually very few abstractions which pertain to a given pair of analogues and target problem. The other asset is that instantiation is a goal-directed process. Once *NLAG* has selected a pertinent abstraction, the rest of its work — finding the relevant facts which constitute first the source and then the target of the analogy — is essentially a pair of top-down searches. In each case, it seeks the specific facts which *complete* these respective **RKK**-related perspectives. To find the source instantiation, *NLAG* performs a top-down search, trying to prove each fact of this perspective which is not explicitly in the KB. In the target case, it seeks just the particular facts which serve as the residue of a specific formula — again, a very directed process.¹⁰

Possible Limitations: As with any bias, the use of abstractions presents some drawbacks. Permitting *NLAG* to perform top-down searches does provide a greater efficiency, but only for a narrow range of problems. Here, *NLAG*’s coverage is limited to those abstractions which it knows.

In particular, the reliance on abstractions means that an abstraction-based analogy cannot be found unless both **Abstraction(S)** and the definition of **S** are in the initial theory, *Th*. (E.g., both **Abstraction(RKK)** and the definition of **RKK** exhibited in Figure 4-3 must be included in the learner’s starting theory.)

⁹Notice the simplification of using only known vocabulary represents another bias. See Subsection 9.3.2.

¹⁰This means that this learning-by-analogy task can often be reduced to selecting and instantiating the appropriate existing abstraction. Note:4-3 responds to the unfair charge that this renders the overall task trivial.

H_{Abst} has already stated the claim that this restriction prunes away relatively few of the useful analogies. Figure 4-6 illustrates this by showing how thin the difference is between \vdash_{PT} and \Vdash_{PT} . (Section 7.6 explains why.) As analogical inference is, at best, a mechanism of plausible reasoning, eliminating this $\vdash_{PT} - \Vdash_{PT}$ sliver is not that serious a drawback (see Note:1-2).

4.4 Why Abstractions *Can* be used for Analogical Inference

The previous section demonstrated why it is efficient to use abstractions for analogical inference. This efficiency is worthless if the needed abstractions do not exist or if the resultant conjectures are not reasonable. This section addresses the question of why this use of abstractions is possible, illustrating that [1] these abstractions do occur in many different domains and [2] they do tend to suggest plausible conjectures. (Another requirement is that each abstraction appears in a re-usable form. This is trivial, as each abstraction is a relation with several formal parameters. Hence, one can “re-use” an abstraction instance by re-instantiating the abstraction with other terms.)

Section 4.1 suggested that each abstraction is a model, embodying the cohesive cluster of facts needed to produce a useful analogy. We consider these clauses *cohesive* because they work together to yield solutions to some problems. As an abstraction helped to solve a problem in one domain, we intuitively believe that it might work again in a new domain. That is, an abstraction provides a *predictive bias*.

This claim is easy to justify in artificial domains, in which designers construct artifacts. There we can assume that the human designer has built in certain features, probably for cognitive simplicity or efficiency. In particular, we know that two artifacts designed from the same plan will share a great many features — *viz.*, those derived from this original plan. Consider, for example, a pair of sorting programs

based on the same specification, or houses built using the same architectural floor-plan, or dishes prepared using the same recipe.

This argument extends to “artifactual properties” of natural objects, for example, their names. This is why we expect diseases with similar names to exhibit similar manifestations; we can assume that the “name-designer” assigned the names on this basis.

It also seems true that objects in diverse natural domains share many similarities. Proving this intuition requires a strong inductive leap. It requires assuming that facts which hold in one domain will also hold in another. The rest of this section presents a number of examples which reinforce this claim.¹¹

As one example, there are many places where some quantity is conserved at every junction of devices — e.g., current and liquid flow-rate as well as force, torque and heat flow-rate. In each of these, there is also another term which is invariant around each loop of devices: we have already mentioned Pressure and Voltage, now add velocity, angular velocity, and temperature. (See Table 4-1.)

As a different example: remember when you learned that matrices were closed under matrix addition. It then made sense to ask about things like inverses and identities. Why? Well, those properties held for the integers, reals, complex numbers, *etc.* They appeared together again when considering rotations of a cube (or even Rubic’s cube); and again, in particle physics. So, we argue, why not here as well?

The *Program Plans* and *clichés* used in the Programmer Apprentice project [Ric81,Wat71,Bro81,Ric79] are exactly the *general re-usable clusters* we call abstractions.

This re-usability does not pertain only to hard scientific models. The case frames used to understand natural language provide one common use of abstractions (*cf.*

¹¹Unfortunately, it appears that the only way to justify this claim is empirical; by observing that the world seems to have this nice continuity property. This is clearly related to Hume’s famous dilemma [Hum02]; see Note:4-4.

Note:4-5 is also relevant, as it explains why I do not care whether this continuity is an ontological fact, relating to the true nature of the world, or epistemological, dealing only with our theories of the world.

2.7.2-1 LIST OF ANALOGOUS VARIABLES AND ELEMENTS

| | DISCIPLINE | INTEGRATED THROUGH VARIABLE | ACROSS VARIABLE | INTEGRATED ACROSS VARIABLE | ELASTIC OR THROUGH STORAGE ELEMENT K' | CAPACITIVE OR ACROSS STORAGE ELEMENT C' | ENERGY DISSIPATION ELEMENT D' |
|-------------|--------------------------|-----------------------------|--|------------------------------------|---|---|---------------------------------|
| Translation | Force F | Momentum Mv | Velocity difference Δv | Displacement difference ΔX | Translatory spring K | Mass M | Damper D |
| Rotation | Torque T | Angular momentum $J\omega$ | Angular velocity difference $\Delta\omega$ | Angular difference $\Delta\theta$ | Torsion spring K_t | Moment of inertia J | Torsional damper D_t |
| Electrical | Current i | Charge q | Voltage difference ΔV | Flux linkage $\Delta\lambda$ | Reciprocal inductance $1/L$ | Electrical capacitance C | Reciprocal resistance |
| Fluid | Volumetric flow rate Q | Volume V | Pressure difference ΔP | Pressure momentum $\Delta\Gamma$ | Reciprocal inductance $1/I$ | Fluid capacitance C | Reciprocal resistance |
| Thermal | Heat flow rate q | Heat flow B | Temperature difference ΔT | — | — | Thermal capacitance C | Heat conduction or dissipation |

Table 4-1: Other Examples of Lumped Element Linear Systems (from [Coc80, p80])

[vGDB81, esp C4] and [Fil68]).

As another example, Lakoff and Johnson have a large compendium of other instances of this abstraction idea, taken from day to day speech [LJ80]. For example, notice that “height” has the nice property *that every two heights are comparable*. When we start comparing emotions, like happy to sad, we see that it, too, embodies a linear ordering — and so we talk about

“feeling *up* in the clouds” or
 “being *down* in the dumps”,

as well as

| | |
|--|------------------------------------|
| I’m feeling <i>up</i> . | He’s really <i>low</i> these days. |
| That <i>boosted</i> my spirits. | I <i>fell</i> into a depression. |
| My spirits <i>rose</i> . | My spirits <i>sank</i> . |
| You’re in <i>high spirits</i> . | I’m feeling <i>down</i> . |
| Thinking about her always gives me a <i>lift</i> . | I’m <i>depressed</i> . |

Those authors went on to observe other systematic uses of this linear ordering:

Conscious is up
 Health and life are up
 Having control is up
 More is up
 Forseeable future events are up
 High status is up
 Good is up
 Rational is up

Their book also describes many other systematically used hidden metaphors. (Their “time is money” description is one of their best.) This idea is so convincing that many others have followed suit, including [Red79], [Car81b,CM83] and [Hob83a,Hob83b].

The list in Section 1.1 includes yet other inter-domain connections. Table 4-2 shows that many of those analogies are based on well-known abstractions as well.

What is the impact of this use of abstractions? While this section has focused on their utility in natural domains, I feel this application represents only the tip of the iceberg. These abstractions may have far more applications in artificial domains, in situations where we know the artifacts are designed. As mentioned above, people often follow the same blue-print for several different items; for example, designing one operating system in the same mold as a previous one, or modeling jets after airplanes,¹² or naming conventions, *etc.* In general, there are cognitive reasons for not “re-inventing” the wheel, and instead re-using some existing idea, *mutatis mutandis*.¹³ These are exactly the conditions needed for abstractions to work.

Goal: This is a good point to repeat the goals first mentioned in Section 1.4: this dissertation is solely concerned with the challenge of deriving useful analogies

¹²They resemble one another structurally (*i.e.*, each has a pair of opposing wings perpendicular to the fuselage), “internally” (*i.e.*, each has a set of stewardesses serving the passengers) and behaviorally (*i.e.*, in both cases, the passengers first buys tickets before boarding), *etc.*

¹³Of course, in some cases we could claim that the world forces these constraints, as in deriving the shape of a jet plane from that of airplanes. For this research, this is irrelevant; it is enough to observe that designed artifacts do exhibit clusters of related features. See Note:4-5.

| |
|---|
| <ul style="list-style-type: none"> • “Electric Analogues” (e.g., RKK, RCKK ...) Hydraulics, Electricity, Thermal, Translation, ... • Algebraic Structures (e.g., Group, Ring, Field, ...) Matrix, Numbers, Function spaces, Rotations, ... • Program clichés (e.g., Aggregation Loop) Sum of Vector, Product of List • Information Processing Systems People, Computers • Linear ordering Happy, Up, Good, Quantity, ... • Domain principles (e.g., Anticipate Drug toxicity) Cardiomyopathy, Hypercalcemia • Naming Conventions Chemical Nomenclature, Names of Fonts, Related Diseases, ... |
|---|

Table 4-2: Abstractions used for Useful Analogies

based a pre-specified set of known abstractions. The task of generating additional abstractions on the fly, guided by the goal of finding the analogy which helps to solve this problem, is a related but different line of research. Fascinating as this possibility is, *n.b.*, it is not the topic of this research. We assume, instead, that these useful generalizations have already been accumulated.

Fortunately, abstractions exist all around us, and the scholars of the ages have recorded just such information. As suggested above, there are books on abstract algebraic structures (e.g., [Her64], **Group**), on lumped linear systems (e.g., [Coc80], **RKK**), on program clichés (e.g., [Ric79], **Aggregation-Loop**), *etc.* Our goal is to capitalize on this large corpus of stored information. *NLAG* does so by viewing each abstraction as a useful heuristic, each claiming that a particular cluster of facts will continue to fit together, even in new domains.

4.5 Comparison of Abstractions

In many respects, the concept of abstractions is not new. AI researchers have long considered clustered assemblages of related facts to be useful — consider the literature on frames ([Min75], [BW77], [Bra79], [RG77], [SF80], [SGLS80], *etc.*), scripts ([SA77]), program plans ([Ric79,RW81]), scenarios ([WR82]), and dating back even to LISP's property-list construct (see [MAE*62]). Indeed, they have even been utilized in a prior analogy related work, in Merlin's beta-structures [NM73]. The psychology literature also abounds with this idea, indicating that people are believed to employ this facility internally as well (*cf.*, [AB73]).

Abstractions, however, differ from the other clustering ideas in one important aspect: in how the cluster is indexed and accessed. In each of the other systems, there is but a single access point into each cluster of facts — through perhaps the name of the unit or the lisp atom or a description of the activity. This facilitates the most typical retrieval task, of finding the facts associated with that specific concept. On the other hand, it is difficult to access this bundle of facts in any other manner, rendering this information all-but-unavailable to other related objects, or even to the other terms included in these sentences. Clever tricks — like Generalization (Tangled) Hierarchies [Fin79], Event Units [Nil80], or even elaborate schemes like Partioned Semantic Nets [FH77] — somewhat lessen these losses. Each of these is still limited; each is only able to handle those situations for which it was explicitly coded.

An abstraction, being a flat collection of facts, can have an arbitrary number of “keys” — meaning that such a cluster can have more than one entry point. (The plans used in the Programmer's Apprentice system are similar; each can be accessed via any of its (multiple) internal rôles [Ric79]. See also Sussman's notion of “Slices” in [SGLS80].) Furthermore, a single concept can be associated with many clusters. Consider, for example, the **RKK-EC** cluster of facts discussed in Section 4.1 above. There, it was found via the **Current** entry, together with facts related to the “Find the Current” problem. We could have found this cluster from other

starting points as well — from VoltageDrop for example, or from the facts associated with a “Find the resistance in this series circuit” problem. The other important realization is that there are other clusters which could have been found from this same Current concept. (Section 4.1 elaborates this objective, and Subsection 6.1.3 presents one possible implementation.)

Many of the tasks which seemed awkward from the “single focus” view-point, fall out once one adopts this attitude. As one example, this is probably why Schank both considers retrieval to be a complex indexing task, and regards it as a critical objective in understanding intelligence [Sch85]. The general process of understanding an analogy, however, is the canonical example. Much of the apparent “fuzziness” associated with the way information is transferred (as the learner understands the analogy) can be attributed to the view that a given concept can be associated with but a single cluster of relevant facts — which follows perforce from this single focus assumption. But given this multiple-focus perception and these abstractions, we saw above how straightforward and logical this process is. (See also Note:9-2.)

There is, of course, a cost for this generality: the task of retrieving a cluster from one (or more) of its keys is quite difficult, certainly more challenging than a *getprop*. Indeed, this indexing problem — of going from various symbols and facts to the pertinent abstraction — is a major issue in this research. *ComAbs*’s first two steps, *Find-Kernel* and *Inst-Source*, represent one approach. (See Section 6.1.)

As a final note, realize that most of the current research in analogy deals with some notion related to abstractions; *cf.*, [Gen80b], [Car81b], [WBKL83], [Hob83a], [Ked85], *etc.* Each of these is quickly sketched in Section 3.4 and elaborated in Section 9.4.

Chapter 5

Ranking Analogical Inferences

Chapters 2 and 3 defined the useful analogical inference process and demonstrated that its search space is intractable. Chapter 4 then discussed how the use of abstractions can focus the search. Unfortunately, this is not sufficient to eliminate search; there may still be many possible common abstractions. This chapter suggests several further criteria which can be used to rank these different analogies. The next chapter shows how these rules are used to generate the *a priori* better analogies first.

Section 5.1 sets the stage by posing the underlying question and describing the general framework of the answer. The next two sections present two sets of heuristics. Section 5.2 describes rules which use the problem statement and context to focus the search. Section 5.3 motivates and describes rules based on the “least constraint” maxim. Section 5.4 presents a summary of all of the heuristics used to constrain the legal analogies and describes their interactions.

5.1 Framework for Ranking Analogies

Given any theory, analogues and target problem, there may be many possible legal analogies, each composed of an abstraction and a target instantiation which together satisfy the requirements given in Figure 4-5. One objective of this research is to determine, *a priori*, which of these legal analogies is most likely to be “correct”

or “meaningful”. *NLAG* uses this information to decide which of these analogies should be considered and in what order.

Stated more precisely: we are given a pair of analogies, $\varphi_1(A)$ and $\varphi_2(A)$,¹ where

$$\begin{aligned} Th, A \sim B & \parallel_{PT} \varphi_1(A) \\ Th, A \sim B & \parallel_{PT} \varphi_2(A). \end{aligned} \tag{5.1}$$

The goal is to determine whether $\varphi_1(A)$ is “better than” $\varphi_2(A)$: that is, which is a more likely extension to the initial theory *Th*, given the meta-level knowledge that this hint is given to help solve the given problem. (It turns out that the analogies suggested by this criterion can be found efficiently. This is serendipitous as efficiency was not an original objective.)

The previous chapter discussed one important heuristic: restrict the φ_i to abstraction formulae. This has already been incorporated in the definition of abstraction-based analogies presented in Figure 4-5.

This chapter discusses several other heuristics, separated into two categories. The first group, described in Section 5.2, consists of rules which apply only when considering abstraction-based analogies. These two rules use the problem statement and context (respectively) to focus the search for desirable \parallel_{PT} analogies.

Section 5.3 discusses the “Least Constraint” maxim, an embodiment of the *I_{Least}* intuition presented in Section 3.4. Basically, this maxim prefers “closer” analogies: i.e., it favors $\varphi_1(A)$ over $\varphi_2(A)$ if $\varphi_1(A)$ is “closer” to the currently known world than $\varphi_2(A)$. Section 5.3 provides a semantic basis for this single idea and discusses three heuristics used to implement it. Since these rules implement a method of focusing general analogical inference, they have a greater utility than the ones discussed in Section 5.2. (Recall the latter rules pertain only to abstraction-based analogical inference.)

Section 5.4 summarizes the six rules presented in this dissertation (in this chapter and the last) and discusses how this total collection of rules interact. The data presented in Chapter 7 further demonstrates their synergism.

¹Two notes: [1] This chapter is concerned with ways of comparing given analogies, not with generating them. [2] Of course, analogical formulae are usually *n*-ary. We initially describe them as unary formulae for pedagogical reasons; this chapter deals with the more general case later.

Chapter 6 is a continuation of this chapter in two ways. First, this chapter only sketches how each rule is implemented; Chapter 6 provides the details. Secondly, this chapter's rules do not completely specify an abstraction-based analogical inference process: in particular, they do not completely specify the order in which certain inferences should be made. The additional rules described in Chapter 6 complete the partial ordering. There is an important distinction between the rules of this chapter and the next, though. The rules shown in this chapter are part of the definition of abstraction-based analogical inference. By contrast, the rules suggested in the next chapter are but nuances of my *NLAG* implementation, often arbitrary decisions made to handle certain situations.

5.2 Abstraction-Based Heuristics

This section presents two heuristics and discusses their consequences. Both pertain only to the abstraction-based analogical inference process: *i.e.*, each heuristic depends critically on the fact that the analogy formula is an abstraction formula. (Hence, each rule can be considered a modification of the overarching rule, H_{Abst} .) The first, H_{JK} , provides a way of using the target problem to find the relevant abstractions. The second, H_{CC} , insists that all the arguments of an abstraction belong to a single context.

(These rules may conflict with the rules presented in the next section. In all cases, these rules take precedence. Section 5.4 explains why.)

5.2.1 H_{JK} : Justification Kernel

“Look at the unknown! And try to think of a familiar problem having the same or a similar unknown.”

How to Solve It, Polya (1957)

This rule uses the problem statement as a guide to determine which abstractions should be considered and in what order. When it applies, it both reduces the space of eligible abstractions and provides an ordering for those which remain. (*I.e.*, it is both an ordering and a pruning heuristic.)

In a nutshell, H_{JK} uses the target problem to suggest a related query in the source domain, then uses the support for this source query to suggest which abstractions may be relevant. The first problem is finding an appropriate query in the source domain. Ideally, it should be just like the given target problem, but in the different source domain. For example, from the target problem $PT =$ “Find the flowrate in a given hydraulics system”, H_{JK} finds the analogous source domain problem, $PS =$ “Find the current, in a [related] electric circuit”. More precisely, it derives $PS = (\text{current } ?1 ?2 ?3 ?4)^2$ from $PT = (\text{flowrate } j\text{-wc- a pipe1 s0 } ?fr)$ by lexically replacing the source analogue with the target and turning the other constants into variables.

The H_{JK} rule then considers how to solve this analogous source query, PS : in particular, it determines which facts would be relevant. This corresponds to the support of this query from the initial theory. These are the “kernel facts”, written $\Sigma = \mathcal{ST}_{Th}(PS)$. (Definition 8 on page 31 defines the $\mathcal{ST}(\dots)$ support operator.)

The members of Σ include a *trace* of the facts which would be used to solve this problem. Now recall Section 4.1’s claim that the abstractions themselves were originally formed from just such problem solving traces! This means that Σ contains the facts which would have led to the formation of the relevant abstractions in the first place, which suggests that we consider the abstractions derived from these support facts.

To illustrate this, consider how a backward-chaining system would address the “Find the current” query. It would use rules of the form

$$(\text{If } \langle \text{ante} \rangle (\text{Current } \dots)), \quad (5.2)$$

which, in turn, is derived from the fact (Kirchoff1 Current) together with the rule

$$(\text{If } (\text{Kirchoff1 } ?t) (\text{If } \langle \text{ante} \rangle (?t \dots))). \quad (5.3)$$

This means the support for this source query, $\Sigma = \mathcal{ST}_{Th}(\text{current } ?1 ?2 ?3 ?4)$, includes this (Kirchoff1 Current) fact. (I.e., (Kirchoff1 Current) $\in \Sigma$.) This

²Recall each $?i$ term denotes an existential variable. Also, some clauses are written in LISP’s s-expression notation rather than the other functional form; e.g., “(current ?1 ?2 ?3 ?4)” rather than “current(?1 ?2 ?3 ?4)”.

derivation similarly depends on the other **RKK**-related facts, including (Kirchoff2 VoltageDrop) and (Ohms Current VoltageDrop Resistance Resistors).³

We see that the support set Σ contains many of the facts associated with instances of the relevant abstractions. In fact, we can approximate how much this problem has in common with an abstraction instance by measuring the size of the intersection of its support set with that source kernel, preferring abstraction instances whose supports include many elements in common with Σ . This suggests that we seek the abstraction, \mathbf{S} , which maximizes

$$\mathcal{ST}_{Th}(PS) \cap \mathcal{ST}_{Th}(\mathbf{S}(b_1, \dots, B, \dots, b_n)). \quad (5.4)$$

This is the sense of the H_{JK} rule:

Heuristic 2 H_{JK} : Using $\Sigma = \mathcal{ST}_{Th}(PS)$,

(i) Prefer \mathbf{S}_1 over \mathbf{S}_2 if

$$\|\Sigma \cap \mathcal{ST}_{Th}(\mathbf{S}_1(c_1, \dots, B, \dots, c_n))\| > \|\Sigma \cap \mathcal{ST}_{Th}(\mathbf{S}_2(d_1, \dots, B, \dots, d_m))\|$$

(ii) Avoid \mathbf{S}_1 if

$$\Sigma \cap \mathcal{ST}_{Th}(\mathbf{S}_1(c_1, \dots, B, \dots, c_n)) = \{\}.$$

where \mathbf{S}_1 and \mathbf{S}_2 are each abstractions and $[c_1, \dots, c_n]$ and $[d_1, \dots, d_m]$, their respective binding list. (For each abstraction, we use the binding list which maximizes the size of the intersection.)

An example may help explain this ordering rule. Consider again the electricity and hydraulics situation. When should the **RKK** abstraction be considered before **KK**? H_{JK} claims this should depend on the relative sizes of Ψ_1 and Ψ_2 , where

$$\begin{aligned} \Psi_1 &= \Sigma \cap \mathcal{ST}_{Th}(\mathbf{KK}(\text{Current}, \text{VoltageDrop})) \\ \Psi_2 &= \Sigma \cap \mathcal{ST}_{Th}(\mathbf{RKK}(\text{Current}, \text{VoltageDrop}, \text{Resistance}, \text{Resistors})). \end{aligned} \quad (5.5)$$

³In fact, this derivation actually uses all four of the **RKK**-facts. This is a further corroboration of my belief in the coherence of abstractions.

As a first observation, notice that $\Psi_1 \subseteq \Psi_2$, as

$$\begin{aligned} \mathcal{ST}_{Th}(\text{KK}(\text{Current}, \text{VoltageDrop})) \subset \\ \mathcal{ST}_{Th}(\text{RKK}(\text{Current}, \text{VoltageDrop}, \text{Resistance}, \text{Resistors})). \end{aligned} \quad (5.6)$$

(That is, any fact which supports an instance of **KK** must also support the more specific **RKK** instance.)

Now suppose the containment is proper, i.e., $\Psi_1 \subset \Psi_2$. This means that some fact in $\Psi_2 - \Psi_1$ is used to support *PS*. (For example, perhaps the electric Ohms law is needed to solve *PS*.) As **RKK** has more “parts” which contribute to solving the source problem than **KK**, we feel **RKK** is a better common abstraction than **KK**, and so should be considered first.

This example deals with abstractions whose support sets are tightly related; Equation 5.6 shows that one is contained in the other. In general, a given Σ support set might intersect the support sets of many diverse abstraction instances. H_{JK} handles these cases by comparing $\|\Psi_1\|$ with $\|\Psi_2\|$, rather than just Ψ_1 and Ψ_2 . (I.e., it deals with the cardinality of the intersection of these sets, rather than with the particular facts in that intersection.)

Of course, this definition is meaningless unless we constrain which facts can qualify for membership in these support sets. We address this problem by considering only the *leaf clauses* defined in Subsection 6.2.3.

The second part of Heuristic 2 indicates that H_{JK} also prunes away the extreme points of this measure. It tells *NLAG* not to consider an abstraction **S** if none of its instances are supported by any member of *PS*'s support: i.e., where

$$\Sigma \cap \mathcal{ST}_{Th}(\mathbf{S}(b_1, \dots, b_n)) = \{\}.$$

Why? Finding the intersection empty means that this abstraction includes no facts which apply to this problem. (Of course, this requires the assumption that every fact in the initial theory points back to its justification. Otherwise, if this “justification tie” has not been established, the system might miss an applicable abstraction.)

We close this subsection with some final comments. First, realize that this rule is only a heuristic: There is no intrinsic reason why the lexically derived *PS* should

have anything in common with the original PT . Had we worded the PT target query differently, this substitution might have produced a totally worthless source query.

Secondly, H_{JK} is not always applicable. In particular, for certain hints and target problems, there may be no related source problem. Consider this same “Find the flowrate” target problem, but imagine we were given a different pair of analogues, e.g., PressureDrop~VoltageDrop rather than FlowRate~Current. Here, there is no corresponding PS . In these cases, $NLAG$ uses a starting kernel which consists of all sentences which lexically include the source analogue. (This is $\Sigma = \mathcal{P}_A(Th)$, using the notation defined in Note:5-1. That note also proves that this set is adequate: i.e., that it finds all abstraction instances.)

Thirdly, this rule constrains which abstraction formulae may be used but says nothing about how to re-instantiate this abstraction using the target analogue.

Fourth, as the current version of H_{JK} deals only the *cardinality* of the intersection of the support sets, $\|\mathcal{ST}_{Th}(PS) \cap \mathcal{ST}_{Th}(S_1(c_1, \dots, B, \dots, c_n))\|$, it is, implicitly, assigning equal weight to all component facts. Later versions may weigh different facts differently, to encode how important each fact is towards establishing its associated abstraction. The current unweighted form proved sufficient for all of the examples I tried.

Finally, $NLAG$ operationalizes this H_{JK} heuristic by using this set of support facts, $\Sigma = \mathcal{ST}_{Th}(PS)$, as a starting kernel. It then applies a forward-chaining scheme to find the abstraction instances which follow from this set, ordered by the number of involved kernel fact. (This also means that our version of H_{JK} resolves ties in the order of fewest-deductions first: see Note:3-1.) Subsection 6.1.3 elaborates this description.

5.2.2 H_{CC} : Common context

This heuristic is a pruning rule. When instantiating the abstraction in the target domain, H_{CC} instructs $NLAG$ to consider only n -tuples of terms which all come from the same context. This idea is best introduced by example:

Figure 4-3 shows that the only requirements **RKK** places on its second argument, $?2$, is that it satisfy (Kirchoff2 $?2$) and (Ohms $?1 ?2 ?3 ?4$). Notice this permits the $?2 \mapsto \text{VoltageDrop}$ assignment. In fact, there is a good reason to consider this possibility, as (Kirchoff2 VoltageDrop) alone provides strong support for VoltageDrop. Of course, the other proposition, (Ohms FlowRate VoltageDrop PipeCharacter Pipes), is not known to be true. This is easily surmounted, as the \Vdash_{PT} process is allowed to simply postulate this assertion. (Recall that \Vdash_{PT} is *expected* to postulate clauses it cannot prove.)

The only reason this $?2 \mapsto \text{VoltageDrop}$ assignment seems wrong to us is because we know a bit more about VoltageDrop and FlowRate. In particular, we know that VoltageDrop deals with electricity while FlowRate deals with hydraulics. As one solution to this predicament, we could insist that each abstraction include an explicit clause which forces all of its terms to be associated with the same “context”. For example, we could use an embellished **RKK_{CC}** relation rather than **RKK**, where **RKK_{CC}** includes an additional clause which insists that all of its instantiated terms belong to the same sub-theory: *i.e.*,

$$\begin{aligned} \mathbf{RKK}_{CC}(t, c, r, l) &\iff \\ &\mathbf{RKK}(t, c, r, l) \ \& \ \text{SameTheory}(t, c, r, l). \end{aligned} \tag{5.7}$$

where the additional SameTheory clause guarantees that all of **RKK_{CC}**’s arguments come from the same domain — here, either all from electricity or all from hydraulics. As SameTheory(FlowRate, VoltageDrop, r, l) is false, **RKK_{CC}** would not consider the $?2 \mapsto \text{VoltageDrop}$ instantiation.

I choose, instead, to effect this “same context for all arguments” rule at the meta-level, using the *H_{CC}* rule:

Heuristic 3 *H_{CC}*: Avoid $S(a_1, \dots, A, \dots, a_n)$ unless
 $\forall j \text{ TheoryOf}(A) = \text{TheoryOf}(a_j)$.

This means we only consider a proposed target instantiation if all of its arguments belong to the same context.

This H_{CC} rule is basically a convenience, one which spares us the need to generate a more complex definition for each abstraction. (It is implemented using MRS's theory context mechanism, see [Rus85, Chapter 9]. Section 6.1.5 provides more details.)

This rule may be justified by observing that all members of an abstraction's n -tuple of concepts tend to come from the same theory. This follows from the observation that this co-participation in "reasonable" relations (read "abstractions") is partly what defines and separates different theories.

5.3 Least Constraining Analogies

This section motivates, defines and operationalizes the "least constraint" maxim. Stated simply, this maxim favors the analogy which imposes the fewest additional constraints on the currently known world: *i.e.*, it corresponds to the I_{Least} intuition of Section 3.4. Subsection 5.3.1 first discusses this idea intuitively, emphasizing how it fits into our framework of useful analogies. Subsection 5.3.2 then proposes a syntactic criterion. When that attempt fails, Subsection 5.3.3 successfully provides a correct semantic account. Subsections 5.3.4 through 5.3.6 present three operational heuristics which follow from this single idea.

5.3.1 Motivation for the Least Constraint Maxim

All of the heuristics presented until here deal with abstractions, realizing the I_{Close} intuition presented in Section 3.4. That section also claimed that we should prefer the minimal extension; suggesting that it is sufficient to consider the abstraction which is "closest" to the known world. This subsection describes this "closeness" intuition and argues why it is applicable to abstraction-based analogies.

Intuitively, we feel a conjecture is "*close*" to the known world if it imposes few additional constraints. In the analogy context, this closeness measure describes how likely the analogy formula is to hold for the target analogue.⁴ It is measured by

⁴We say *an abstraction holds for a concept* if there is some instantiation of the abstraction including that concept. *E.g.*, "Group holds for +", as $\text{Group}(\mathcal{R}, +, 0)$ holds.

considering *how much more we must require of the world* for this target sentence to hold: the fewer additional requirements must be made, the closer this abstraction is to holding for the target analogue.

Now to make that “*how much more we must require of the world*” comment more precise. Consider what is required for the target instantiation, $\varphi(A)$, to hold. At one extreme, $\varphi(A)$ may already be true; i.e., $Th \models \varphi(A)$. This means the world already satisfies this requirement; nothing else is required. Otherwise, if $Th \not\models \varphi(A)$, we have to impose some additional constraints on the world before $\varphi(A)$ can hold. This amounts to finding some independent sentence, δ , such that $Th + \delta \models \varphi(A)$. The rest of this subsection illustrates this notion and describes how it can be used to compare different analogies. The next several subsections attempt to quantify this measure.

Imagine first that we know nothing about the target analogue, A . Now ask whether $Foo(A)$ is more likely to hold than $Bar(A)$. Knowing nothing about A , the choice seems arbitrary. The best we can do is consider the Foo and Bar predicates; and compare their *a priori* likelihoods. (As an obvious example, imagine $Foo(x) \Leftrightarrow Even(x)$ and $Bar(x) \Leftrightarrow 3 < x < 6$, where the universe of discourse is the natural numbers. Here, $Foo(A)$ is clearly more likely to be true than $Bar(A)$, since there are many more even numbers than natural numbers between 3 and 6.) This means that the chance that any particular formula involving A holds should reduce to the *a priori* likelihood of that formula.

This was the simple case. What if we know something about the target analogue? With more information, A should “move” closer to some formulae and, relatively speaking, further from others. (The discussion around page 88 makes this more precise.)

The least constraint maxim suggests that we work first on the formula which is nearest to becoming complete. The overall idea corresponds to the I_{Least} intuition. Now to address Section 3.4’s claim that this constraint is especially powerful when coupled with the I_{Close} insight. Recall I_{Close} claims that we should seek cohesive clusters of facts, where a collection of facts is considered cohesive if a subset of these facts suggest the others. Hence, the I_{Close} intuition claims that observing that some

facts hold suggests that the other facts in that cluster might hold as well. More quantitatively, we can consider how many of a cluster's facts are known to hold initially. Intuitively, the more that hold initially, the greater the chance that the remaining facts (those which complete this closure) will hold as well.

Chapter 4 made this concept of clusters more concrete by proposing that each abstraction instance is such a complete cluster of facts. We can now assemble the I_{Least} and I_{Close} intuitions: together they suggest that we first consider the abstraction which is closest to being complete; i.e., we seek the abstraction instance which imposes the fewest additional requirements on the world. As another way of thinking about this, this pair of intuitions state that a collection of facts is useful if it contains just enough to solve some problem (i.e., if it corresponds to some abstraction) and no more (i.e., if it is close to the known world).

So far, we have considered only the question of which facts should be added to a theory, based on which related facts already hold. Why is this relevant for analogies? It provides a means to determine which analogy (read "common abstraction") is most likely to hold, as the likelihood that the same abstraction has instances in both the source and target domain depends on how much the two analogues share *a priori*, with respect to that abstraction. (I.e., it depends on how many of the abstraction's component facts⁵ already hold for the analogues.) As the source abstraction instance already holds completely, this measure reduces to considering how much is required for the target abstraction instance to hold. Hence, the less we have to add to the initial theory to obtain the target instantiation, the more the two analogues must have shared initially. The implicit claim is that the more these two analogues have in common *a priori*, the more likely the resulting analogy is.

In summary, this least constraining maxim suggests that we work first on the formula whose target instance is nearest to being complete. While this measure is well-defined for arbitrary formulae (and hence arbitrary analogies), it is more useful when coupled with the H_{Abst} rule — i.e., when used to rank common abstractions. Due to the coherence of the abstraction's clauses, this measure reflects the feasibility

⁵The proposition, σ , is a component fact of the relation, R , if it is a member of R 's leaf decomposition; i.e., if $\sigma \in \Gamma$ for some Γ satisfying $LD(Th, R(\dots), \Gamma)$. This uses Definition 20 from Subsection 6.2.3.

of the “completed” analogy (i.e., “common abstraction”).

This is consistent with Section 3.4’s claim that I_{Least} is appropriate when dealing with coherent sets of facts; i.e., as a refinement of I_{Close} . That is, a useful analogy need only convey the information needed to solve the specific problem. As each abstraction encodes such information, finding the “closest” abstraction is sufficient. This also explains why these I_{Least} -based heuristics take a back seat to the I_{Close} -based ones. (See Section 5.4.)

5.3.2 Syntactic Criterion: Additions to the Theory

This subsection attempts to express this I_{Least} intuition *syntactically*. Section 4.3 indicated that each analogical inference requires that a set of conjectures be added to the initial theory. Perhaps we should simply count the number of conjectures required and use that number as a measure of how constraining the proposed abstraction instance is. The rest of this subsection defines, and then refutes, this “*the fewer the number of conjectures, the better*” proposal.

As our framework, consider

$$\begin{aligned} Th, A \sim B \Vdash_{PT} \varphi_1(A) &\sim Th + \Delta_1 \models \varphi_1(A) \\ Th, A \sim B \Vdash_{PT} \varphi_2(A) &\sim Th + \Delta_2 \models \varphi_2(A) \end{aligned} \quad (5.8)$$

where each Δ_i is a collection of conjectures; i.e., $\Delta(Th, \varphi_i(A), \Delta_i)$. This proposal advocates choosing φ_1 over φ_2 if Δ_1 has fewer conjectures than Δ_2 :

Definition 14 *LessConstraints* $_{syn}(\varphi_1(A), \varphi_2(A))$ iff $\|\Delta_1\| < \|\Delta_2\|$.

Stating this proposal precisely requires quantifying the number of conjectures. Unfortunately, this is rather difficult if not altogether meaningless. For example, consider Kirchoff’s two laws which deal with through and cross terms. By this measure, it makes a difference whether we express this as a single fact or in an expanded form, as a pair of facts. That is,

$$\begin{aligned} \|\{\text{KK}(\text{FlowRate}, \text{VoltageDrop})\}\| &= 1 \\ \|\{\text{Kirchoff1}(\text{FlowRate}), \text{Kirchoff2}(\text{VoltageDrop})\}\| &= 2, \end{aligned} \quad (5.9)$$

even though

$$\forall t c \text{ KK}(t, c) \iff [\text{Kirchoff1}(t) \ \& \ \text{Kirchoff2}(c)]. \quad (5.10)$$

Even worse, realize that we can satisfy Equation 5.8 by simply adding in the desired fact, *i.e.*, let $\Delta_1 = \{\varphi_1(A)\}$. This means that $\|\Delta_1\| = 1$. As this is true for any φ_i (*i.e.*, $\|\Delta_i\| = 1$ for any possible φ_i), this *LessConstraints_{syn}* measure is meaningless.

The underlying problem is the assumption that every conjecture should be weighted the same; *i.e.*, the implied *One Conjecture, One Vote* policy. Though some conjectures clearly contribute more, there is no way of stating this syntactically. Fortunately, it is possible to state this semantically.

5.3.3 Semantic Criterion: Constraints on Possible Worlds

This subsection discusses a semantic criterion for this *I_{Least}* idea, in terms of sets of possible worlds. This requires a brief digression to present the basic framework.⁶ We employ this semantic system when comparing conjectures, using it to determine which conjecture imposes the least constraints on our known model of the world. This leads to a measure which *I_{Least}* can use.

We begin with some partial knowledge of the world. In particular, we assume that we know the identity of all of the objects, but not all of the relationships: *i.e.*, each constant symbol is completely specified, while arbitrary relation symbols may be only partially specified. Making this precise requires assuming a pre-defined ontology, based a particular fixed lexicon and a known and finite universe of objects. Within these assumptions, we can consider the class of worlds⁷ which are consistent with our limited knowledge.

⁶This is based on the partial interpretation model presented in Section 8.2. The reader has the option of regarding the following arguments as intuitions, or of first reading that section. In the former case, the reader should ignore the subsequent footnotes, as they serve only to reword the text into Chapter 8's more precise notation.

⁷This dissertation considers the terms "world", "model" and "(partial) interpretation" to be equivalent.

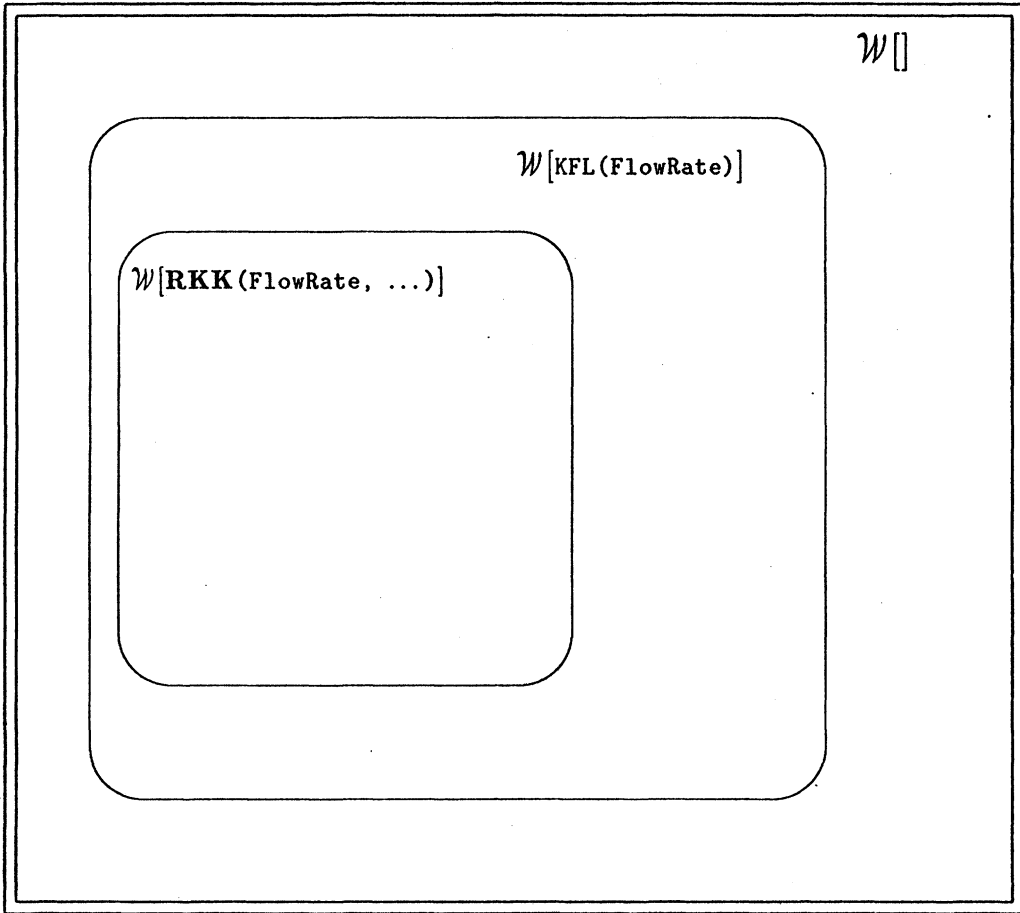


Figure 5-1: Possible Worlds

Legend:

K#1 ↔ Kirchoff1

K#2 ↔ Kirchoff2

RKK(FlowRate, ...) ↔ **RKK(FlowRate, PressureDrop, PipeCharacter, Pipes)**

Figure 5-1 shows an example of this. Every point in this space represents a possible world. The overall box, labeled $\mathcal{W}[]$, contains all legal interpretations. We insist that each of these satisfies the current theory, Th .⁸

Adding a new conjecture constrains the set of allowable possible worlds. (See Section 8.2.) Here, each member of the contained region labeled $\mathcal{W}[K\#1(FlowRate)]$ represents a world in which Kirchoff's First Law holds for $FlowRate$, as well as the other facts included in the initial Th . (As shown in Figure 5-1's Legend, $K\#1$ is a shorthand for Kirchoff1, $K\#2$, for Kirchoff2. and $RKK(FlowRate, \dots)$, for $RKK(FlowRate, PressureDrop, PipeCharacter, Pipes)$.) Semantically, finding the interpretation I in that box, i.e., $I \in \mathcal{W}[K\#1(FlowRate)]$, means that the object, FlowRate, is included in "bucket" of objects labeled Kirchoff1 ^{I} . For comparison, if we select an interpretation outside that box, i.e., $J \in \mathcal{W}[] - \mathcal{W}[K\#1(FlowRate)]$, we do not know whether FlowRate is in the bucket Kirchoff1 ^{J} .⁹

There is a further embedded $\mathcal{W}[RKK(FlowRate, \dots)]$ region. Each member here represents a world in which this resistor analogue, RKK , holds for $FlowRate$ and some other arguments (e.g., $PressureDrop$, $PipeCharacter$, $Pipes$). Finding interpretation J in this box (i.e., $J \in \mathcal{W}[RKK(FlowRate, \dots)]$) means that FlowRate is in the RKK₁ ^{J} bucket.

Each of these regions contains a collection of possible worlds. We imagine the *real world* is one of these points, somewhere in this overall space (i.e., within $\mathcal{W}[]$). Now consider the *a priori* likelihood that it is in the $\mathcal{W}[K\#1(FlowRate)]$ region — i.e., that $Kirchoff1(FlowRate)$ holds. In particular, how does this compare with the chance that the real world is in the $\mathcal{W}[RKK(FlowRate, \dots)]$ region — i.e., that $RKK(FlowRate, \dots)$ holds. Given the containment of these regions, it is clearly more likely that the real world is in the larger $\mathcal{W}[K\#1(FlowRate)]$ region than in the

⁸Technically, we also insist that each interpretation is a comparable extension of the underlying interpretation \mathcal{RW} . Hence, each point in this space represents an interpretation which is in $Allowed(\mathcal{RW}, Th)$. In general, the tag $\mathcal{W}[\sigma]$ refers to all the worlds consistent with $Th + \sigma$: i.e., it abbreviates the set $Allowed(\mathcal{RW}, Th + \sigma)$.

⁹Using Chapter 8's notation, $I \in \mathcal{W}[K\#1(FlowRate)]$ means that FlowRate \in Kirchoff1 ^{I} , and $J \in \mathcal{W}[] - \mathcal{W}[K\#1(FlowRate)]$, that FlowRate \in Kirchoff1 ^{J} or FlowRate \in Kirchoff1 ^{J} . Notice we can write FlowRate rather than FlowRate as we assume that all constants are fixed. See page 173 in Section 8.2.

smaller $\mathcal{W}[\mathbf{RKK}(\text{FlowRate}, \dots)]$ region.

Why is this relevant? We can use this measure to compare how constraining a new conjecture is. This, in turn, suggests a way of comparing analogical inferences.

Stating the problem: Given what we know (*viz.*, a collection of known facts and semantic assignments), we can ask how likely it is that the independent conjecture $\text{Kirchoff1}(\text{FlowRate})$ is also true. In particular, how does this likelihood compare with the chance that $\mathbf{RKK}(\text{FlowRate}, \dots)$ holds? We answer this question by considering the *Semantic Likelihood* of each conjecture. This is defined in terms of the size of the derived interpretation sets, $\mathcal{W}[\mathbf{K\#1}(\text{FlowRate})]$ and $\mathcal{W}[\mathbf{RKK}(\text{FlowRate}, \dots)]$. Which of these conjectures imposes fewer constraints on the world? Clearly the one which removes the fewest possible worlds; *i.e.*, the one which leaves the largest set of possible interpretations. This means that I_{Least} prefers the conjecture which reduces the total space of interpretations the least, since this conjecture imposes fewer additional constraints on the world.

This allows us to express the I_{Least} intuition semantically, leading to a way of ranking of analogies based on semantic likelihood. Using the same framework shown Equation 5.8, we prefer $\varphi_1(\mathbf{A})$ over $\varphi_2(\mathbf{A})$ if $\varphi_1(\mathbf{A})$ has a higher semantic likelihood than $\varphi_2(\mathbf{A})$: *i.e.*,

Definition 15 $\text{LessConstraints}_{\text{Sem}}''(\varphi_1(\mathbf{A}), \varphi_2(\mathbf{A})) \quad \text{iff} \quad \mathcal{W}[\varphi_1(\mathbf{A})] \supset \mathcal{W}[\varphi_2(\mathbf{A})]$.

The rest of this subsection elaborates this description and presents some relevant observations.

So far, we have handled the simplest of cases, dealing only with set containment. Notice that containment corresponds to logical implication: *i.e.*, $\mathbf{RKK}|_1(\text{FlowRate}) \Rightarrow \text{Kirchoff1}(\text{FlowRate})$ means that $\mathcal{W}[\mathbf{RKK}(\text{FlowRate}, \dots)] \subseteq \mathcal{W}[\mathbf{K\#1}(\text{FlowRate})]$.

We consider two complications. First, we may have to compare sets of possible worlds which are not as closely related. Secondly, this measurement should be based on what is currently known.

As a more complex example, imagine all we knew about FlowRate is that it is a function. It might then participate in either the \mathbf{RKK} or the \mathbf{Group} abstraction.

Figure 5-2 shows this pictorially.

The $LessConstraints''_{Sem}$ measure defined in Definition 15 is worthless here, as neither set is contained in the other; in fact, the regions $\mathcal{W}[\mathbf{RKK}(\text{FlowRate}, \dots)]$ and $\mathcal{W}[\mathbf{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]$ do not even intersect.

If we make the assumption that all possible worlds are equally likely, we can extend $LessConstraints''_{Sem}$ into the more comprehensive $LessConstraints'_{Sem}$ relation, based on the cardinality of the sets.¹⁰

Definition 16 $LessConstraints'_{Sem}(\varphi_1(A), \varphi_2(A))$ iff $\|\mathcal{W}[\varphi_1(A)]\| > \|\mathcal{W}[\varphi_2(A)]\|$.

We use $\varphi_1(A) \stackrel{LC}{>} \varphi_2(A)$ to abbreviate $LessConstraints'_{Sem}(\varphi_1(A), \varphi_2(A))$. This says that the semantic likelihood of σ is proportional to the area in the $\mathcal{W}[\sigma]$ region.

As

$$\|\mathcal{W}[\mathbf{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]\| > \|\mathcal{W}[\mathbf{RKK}(\text{FlowRate}, \dots)]\|,$$

$\mathbf{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)$ would be preferred here, i.e.,

$$\mathbf{Group}(\text{Js}, \text{FlowRate}, \text{J}_1) \stackrel{LC}{>} \mathbf{RKK}(\text{FlowRate}, \dots). \quad (5.11)$$

The second consideration has been implicit in the description: the semantic likelihood of a new fact should be relative to what is currently known. This means we must first restrict the space of allowed worlds, in order to consider only the ones which are consistent with the facts we now believe. This means that this likelihood measure changes as more assertions are added to the theory.

We see that $LessConstraints'_{Sem}$ finds $\mathcal{W}[\mathbf{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]$ better than $\mathcal{W}[\mathbf{RKK}(\text{FlowRate}, \dots)]$ in the current situation. Now the situation changes: imagine we are now told that FlowRate satisfies Kirchoff's First Law. This is represented in Figure 5-3. Notice the $\mathcal{W}_{Th}[\mathbf{K\#1}(\text{FlowRate})]$ region has been double-circled; this convention indicates that this fact is known to be true — here, it means that $\mathbf{Kirchoff1}(\text{FlowRate})$ now holds. The Th subscript is used to indicate the “base

¹⁰This measure is meaningful as these sets are finite. This follows from Definition 24's definition of *Allowed* and the current assumption that the universe is finite.

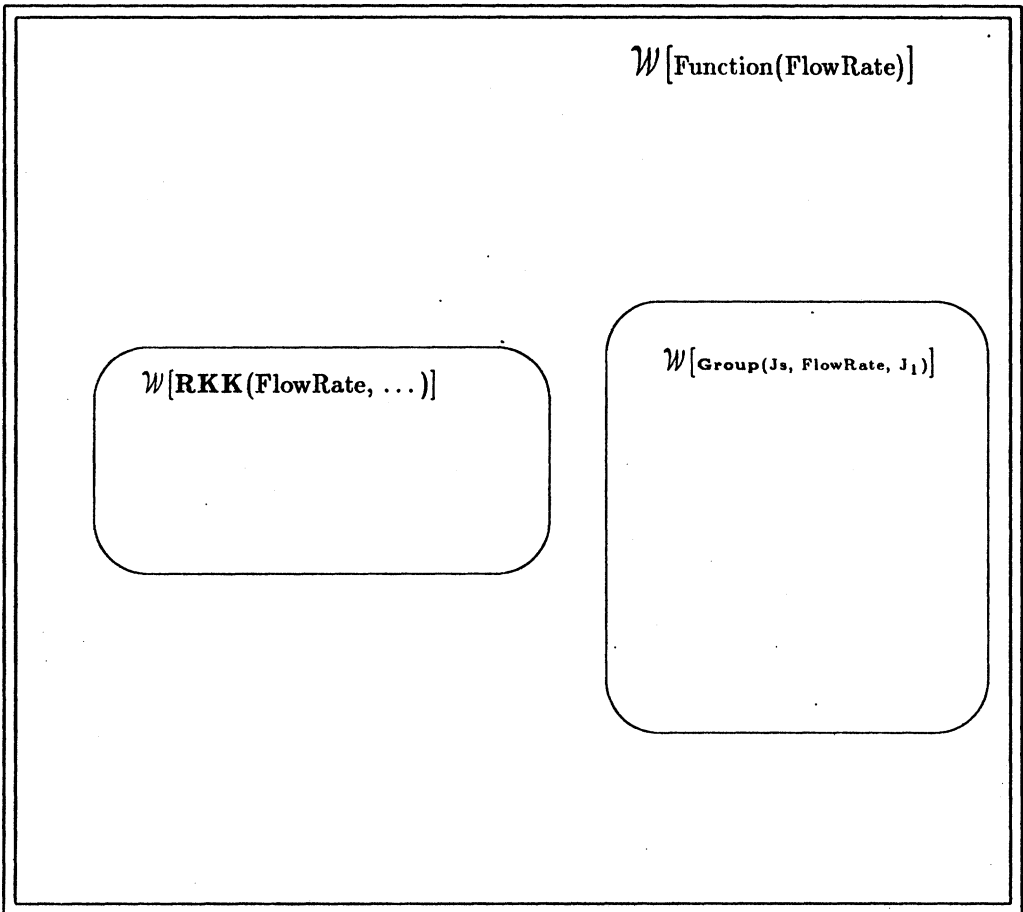


Figure 5-2: NonOverlapping Possible Worlds

theory”; these are the set of facts which we accept as true. Hence, the notation $\mathcal{W}_{Th}[\sigma]$ refers to the set of all worlds consistent with $Th + \sigma$. (As this theory was static for the purposes of the discussion above, it was left implicit.)

As we now know that the “real world” appears somewhere within this $\mathcal{W}_{Th}[\text{K\#1(FlowRate)}$ region, we need only consider the worlds within this smaller region rather than the entire space (represented by the $\mathcal{W}_{Th}[\text{Function(FlowRate)}]$ region). That is, this $\mathcal{W}_{Th}[\text{K\#1(FlowRate)}]$ region now represents the total set of allowable worlds. Notice that only some of the interpretations consistent with $\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)$ are allowed: namely, only the worlds which occur within this double-circled region. This means that the semantic likelihood that $\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)$ holds is now proportional to the part of the $\mathcal{W}_{Th}[\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]$ region which intersects the known set of allowed worlds, $\mathcal{W}_{Th}[\text{K\#1(FlowRate)}]$. Hence, the (relative) semantic likelihood of $\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)$ has decreased from $\mathcal{W}_{Th}[\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]$ to

$$\mathcal{W}_{Th'}[\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)] = \frac{\mathcal{W}_{Th}[\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]}{\mathcal{W}_{Th}[\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)] \cap \mathcal{W}_{Th}[\text{K\#1(FlowRate)}]}, \quad (5.12)$$

where Th' refers to $Th + \text{K\#1(FlowRate)}$. Notice we are using the information known (in particular, about Flowrate) to dictate the likelihood of the different conjectures.

The size of the new $\mathcal{W}_{Th'}[\text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1)]$ region is considerably reduced; in fact, it is now smaller than $\mathcal{W}_{Th'}[\text{RKK}(\text{FlowRate}, \dots)]$. This means that

$$\text{RKK}(\text{FlowRate}, \dots) \stackrel{\text{LC}}{>_{|Th|}} \text{Group}(\text{Js}, \text{FlowRate}, \text{J}_1), \quad (5.13)$$

using $\varphi_1(\text{A}) \stackrel{\text{LC}}{>_{|Th|}} \varphi_2(\text{A})$ to abbreviate $\text{LessConstraints}_{Sem}(\varphi_1(\text{A}), \varphi_2(\text{A}), Th)$, where

Definition 17 $\text{LessConstraints}_{Sem}(\varphi_1(\text{A}), \varphi_2(\text{A}), Th)$ iff $\|\mathcal{W}_{Th}[\varphi_1(\text{A})]\| > \|\mathcal{W}_{Th}[\varphi_2(\text{A})]\|$

Recall from Equation 5.11 that this had not been true earlier.

We conclude with some final comments. First, this semantic likelihood measure has many of the desired properties. For example, logically equivalent statements

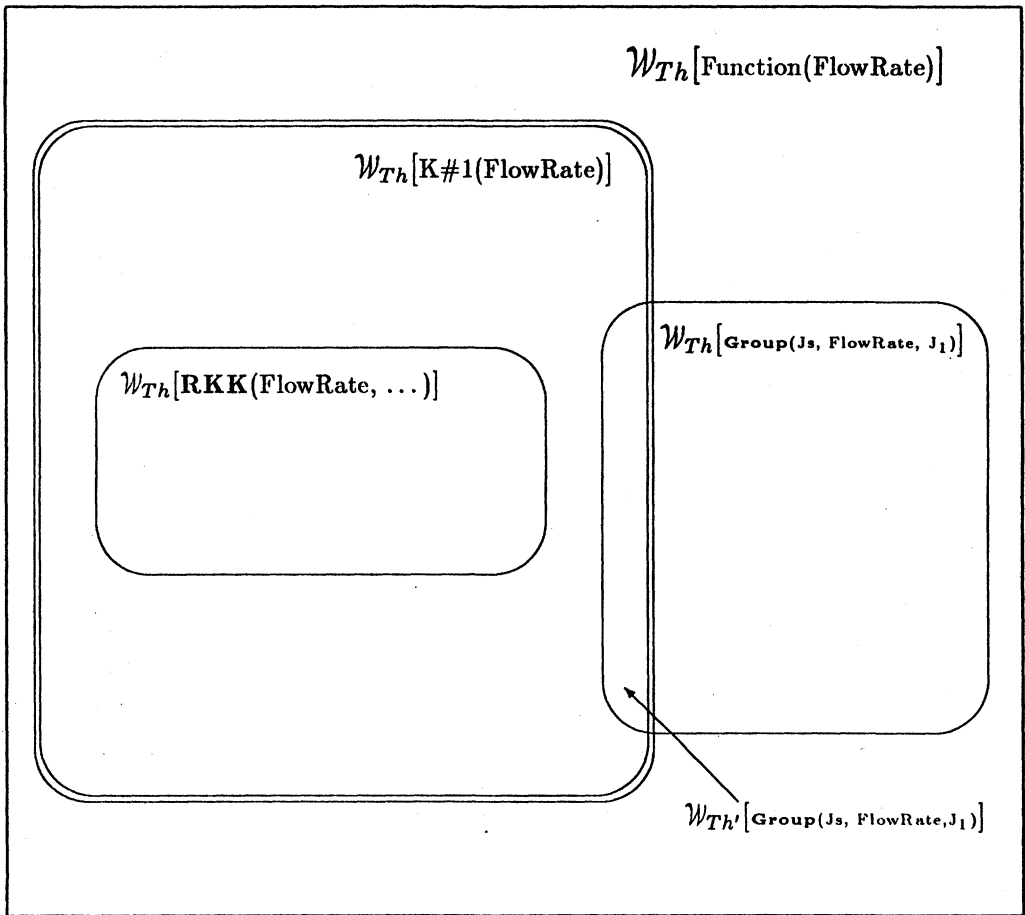


Figure 5-3: Constrained Possible Worlds

lead to exactly the same set of interpretations; e.g.,

$$\mathcal{W}_{Th}[\mathbf{KK}(\text{FlowRate}, \text{PressureDrop})] = \mathcal{W}_{Th}[\mathbf{K\#1}(\text{FlowRate}) \ \& \ \mathbf{K\#2}(\text{PressureDrop})].$$

This means that “old” facts do not impose any constraint on the known worlds: if the “new” conjecture σ is already implied by the theory, Th (i.e., $Th \models \sigma$), then $\mathcal{W}_{Th}[\sigma] = \mathcal{W}_{Th}[\]$. We can also consider what happens if we attempt to add some *inconsistent* assertion to the theory. As $Th + \sigma$ is inconsistent, this expanded theory can have no possible models; i.e., $\mathcal{W}_{Th}[\sigma] = \{\}$. This is as constraining as possible!

This calculation, in terms of the number of possible interpretations, is not computable in general. We can, however, still catch some interesting subcases. These lead to a body of related heuristics (described in the next three subsections) which use this semantic likelihood measure both to guide this search and to restrict the space.

First, though, notice that the least constraint criterion leads to good answers only when there is some bias to prevent trivial instances. Consider, for example, the $\mathbf{B} \neq \mathbf{Fred}$ relationship induced by the analogy formula $\varphi(x) \equiv x \neq \mathbf{Fred}$. As this imposes very few constraints on the world, it is probably near optimal, based on the $\mathbf{LC} >_{[Th]}$ partial order. However, it is hard to imagine situations where this analogy formula would be very useful or meaningful. This is why we consider this I_{Least} criterion only over the space of possible abstraction instances. (I.e., the H_{Abst} rule takes precedence. In fact, Section 5.4 explains why each of the abstraction-based rules, discussed in the previous section, take precedence over this I_{Least} maxim.)

5.3.4 H_{MGA} : Most General Abstraction

The “most general abstraction” heuristic, H_{MGA} , is an ordering one — it does not reduce the size of the space of possible analogies but does order it by determining which abstraction should be considered first.

It is used to compare different abstractions, e.g., \mathbf{RKK} versus \mathbf{KK} . Basically, this rule catches the simple case of logical implication:

Heuristic 4 H_{MGA} : If $S_1(\dots x, \dots) \Rightarrow S_2(\dots x, \dots)$, then prefer S_2 over S_1 .

The effect of this rule is to prefer **KK** to **RKK** to **RCKK**, and **Monoid** to **Group** to **Field**.

Notice that this rule is always consistent with I_{Least} : i.e., it always favors the analogy which imposes fewer constraints. This is because, for example, **KK**(t, c) is never more constraining than **RKK**(t, c, r, l), independent of the facts in the initial theory, Th .

In most cases, this is obvious. The one possibly problematic case occurs when some proposition in $ST_{Th}(S_1(c_1, \dots B, \dots c_n)) - ST_{Th}(S_2(d_1, \dots B, \dots d_m))$ is known initially. For example, perhaps we know initially that **Ohms**(**FlowRate**, **PressureDrop**, **PipeCharacter**, **Pipes**) held. Notice that this proposition is in that difference between **RKK**'s and **KK**'s support,

$$\begin{aligned} \text{Ohms}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \in \\ ST_{Th}(\text{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes})) & \quad (5.14) \\ - ST_{Th}(\text{KK}(\text{FlowRate}, \text{PressureDrop})). \end{aligned}$$

Even here, when $\text{Ohm}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \in Th$, we have

$$\begin{aligned} \mathcal{W}_{Th}[\text{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes})] \\ \subseteq \mathcal{W}_{Th}[\text{KK}(\text{FlowRate}, \text{PressureDrop})]. \end{aligned} \quad (5.15)$$

Hence, the least constraint maxim holds, and so H_{MGA} is justified in preferring **KK** over **RKK**.

We can be even more extreme: Suppose that every member of this difference is included in the initial theory: i.e., that

$$ST_{Th}(S_1(c_1, \dots B, \dots c_n)) - ST_{Th}(S_2(d, \dots B, \dots d_m)) \subseteq Th. \quad (5.16)$$

Under this circumstance, these two abstraction instances are equally likely:

$$\mathcal{W}_{Th}[S_1(c_1, \dots B, \dots c_n)] = \mathcal{W}_{Th}[S_2(d, \dots B, \dots d_m)]. \quad (5.17)$$

As a final comment, realize that, even though H_{MGA} always follows I_{Least} , it is still just an heuristic. There are reasonable but contrary arguments which suggest

that the more specific S_1 abstraction should be preferred to the less specific S_2 . Section 5.4 returns to this point.

5.3.5 H_{FC} : Fewest Conjectures

The “fewest conjectures” heuristic, H_{FC} , is also an ordering rule. Here, we are considering a single common analogy formula, and are deciding how to instantiate it in the target domain. When comparing different instantiations, is there any way to determine which should be considered first? In particular, how can we determine which possible instantiation is least constraining?

For example, suppose we are comparing the abstraction instances

$$\begin{aligned} \mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \\ \mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{Cost}, \text{Pipes}) \end{aligned} \quad (5.18)$$

and imagine that we knew

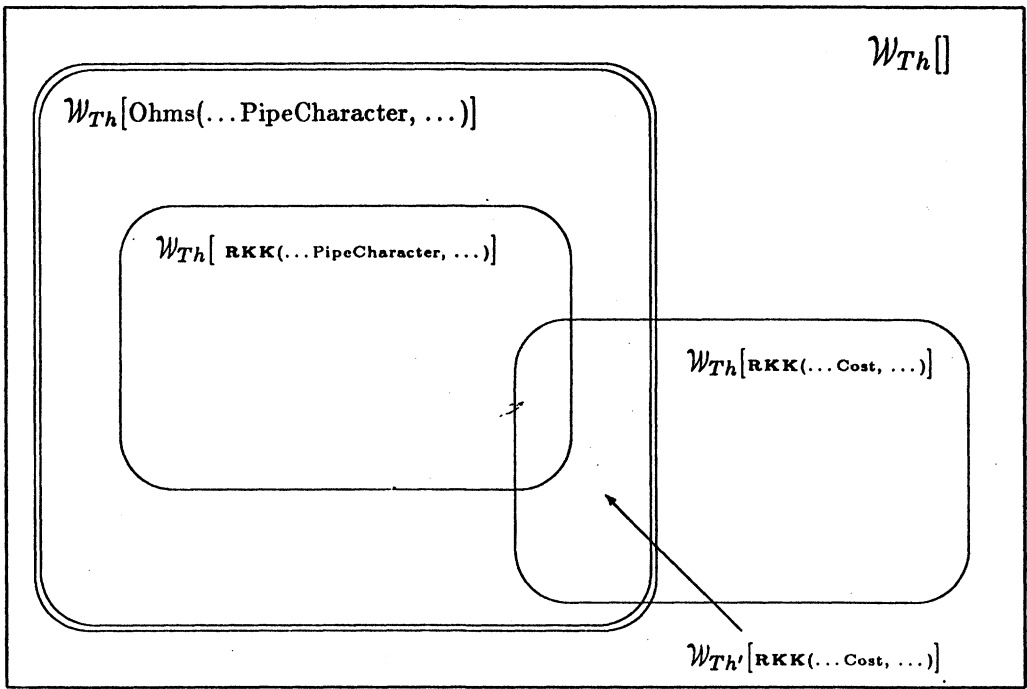
$$\begin{aligned} Th \models \text{Ohms}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \\ Th \not\models \text{Ohms}(\text{FlowRate}, \text{PressureDrop}, \text{Cost}, \text{Pipes}). \end{aligned} \quad (5.19)$$

The fact that Ohms law holds for PipeCharacter suggests that the corresponding **RKK** instantiation (involving PipeCharacter) holds in more of the possible worlds than the other one, involving Cost. Figure 5-4 shows this pictorially.

Why should this be true? Assume the two sets $\mathcal{W}_{Th}[\mathbf{RKK}(\dots\text{PipeCharacter}, \dots)]$ and $\mathcal{W}_{Th}[\mathbf{RKK}(\dots\text{Cost}, \dots)]$ are initially the same size, relative to a starting theory devoid of Ohm(...) facts. Now we learn that $\text{Ohms}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes})$ holds: this means that we need only consider the interpretations within the double-circled $\mathcal{W}_{Th}[\text{Ohms}(\dots\text{PipeCharacter}, \dots)]$ region. How does this affect the possible worlds associated with $\mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes})$ and $\mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{Cost}, \text{Pipes})$? That is, what are the relative sizes of

$$\begin{aligned} \mathcal{W}_{Th'}[\mathbf{RKK}(\dots\text{PipeCharacter}, \dots)] \quad \text{and} \\ \mathcal{W}_{Th'}[\mathbf{RKK}(\dots\text{Cost}, \dots)], \end{aligned}$$

5.3. LEAST CONSTRAINING ANALOGIES



Legend:
 $Ohms(\dots PipeCharacter, \dots) \leftrightarrow Ohms(FlowRate, PressureDrop, PipeCharacter, Pipes)$
 $RKK(\dots PipeCharacter, \dots) \leftrightarrow RKK(FlowRate, PressureDrop, PipeCharacter, Pipes)$
 $RKK(\dots Cost, \dots) \leftrightarrow RKK(FlowRate, PressureDrop, Cost, Pipes)$
 $Th' \leftrightarrow Th + Ohms(\dots PipeCharacter, \dots)$

Figure 5-4: Possible Worlds: H_{FC} Demonstration

using $Th' = Th + Ohms(\dots PipeCharacter, \dots)$?

By definition,

$$\mathcal{W}_{Th'}[\sigma] = \mathcal{W}_{Th}[\sigma] \cap \mathcal{W}_{Th}[Ohms(\dots PipeCharacter, \dots)] \quad (5.20)$$

holds for all sentences σ . As

$$\mathbf{RKK}(\dots PipeCharacter, \dots) \Rightarrow Ohms(\dots PipeCharacter, \dots)$$

we know that

$$\mathcal{W}_{Th'}[\mathbf{RKK}(\dots PipeCharacter, \dots)] = \mathcal{W}_{Th}[\mathbf{RKK}(\dots PipeCharacter, \dots)];$$

this means the set of legal worlds associated with $\mathbf{RKK}(\dots PipeCharacter, \dots)$ is unaffected.

Now consider how this new assertion affects the set of $\mathcal{W}_{Th}[\mathbf{RKK}(\dots Cost, \dots)]$ interpretations. Figure 5-4 shows that this new assertion does cut into this set: i.e., $\mathcal{W}_{Th'}[\mathbf{RKK}(\dots Cost, \dots)]$ is smaller than the original $\mathcal{W}_{Th}[\mathbf{RKK}(\dots Cost, \dots)]$.

Hence, learning that $Ohms(FlowRate, PressureDrop, PipeCharacter, Pipes)$ holds makes $\mathbf{RKK}(FlowRate, PressureDrop, PipeCharacter, Pipes)$ the clear favorite, since the resulting $\mathcal{W}_{Th'}[\mathbf{RKK}(\dots Cost, \dots)]$ is much smaller than $\mathcal{W}_{Th'}[\mathbf{RKK}(\dots PipeCharacter, \dots)]$. This means that

$$\mathbf{RKK}(FlowRate, PressureDrop, PipeCharacter, Pipes) \stackrel{LC}{>}_{[Th']} \mathbf{RKK}(FlowRate, PressureDrop, Cost, Pipes). \quad (5.21)$$

Can we state this syntactically? First, observe that there are only three conjectures which must be postulated in this first case (the one dealing with $PipeCharacter$), rather than the four in the second case, for $Cost$.

Perhaps we could just count these clauses? While Subsection 5.3.1 has demonstrated that the simple “count the conjectures” idea is not meaningful in general, this is a special case for two reasons. First, one set of conjectures is a subset of the other. Secondly, each conjecture here is a leaf clause (see Definition 19 on page 121).

This leads to the fewest conjectures heuristic, H_{FC} :

Heuristic 5 H_{FC} : If $\Delta_{LR}(Th, S(c_1, \dots, A, \dots, c_n)) = \Delta_1$
and $\Delta_{LR}(Th, S(d_1, \dots, A, \dots, d_n)) = \Delta_2$
and $\Delta_1 \subset \Delta_2$
Then Prefer $[c_1, \dots, c_n]$ over $[d_1, \dots, d_n]$.

(This uses the Δ_{LR} function defined in Definition 21 in Subsection 6.2.3.)

We repeat the above example to illustrate this rule. Consider the minimal Δ_i for which

$$\begin{aligned} Th + \Delta_{PC} &\models \mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \\ Th + \Delta_{Cost} &\models \mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{Cost}, \text{Pipes}) \end{aligned} \quad (5.22)$$

using the theory described above. Here,

$$\begin{aligned} \Delta_{PC} &= \left\{ \begin{array}{l} \text{Kirchoff1(FlowRate)} \\ \text{Kirchoff2(PressureDrop)} \\ \text{ConservedThru(FlowRate, Pipes)} \end{array} \right\} \\ \Delta_{Cost} &= \left\{ \begin{array}{l} \text{Kirchoff1(FlowRate)} \\ \text{Kirchoff2(PressureDrop)} \\ \text{ConservedThru(FlowRate, Pipes)} \\ \text{Ohms(FlowRate, PressureDrop, Cost, Pipes)} \end{array} \right\} \end{aligned} \quad (5.23)$$

As $\Delta_{PC} \subset \Delta_{Cost}$, H_{FC} prefers the binding list $[\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}]$ over $[\text{FlowRate}, \text{PressureDrop}, \text{Cost}, \text{Pipes}]$ when instantiating the abstraction \mathbf{RKK} in the target domain.

In this simple case, when one set of conjectures is a subset of the other (i.e., $\Delta_1 \subseteq \Delta_2$), there is no need to worry about leaf clauses, or for that matter, about instantiating the same abstraction. We can still guarantee the conclusion derived from Δ_1 (call it ρ_1) is less constraining than the one derived from Δ_2 (ρ_2), since $\Delta_1 \subseteq \Delta_2$ always guarantees that $\|\mathcal{W}_{Th}[\rho_1]\| \geq \|\mathcal{W}_{Th}[\rho_2]\|$.

While $NLAG$ could perform this test in general, evaluating for a possible subset

relationship between every pair of abstraction instances even for different abstractions, this is too expensive to be worthwhile. (This is not too great a loss, because the component parts of most pairs of abstractions are usually disjoint. In cases where they overlap, it is usually the case that one abstraction logically implies the other; and the H_{MGA} already catches this case.)

$NLAG$ actually uses a slight generalization of this simple rule. The rest of this subsection describes this more general H'_{FC} rule, in terms of two extensions. It is important to realize that these two rules (H_{FC} and H'_{FC}) usually produce identical advice. The only case when they differ is when H_{FC} considers the two possible analogies to be incomparable: i.e., when it has no opinion. Hence, H'_{FC} is a proper extension of H_{FC} , one which is allowed to make different decisions in situations when H_{FC} doesn't care.

The first extension allows us to compare different leaf clauses which differ by a constant to constant mapping. For example, consider the binding lists

$$\begin{array}{l} [\text{FlowRate, PressureDrop, PipeCharacter, Orifices}] \\ [\text{FlowRate, PressureDrop, PipeCharacter, Pipes}] \end{array}$$

in this same situation, i.e., using the theory Th' which includes $\text{Ohms}(\text{FlowRate, PressureDrop, PipeCharacter, Pipes})$. This leads to the same Δ_{PC} . However, the new Δ_{Orif} is

$$\Delta_{Orif} = \left\{ \begin{array}{l} \text{Kirchoff1}(\text{FlowRate}) \\ \text{Kirchoff2}(\text{PressureDrop}) \\ \text{ConservedThru}(\text{FlowRate, Orifices}) \\ \text{Ohms}(\text{FlowRate, PressureDrop, PipeCharacter, Orifices}) \end{array} \right\} \quad (5.24)$$

The important observation is that $\Delta_{Orif} \not\subseteq \Delta_{PC}$, as $\text{ConservedThru}(\text{FlowRate, Orifices}) \notin \Delta_{PC}$. However, this proposition is identical to $\text{ConservedThru}(\text{FlowRate, Pipes})$, modulo the substitution of the constant Pipes for Orifices ; and this related proposition, $\text{ConservedThru}(\text{FlowRate, Orifices})$, is in Δ_{PC} .

As H'_{FC} is willing to accept the leaf clauses which differ only by a "homomorphism" transformation (one which changes constants into constants), it can compare

the clauses associated with the bindings [FlowRate, PressureDrop, PipeCharacter, Pipes] and [FlowRate, PressureDrop, PipeCharacter, Orifices], and so can prefer the first.

While this extension to H_{FC} is not guaranteed to follow I_{Least} 's edict, it does seem fairly plausible. It follows from the assumption that any pair of unprovable leaf clauses which differ by a substitution of constants are approximately as likely. That is,

$$\|\mathcal{W}_{Th}[\phi(c)]\| \approx \|\mathcal{W}_{Th}[\phi(d)]\| \quad (5.25)$$

whenever $\phi(x)$ is a leaf clause and $Th \not\models \phi(c)$ and $Th \not\models \phi(d)$.

H_{FC} 's other extension is not as well justified. In fact, this rule actually only considers the total number of leaf conjectures, preferring the analogy which requires the fewest. Unfortunately, there is no reason to assume that this rule follows the I_{Least} principle. That would require two further assumptions, both unjustifiable. First, we would have to assume that

$$\|\mathcal{W}_{Th}[\rho_1]\| \approx \|\mathcal{W}_{Th}[\rho_2]\| \quad (5.26)$$

whenever ρ_1 and ρ_2 are each leaf clauses and $Th \not\models \rho_1$ and $Th \not\models \rho_2$. Second, we would have to assume that these clauses are fairly independent; i.e., that

$$\frac{\|\mathcal{W}_{Th}[\rho_1 \& \rho_2]\|}{\|\mathcal{W}_{Th}[\rho_2]\|} \approx \frac{\|\mathcal{W}_{Th}[\rho_1]\|}{\|\mathcal{W}_{Th}[\]\|} \quad (5.27)$$

for any pair of unprovable leaf clauses. This, unfortunately, need not be true, even for leaf clauses.

5.3.6 H_{FT} : Findable Terms

The "findable terms" heuristic, H_{FT} , is a pruning rule which reduces the size of the search space. It is closely related to the prior "fewest conjectures" rule, H_{FC} .

Instantiating the abstraction in the target space is a "residue" task, seeking an instantiation of the abstraction which is consistent with the initial theory. (The term "residue" is discussed in Subsection 6.2.3.) In its full generality, a residue process

could bind the variables to almost arbitrary terms; the only major requirement on the $?i$ is in $\mathbf{RKK}(\text{FlowRate}, ?2, ?3, ?4)$ is that $Th_{CF} \not\models \neg\mathbf{RKK}(\text{FlowRate}, ?2, ?3, ?4)$.

The H_{FT} rule makes $NLAG$ more selective. This subsection first illustrates this rule with an example and then provides a formal description and justification.

Imagine Th_{CF} includes no facts which lexically include the term, *frob*, which is, however, included in \mathcal{L} , the language of Th_{CF} . A complete satisfaction process would have to consider $\mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \textit{frob}, \text{Pipes})$, as

$$\begin{aligned} Th &\not\models \mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \textit{frob}, \text{Pipes}) \\ Th &\not\models \neg\mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \textit{frob}, \text{Pipes}) \end{aligned}$$

H_{FT} allows us to avoid such cases. It tells $NLAG$ to consider only constants which “have some support”, that is, which can be proven with respect to some derivable clause. This means it only allows $?i$ to be bound to a_i if

$$\exists \delta_j \in MI(S). Th \models \delta_j(x_1, \dots, a_i, \dots, x_n), \quad (5.28)$$

for some δ_j which is derivable from S and some set of other terms $\{x_k\}_{k \neq i}$. This uses the set of material implications following from S ,

Definition 18 $MI(S) = \{\delta \mid Th \models \forall x_i S(x_1, \dots, x_n) \Rightarrow \delta(x_1, \dots, x_n)\}$

Here, this means we are considering only those δ_j such that

$$Th \models \forall x_i \mathbf{RKK}(x_1, \dots, x_n) \Rightarrow \delta_j(x_1, \dots, x_n) \quad (5.29)$$

We can find this $\{\delta_j\}$ set by using \mathbf{RKK} 's definition, and following implication links, see Figure 5-5.

Hence, when $NLAG$ is determining the bindings to consider for its third argument r , the H_{FT} heuristic may insist that only terms which are functions be considered: i.e., that $r \in \text{Functions}$ be provable. This would force $NLAG$ to consider only the terms which are known to be functions when instantiating this argument.

H_{FT} ordains that some constraining clause be used for each variable; that constraint, however, is not limited to this type information. $NLAG$ gets to pick which constraining clause δ_j to use for each $?i$. In general, $NLAG$ uses a set of meta-level

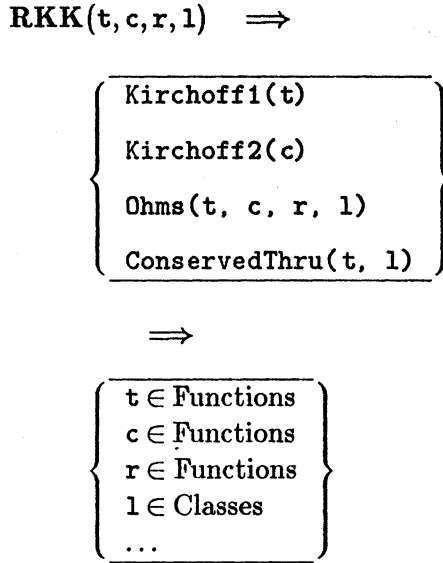


Figure 5-5: RKK's Material Implication

rules to select the generator for each argument from among the formulae present in $MI(S)$ which lexically include that argument (in the sense that the $r \in \text{Functions}$ clause *lexically includes* RKK's third argument, r). The current implementation defines the generator as the atomic formula which has the highest "coverage" value, where coverage is based on the associated relation symbol. (As different functions may be better for different data bases, this function is intentionally left underspecified. My *NLAG* implementation employs a user-modifiable second-order function to determine this coverage value.)

The H_{FT} rule is sufficient to eliminate this frob case. That is, as frob does not qualify as a function (i.e., $Th \not\models \text{frob} \in \text{Functions}$), H_{FT} tells *NLAG* not to consider it as a candidate for RKK's third argument.

Now for the formal statement:

Heuristic 6 H_{FT} : Allow target instantiation $S(\dots a_p, \dots)$ only if
 $\exists \delta_j \in MI(S), x_{k \neq p}. Th \models \delta_j(x_1, \dots a_p, \dots x_n).$

Semantically, this rule means we are not considering the extremely constraining case where we would have to add in *everything* about a concept. Consider, for example, how many previously allowed worlds are eliminated when we assert $\text{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{frob}, \text{Pipes})$. Notice this is far more than are eliminated by asserting $\text{RKK}(\text{FlowRate}, \text{PressureDrop}, x, \text{Pipes})$ for any x which is known to be a function, i.e., where $Th \models x \in \text{Functions}$. (That is, given how little we know about *frob*,

$$\begin{aligned} & \|\mathcal{W}_{Th}[\text{RKK}(\text{FlowRate}, \text{PressureDrop}, x, \text{Pipes})]\| \\ & > \|\mathcal{W}_{Th}[\text{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{frob}, \text{Pipes})]\| \end{aligned}$$

for any $x \in \text{Functions}$.)

In terms of the implementation, H_{FT} means that *NLAG* can use one clause (taken from $MI(S)$) as a *generator* for each variable. (See Subsection 6.1.5.)

Some final notes:

- In theory, this rule could constrain the search tremendously. In normal practice, though, it does not eliminate many otherwise possible values. This is because few knowledge bases contain quantities as unknown as this *frob*. Most systems tag each concept with at least some type information: i.e., most concepts are at least in some *IsA*-hierarchy or declared a member of some class. As most abstractions include type restrictions on their arguments as well, the general satisfaction process would already have eliminated most such terms.
- H_{FT} means that *NLAG* does not deal with new terms. (See the comment on page 53.) While our use of abstractions does make it straightforward to generate new terms (i.e., we could define a new constant as precisely the “value” required to fill some formal parameter), we choose not to do so. Note:5-2 defends this decision.
- This rule is related to the abduction rule of the plausible inference: from $A \Rightarrow B$ and B , plausibly infer A . (This is also called “affirmation of the consequent”.)

5.4 Summary of Heuristics

So far, we have discussed several diverse heuristics which help us navigate about the space of legal analogical inferences. This section wraps up the discussion. Subsection 5.4.1 first provides a thumbnail sketch of each rule together with a quick classification. Subsection 5.4.2 then describes how the rules interact with one another.

As a final note, we will hereafter use $\|_{PT}$ to refer to the abstraction-based useful analogical inference process *which incorporates the heuristics described in this chapter*. (Technically, we should use a new symbol for this more refined process, but the reader is already, no doubt, overwhelmed with notation.)

5.4.1 Classification and Synopsis of Heuristics

The quickest summary of these rules comes from Section 3.4: *find enough new information about the target analogue to solve the target problem, and then stop*. The abstraction-based rules operationalize the *find enough information* clause, and the I_{Least} rules, the *then stop* part.

There are several other ways of describing the specific rules. The basic outline of this dissertation provides one structure, separating the overarching *use only abstraction* rule in Chapter 4 from the other abstraction-based rules of Section 5.2 and the general purpose rules of Section 5.3. Another cut divides the rules based on whether the rule prunes or orders the space. A third property concerns the use of the rule: whether it helps determine the common formula to use or describes how to instantiate such an analogy formula.

Table 5-1 classifies the heuristics. It includes a quick summary of each rule, tells where it is defined and describes how it fits into the dimensions mentioned above. These are arranged in order of appearance. Another important classification is the rule's applicability: the structuring below reflects this as well. (The next subsection shows that this order of appearance mirrors the relative prominence of these rules; the most prominent ones are shown first.)

- Abstraction-Based Heuristics

- H_{Abst} : Use Abstractions (Chapter 4, Heuristic 1 and Figure 4-5)
Summary: Permit only certain formulae (namely, abstraction formulae) to be used as analogy formulae.
 Pruning rule, pertains to common formula.
- H_{JK} : Justification Kernel (Subsection 5.2.1, Heuristic 2)
Summary: Use the target problem to select relevant abstraction.
 Pruning and ordering rule, pertains to common formula.
- H_{CC} : Common Context (Subsection 5.2.2, Heuristic 3)
Summary: Insist that all the constants used to instantiate an abstraction come from the same theory.
 Pruning rule, pertains to instantiation.

- I_{Least} -Based Heuristics

- H_{MGA} : Most General Abstraction (Subsection 5.3.4, Heuristic 4)
Summary: Prefer the most general abstraction.
 Ordering rule, pertains to common formula.
- H_{FC} : Fewest Conjectures (Subsection 5.3.5, Heuristic 5)
Summary: For a given abstraction, prefer the instantiation instance which requires the fewest conjectures.
 Ordering rule, pertains to instantiation.
- H_{FT} : Findable Terms (Subsection 5.3.6, Heuristic 6)
Summary: When instantiating an abstraction, consider just constants which have some "support".
 Pruning rule, pertains to instantiation.

Table 5-1: Synopsis of Heuristics

5.4.2 Interactions of Heuristics

How do these rules interact? In particular, what happens if a pair of rules provide conflicting advice? The previous sections mentioned a few instances of such conflicts on a case-by-case basis. There is a single, general meta-rule: *always prefer an abstraction-based rule over a I_{Least} -based rule*. Why? The abstraction-based rules are more definite; each has a specific reason to justify its use. By contrast, the I_{Least} -inspired rules are weaker, based only on a general-purpose heuristic. Hence, we only use the latter rules when nothing else pertains. (Notice this means that almost any other rule should dominate these I_{Least} rules as well.) This is consistent with the comments first presented in Section 3.4: *viz.*, that I_{Least} should be used as a further refinement of the I_{Close} maxim.

Now for the particulars. Consider first the three rules which specify which analogy formula to use: H_{Abst} insists that the formula is an abstraction, H_{JK} and H_{MGA} then determine which abstractions should be used and in what order. These latter two heuristics often conflict: H_{JK} may prefer the abstraction S_1 over S_2 , even though S_1 is more specific (i.e., contrary to H_{MGA} 's wishes). $NLAG$ follows H_{JK} 's advice in these situations. For example, in the electricity and hydraulics situation, $NLAG$ may consider the **RKK** abstraction before **KK**.

Why? The only reason H_{JK} can prefer the more specific S_1 over S_2 is if the support for the source problem included a fact which is in S_1 's support but not in S_2 's. Subsection 5.2.1 argued that this is desirable, even if it violates H_{MGA} . (In fact, the **RKK** versus **KK** example described in that subsection does just that.)

However, H_{JK} rule takes no stand when the two sets of possible worlds are of equal size; i.e., when $\|\mathcal{W}[S_1(\dots)]\| = \|\mathcal{W}[S_2(\dots)]\|$. The H_{MGA} rule is often relevant here. For example, if $\|\mathcal{W}[\mathbf{RKK}(\dots)]\| = \|\mathcal{W}[\mathbf{KK}(\dots)]\|$, H_{MGA} would recommend that **KK** be considered first. This makes sense: in this situation, we have no reason to prefer the more constrained, and therefore more expensive, **RKK** abstraction over **KK**. (The discussion at the end of Section 7.3 describes this interplay and explains the advantages of our approach in this trade-off. See also Note:6-2, which explains why $NLAG$ would consider **KK** after **RKK** has failed.)

The other three rules are used to determine the instantiation of this abstraction formula in the target domain. We saw above that H_{FT} is a special case of H_{FC} , and hence these two must be in agreement. The H_{CC} requirement is not so agreeable. In cases of conflict, it takes precedence over both of the other rules. This is because the H_{CC} rule supplies an essential part to each abstraction — a part which should be present in the abstraction's definition but has been excised (as a convenient shorthand) because we knew H_{CC} would re-supply it. That is, H_{CC} allows us to write the simpler **RKK** (of Figure 4-3), knowing that it is, in effect, interpreted as **RKK_{CC}** (of Equation 5.7).

There are two final points. First, it is easy to find situations where H_{MGA} seems incorrect, where one might legitimately prefer the more specific S_1 abstraction over the less specific S_2 . (E.g., it might make sense to prefer **Group** over **Monoid**, in some situations.) This argues against this use of I_{Least} for choosing abstractions, leaning instead towards the I_{Most} intuition. (The tension between these two insights, I_{Most} and I_{Least} , was first discussed in Section 3.4.) Subsection 9.2.3 discusses this point in detail. For now, we want only to mention that this tension does not appear when considering how to instantiate an abstraction, i.e., for the other set of rules. It is hard to imagine any reason to prefer an instantiation which requires more additional conjectures rather than less (i.e., which violate H_{FC}), or to prefer an unfindable term over one already present (i.e., which violate H_{FT}).

These anti- H_{MGA} arguments point out that there is no real reason why I_{Least} should be true. That is, there is no real reason why the semantically most likely common abstraction should be the best. The only arguments are based on empirical evidence and an intuitive notion of reasonableness, and either can have counterexamples.

The final point pursues this justification theme and deals with how one might justify any of these rules. It is straightforward to express most of these rules semantically. (Subsection 5.3.3 has already done this for the I_{Least} -based rules; Section 8.4 does this for some of the other rules.) We can furthermore provide a semantic justification for the I_{Least} rules, in terms of semantic likelihood of the conjectures. (This was done in Subsection 5.3.3.) However, we were unable to find a semantic

justification for any of the abstraction-based rules. (See Note:4-4.)

Chapter 6

Specification of the NLAG Process

The previous chapters defined and justified the behavior of the general abstraction-based (useful) analogical inference process. This chapter specifies my particular implementation of this process, the *NLAG* program. The next chapter discusses results obtained with this system, to demonstrate its effectiveness.

Section 6.1 describes the structure of the *NLAG* learning-by-analogy system, focusing on the module which seeks common abstractions, *ComAbs*. Section 6.2 describes important facilities which must be present to construct an *NLAG*-like system. Section 6.3 demonstrates that this program finds all and only the \Vdash_{PT} analogies. It also includes an explicit set of conditions which, if satisfied, insures that *NLAG* finds the “correct” analogies and does so efficiently.

The following two points may help in understanding both the *NLAG* algorithm and this particular presentation of it. The first deals with the nature of the algorithm. Section 6.3 demonstrates that *NLAG* is a *bona fide* \Vdash_{PT} algorithm. However, even with the definitions and heuristics given in the previous chapters, the \Vdash_{PT} process is not completely specified. In particular, the general \Vdash_{PT} process may be unable to decide which of a pair of legal analogies should be considered first. *NLAG* uses a variety of other rules to totally order the selection. As hinted in previous chapter, I regard these additional rules as nuances of the implementation rather than important principles. This justifies their rather informal treatment in this chapter.

The second point explicates the objective of this chapter. Its purpose is to

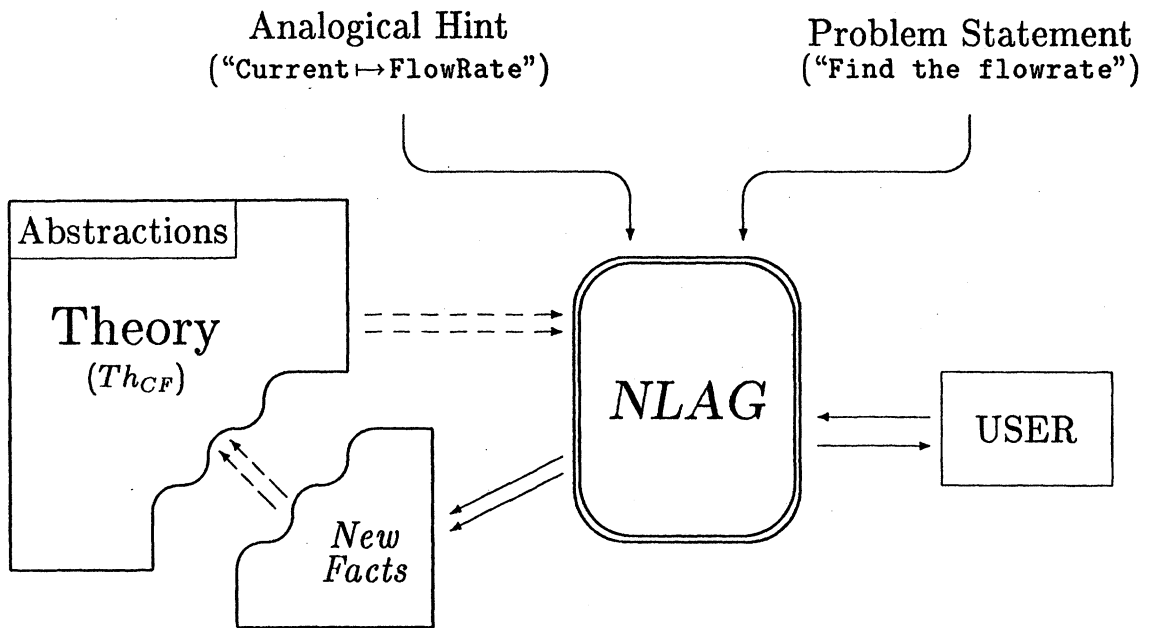


Figure 6-1: Overall Behavior

describe the *NLAG* algorithm in sufficient detail that others can construct their own versions, for their own representation systems and sets of facts. As such, this description is relatively high-level, mentioning very few specific procedures. (To avoid offending the competent reader, this chapter totally avoids showing low-level LISP-y code.)

6.1 Components of *NLAG* System

This section presents a high-level overview and behavioral description of the *NLAG* process. The subsections present various components in more depth. (The actual code used, expressed as MRS assertions, appears in SubAppendix B.1.)

Figure 6-1 shows the basic behavior of the *NLAG* system. Its input is the target problem PT (here, "Find the flowrate") and an analogical hint $AH = \{A \mapsto B\}$ which maps the target analogue into the source analogue (here, $AH_{CF} = \{\text{FlowRate} \mapsto \text{Current}\}$). *NLAG* also has access to the facts in a theory, Th . Here,

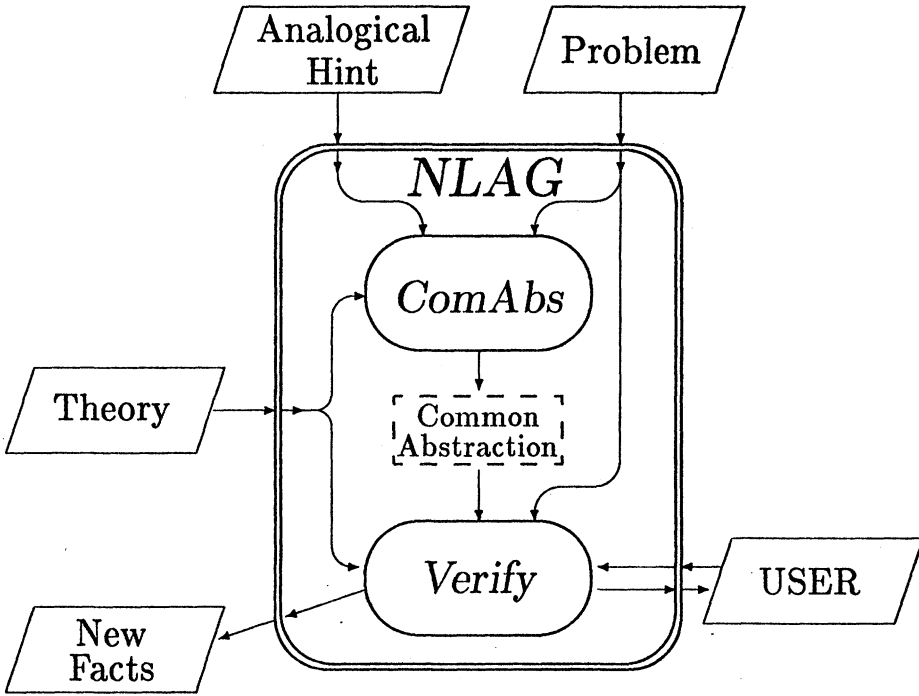


Figure 6-2: Detailed Behavior of NLAG

Th_{CF} includes both EC and FS as well as a library of abstractions, which includes **RKK**. Its output is a set of proposed new FS facts which leads to a solution of the target problem.

(The previous chapters described the result of an \Vdash_{PT} inference as a particular abstraction (e.g., **RKK**) and the target instantiation (e.g., [FlowRate, Pressure-Drop, PipeCharacter, Pipes]). To describe the *NLAG* process, it is easier to view this output as a set of new facts to be added to the learner's theory. These two descriptions are related: these new facts (the *NewFacts*) box shown in the lower left corner in Figure 6-1) are precisely the residue of the instantiated abstraction, $S(a_1, \dots, A, \dots, a_n)$ — i.e.,

$$\Delta(Th, RKK(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}), \langle \text{NewFacts} \rangle)$$

holds.)

Figure 6-2 shows *NLAG*'s two conceptual modules. The first component, *ComAbs*,¹

¹We will continue to use this *slanted fixed-width font* when writing the name of both subroutines and various internal processes. Notice that "NLAG" is already in this font.

is a generator: it seeks pertinent abstractions (e.g., **RKK**) which can be instantiated in both the source and target domains, and passes each such common abstraction to *Verify*. This second process acts as a pruner, deciding whether to add in the suggested new conjectures based on their effectiveness in solving the problem as well as interactions with the user.

The next subsection quickly describes the *Verify* module. The rest of this section discusses *ComAbs* and its children. These latter modules are the meat of the *NLAG* system: they perform, basically, the \Vdash part of \Vdash_{PT} . They are responsible for everything but confirming the effectiveness of the conjectures, and, as such, embody all of the heuristics presented in the dissertation.

6.1.1 *Verify* Module

Verify's purpose is to insure the appropriateness of the conjectures, hereafter labeled $\{\delta_j\}$. It runs the following three tests: a set $\{\delta_j\}$ must pass all three to be approved. First, *Verify* verifies that these new facts provide a solution to the target problem. This involves adding these conjectures to the initial theory, and running (*TRUEP* (*PT*)).² These conjectures pass this test if *TRUEP* returns an answer. If so, *Verify* runs the second test: are these conjectures consistent with the initial theory and each other? (While *ComAbs* does guarantee that *Consistent*(*Th* + δ_j) is true for each individual δ_j , this is not sufficient to guarantee that *Consistent*(*Th* \cup $\{\delta_j\}$) holds.) *Verify* performs this check by running (*TRUEP* (not δ_j)) for each δ_j , each time in an environment which contains both the original *Th* and the other conjectures, $\{\delta_k\}_{k \neq j}$. If every *TRUEP* call fails (i.e., returns *NIL*), we claim these conjectures to be collectively consistent. The final test is user acceptance: the user is asked whether he approves of each new conjecture.³ Unless all are judged

²*MRS* includes two basic querying subroutines, *LOOKUP* and *TRUEP*. Each takes a proposition pattern, and returns the facts which match this pattern and are deducible from the current theory. They differ in that *TRUEP* may perform various deductions when searching for a fact, while *LOOKUP* does not. A fact is *primitively stored* if a simple *LOOKUP* can find it: i.e., if it can be found without any inferences. See [Rus85] for more details.

³In general, the same conjecture may arise in different iterations. To avoid annoying the user, *Verify* only presents a proposed conjecture to the user once and records (and re-uses) his answer.

acceptable, this entire batch is rejected. If this $\{\delta_j\}$ set fails any of these three tests, *NLAG* asks *ComAbs* to produce a new set of conjectures.

There is some additional work required when an answer passes these tests, and even more (involving elaborate clean-up steps) when a proposal is rejected. Many of these are trivialized by the effective use of *MRS*'s theory mechanism.

6.1.2 *ComAbs* Module

This subsection describes the *ComAbs* process, both behaviorally and structurally. The subsequent subsections describe its internal modules in more detail.

Behaviorally, the *ComAbs* process has two subtasks. The first finds deducible abstraction instances in the source domain, $S(b_1, \dots, B, \dots, b_n)$. The second uses this abstraction S and the analogical hint AH to find a corresponding abstraction instantiation, $S(a_1, \dots, A, \dots, a_n)$ — one which can be consistent with the initial theory. For each such target abstraction instantiation, *ComAbs* also determines the residue, i.e., the additional facts which must be conjectured.

Structurally, *ComAbs* has three component parts, shown in Figure 6-3. The first two subprocesses, *Find-Kernel* and *Inst-Source*, work on the first subtask. Together, they return abstractions which have instances which are deducible in the source domain. The third component, *Inst-Target*, attempts to satisfy an instance of this abstraction in the target domain. (*Find-Kernel* produces a single answer. By contrast, each of the other two subprocesses may return many different answers. Each is implemented as a "generator": that is, each generates a single answer at a time.)

The following summary provides a quick synopsis of these modules and indicates which heuristics (from the previous chapters) govern their operations. The next subsections provide more complete descriptions of their respective behaviors.

1. **Find-Kernel:** From the problem statement PT and analogical hint AH , produce a *kernel* of possibly pertinent source facts. This uses the H_{JK} heuristic

This also means that if the user ever rejects a proposition, *Verify* never again asks him about this clause.

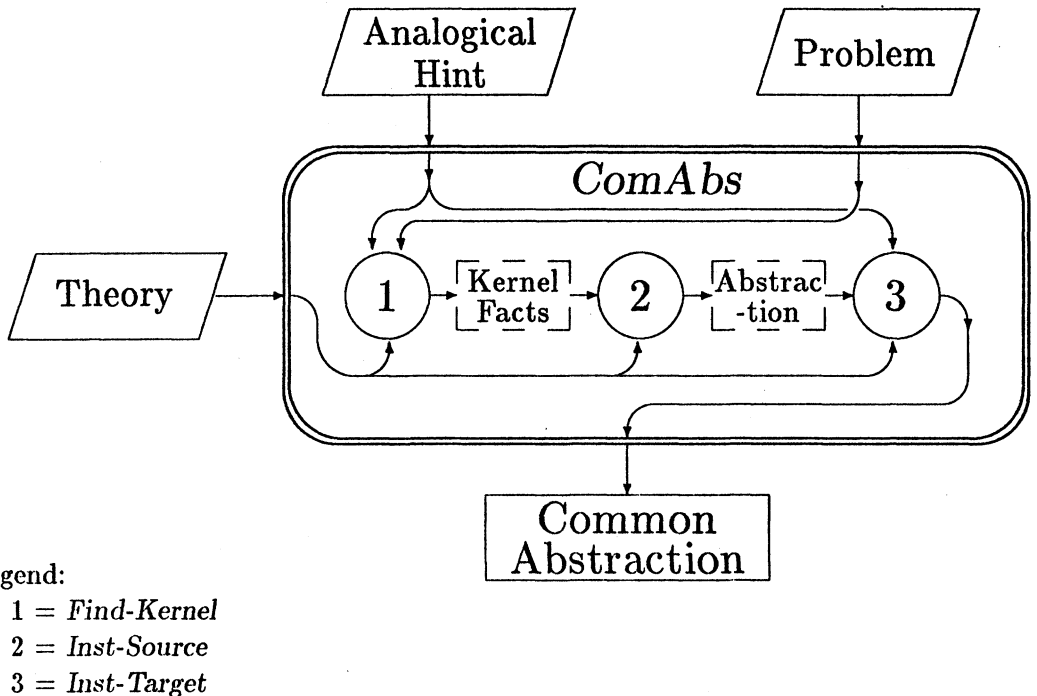


Figure 6-3: Detailed Behavior of ComAbs

(see Subsection 5.2.1).

2. **Inst-Source:** Forward-chain from those kernel facts to deduce abstraction instances $S(b_1, \dots, b_n)$ in the source domain. This uses the H_{Abst} and H_{MGA} heuristics (defined in Section 4.2 and Subsection 5.3.4).
3. **Inst-Target:** From that abstraction and the initial map AH , find a corresponding abstraction instantiation, $S(a_1, \dots, a_n)$, which is consistent with the initial theory. For each abstraction instantiation, *Inst-Target* also returns the additional facts which must be conjectured. This uses the three heuristics: H_{CC} (Subsection 5.2.2), H_{FC} (Subsection 5.3.5) and H_{FT} (Subsection 5.3.6).

6.1.3 1: Find-Kernel Module

The goal of this first step is to determine the source facts which may be relevant to the target problem. We call these the “kernel facts”. Most of this subprocess

implements the H_{JK} heuristic, defined in Subsection 5.2.1. As discussed there, this process begins by syntactically deriving a source problem, PS , by lexically substituting B for A (i.e., the source for the target concept) in PT . In this hydraulics from electricity situation, this leads to a “Find the current” query. *Find-Kernel* next searches for the methods which would help solve this new query. Here, this includes the specific rules which describe how to compute the current in a parallel circuit. *Find-Kernel* then retrieves the justifications for these rules; here, the electrical Kirchoff’s and Ohms Laws. This set is returned as the kernel facts.

As mentioned above, this rule does not always apply: Recall that H_{JK} ’s lexical substitution would not work if we had the same “Find the flowrate” problem to solve, but were given the different hint, e.g., $AH = \{\text{VoltageDrop} \mapsto \text{PressureDrop}\}$. In this situation, *Find-Kernel* defines the kernel to be all known facts which lexically contain the source concept, here *Current*. That is, *Find-Kernel* returns $\mathcal{P}_A(Th)$, using the “lexical A-projection” defined in Definition 35 in Note:5-1. (That note also shows that this set of facts is guaranteed to be sufficient.) Subsequent implementations of this system may use other heuristics here — probably domain dependent as well as general ones. Point FW4 in Section 10.2 discusses this eventuality.

6.1.4 2: *Inst-Source Module*

This second step forward-chains from the kernel facts, searching for deducible abstraction instances in the source domain; i.e., ground statements of the form $S(b_1, \dots, b_n)$, where S is an abstraction and B is the source analogue given in the analogical hint. This can be considered a complex indexing task, using the kernel to identify the proper abstractions. (See Section 4.5.)

In general, this forward-chaining process takes a theory Th and a kernel, Φ (which is a subset of the starting theory, i.e., $\Phi \subseteq Th$), and seeks all facts, σ , which

- are (ground) abstraction formulae, i.e.,
 $AbstForm(\sigma)$ (see Definition 11),
- lexically contain the source analogue, i.e.,

LexInclusion(A, σ) (see definition on page 24), and

- require some member of the kernel, Φ , as a justification, i.e.,

$$ST_{Th}(\sigma) \cap \Phi \neq \{\}.$$

(As we only deal with derivable sentences, the support $ST_{Th}(\sigma)$ is guaranteed to be defined and non-empty.)

Inst-Source operationalizes this by forward-chaining from the kernel facts, stopping when it reaches an atomic expression whose relation is an abstraction and which includes the source analogue as an argument. (This uses the *FC-Find* subroutine described in Subsection 6.2.2.)

As this search is performed in a breadth-first fashion, *Find-Kernel* first finds the abstraction whose instantiation required the fewest number of deductions. Note:6-1 formalizes this fewest deductions measure. (Note:3-1 mentions that this is but a minor embellishment.)

As a final comment, this *Inst-Source* routine does consider more general abstractions, even after a more specific one has failed. (This is described in page A-88 in SubAppendix B.3.1.) Note:6-2 discusses why this is appropriate.

6.1.5 3: *Inst-Target* Module

The third step tries to *satisfy* the sentence $S(?_1, \dots, A, \dots, ?_n)$, where S is the abstraction found in the previous step, A is the target analogue and each $?_i$ is an existential variable. It returns both a binding list, $\{?_k \mapsto a_k\}$, and the associated residue, $\{\delta_j\}$. The residue $\{\delta_j\}$ consists of the conjectures which must be postulated to achieve this instantiation of the abstraction; i.e., $\Delta(Th, S(a_1, \dots, A, \dots, a_n), \{\delta_j\})$. For example, satisfying $RKK(\text{FlowRate}, ?_2, ?_3, ?_4)$ may lead to the binding list $[\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}]$ and may require postulating that $\text{Kirchoff1}(\text{FlowRate})$ and $\text{Kirchoff2}(\text{PressureDrop})$ each hold. This attempt can fail if a proposed conjecture is inconsistent with the facts currently known.

Section 4.3 already discussed some of the problems associated with finding a meaningful residue. This is why *Inst-Target* considers only *leaf residues*. As an

example, the only leaf residues of the **RKK** relation are the different subsets of the $\{\delta_j\}$ shown in Figure 4-7 on page 61. Subsection 6.2.3 provides a more formal definition of this notion of leaf residues and describes *CTruep*, a process which finds only these conjectures. In fact, the *Inst-Target* component reduces to a particular invocation of *CTruep*.

6.2 Underlying System

This section describes the underpinings required to develop a system like the *NLAG* program. My particular *NLAG* procedure is built on top of the MRS system, a expert system building tool described in [GGGS80,Rus85].⁴ Subsection 6.2.1 describes why this system was used: *i.e.*, which of its features made my analogy task easy to implement. Powerful as that system is, *NLAG* required various modifications, mostly additions to enhance its capabilities. The following subsections describe several of those enhancements: Subsection 6.2.2 describes the *FC-Find* process used by the *Inst-Source* module and Subsection 6.2.3, the *CTruep* process used by the *Inst-Target* module. Subsection 6.2.4 closes this section by summarizing all of the significant changes this project has prompted. This includes a host of more minor embellishments, many of which have been incorporated into the growing MRS system. (As such, these ideas and code represent an additional contribution of this project.)

6.2.1 Why Use MRS?

This subsection discusses the MRS system and lists its salient features.

In a nutshell, MRS is like Prolog [CM81,Kow79], augmented with meta-level control. Like Prolog, information is input in a predicate calculus format and retrieved by issuing general query commands. The major difference is that the MRS user can also issue meta-level statement which alter the underlying inference process, representations used, *etc.* This additional facility helped me produce a running

⁴The entire system lives on Diablo, a VAX 11/780 running the Unix^(tm) operating system, Version 4.2.

system quickly, and then alter it effectively.

MRS provides enough useful features that I would encourage anyone who wants to implement an *NLAG*-like system to use such a system. In particular, the support package should include the following features (all present in MRS):

- Standard unification, [CL73,Nil80], using fully indexed propositions.

As MRS's basic *LOOKUP* facility does general retrieval automatically, I did not have to worry about just how to retrieve which facts.

- A back-tracking, backward-chaining inference engine, capable of deriving valid facts from consistent axioms:

(MRS includes this inference engine, named "BC", among others.) This meant that I did not have to write specific programs to handle the bulk of *NLAG*'s processings; straightforward logical conclusions would be inferred automatically from the inventory of facts. In fact, *NLAG*'s actual "code" is actually a set of production rules, and its overarching process is simple backward-chaining using these rules. (This code appears in SubAppendix B.1.)

- A general procedural attachment facility.

This allowed me to write my own, more efficient routines as necessary. MRS provides the hooks with which to use such LISP code.

- A meta-level control facility.

I found many uses of this facility. As its strongest asset, it allowed me to cleanly separate domain information (e.g., the definition of a legal analogy) from control information (e.g., whether one analogy should be returned before another). To be more concrete, both *Inst-Source* and *Inst-Target* rely on this capability to control the order in which they return their answers. Hence, this control information is essential for a running production system.

It was even more useful to *NLAG* *qua* experimental vehicle: Many parts of Experiment#3 involved re-arranging the order in which answers were returned. Each of these required only changing a single meta-level instruction. See Section 7.5 and related sections in Appendix B.

This evidence argues that such a meta-level language is the appropriate way to add later domain-dependent control information, information used to further order the abstractions and their instantiations. While this information can be awkward and cumbersome to describe and use in general, it is natural in this language.

This meta-level information is used in other places as well. For example, *Find-Kernel*'s work is greatly simplified by having access to the stored justifications of earlier deductions.

(The last two points are where MRS improves on Prolog, as well as over most other expert-system building tools.)

6.2.2 *FC-Find* Module

ComAbs's second step, *Inst-Source*, requires the ability to forward-chain from a given set of facts, returning every derived fact which matches a given goal form. The *FC-Find* module was designed to perform this task. In general, this module starts from a collection of axioms and, in breadth-first manner, forward-chains from each, returning each "goal" proposition as it is encountered. (A meta-level assertion specifies these goal propositions.)

This process is modeled after MRS's standard forward-chaining system, *FC*. The differences are:

- *FC*'s input is a single proposition, and it always returns an arbitrary nonNIL value. By contrast, *FC-Find* takes a list of propositions, $\langle \rho_1, \dots, \rho_k \rangle$, and returns the goal propositions it finds (see next point).
- *FC-Find* succeeds when it finds a proposition which matches the goal statement, and returns that goal proposition. That is, whenever *FC-Find* encounters a proposition ρ which satisfies (fc-goal ρ), it returns this ρ . (By using MRS's agenda mechanism, *FC-Find* can be used as a generator, producing all of the legal answers, one at a time.)

- *FC-Find* continues to process a proposition even if that proposition is primitively stored in the theory: i.e., when (*LOOKUP* ρ) returns a nonNIL answer. (By contrast, *FC* stops on finding such a proposition.)
- While considering the other clauses of the antecedent of a rule, *FC-Find* does a full *TRUEP*, while *FC* only does a more limited *LOOKUP*. To illustrate this: Suppose *FC* is forward chaining from the proposition ρ , and encounters a rule, (if (and ρ σ) τ). At this point, *FC* performs a simple (*LOOKUP* σ), and fails if this call returns NIL. By contrast, if *FC-Find* was in this situation, it would instead check the more encompassing (*TRUEP* σ).

6.2.3 CTruep Module

ComAbs's third step, *Inst-Target*, is a particular type of *residue* process, one which returns a residue of a given proposition, with respect to a theory. ([Fin85] coins this term "residue"; see also [Doy79,dK].) This is implemented as a separate subroutine, named *CTruep*.⁵ (This allows it to be used independently — see the experiments reported in Subsection 7.5.)

This subsection describes the *CTruep* process. The first part defines its behavior, characterizing the conjectures it proposes. The second part describes the particular algorithm used to achieve this behavior.

CTruep Behavior

Like all residue procedures, *CTruep* is similar to standard deduction procedures. (In fact, it is built as a slight deviation of MRS's *BC* backward-chaining routine.) All start with some variabilized proposition, here of the form $S(?_1, \dots, A, \dots, ?_n)$, and seek some bindings for the $?_i$ variables, $\{?_k \mapsto a_k\}_k$. The difference is their respective requirements on this $[a_1 \dots a_n]$ instantiation: *CTruep* requires only that the instantiated form be *consistent* with the starting theory, rather than be *deducible* from it. This means it imposes the weaker requirement of *Consistent*($Th + S(a_1, \dots, A, \dots, a_n)$), not $Th \models S(a_1, \dots, A, \dots, a_n)$.

⁵for Conjecturing Truep.

A standard query-answering process returns only this binding list, $\{?_k \mapsto a_k\}_k$. *CTruep*'s output has another part as well: namely, a set of conjectures associated with each binding list, $\{\delta_j\}$. These are the conjectures which, when added to the theory, enable proving $S(a_1, \dots, A, \dots, a_n)$. Using Definition 13, this means that $\Delta(Th, S(a_1, \dots, A, \dots, a_n), \{\delta_j\})$ must hold.

For this to be meaningful, we have to constrain which propositions can qualify as reasonable conjectures. Otherwise (repeating the complaint aired on page 61), we could simply assign δ_1 to be the sentence we were trying to satisfy, $S(a_1, \dots, A, \dots, a_n)$. (We use the term "satisfy" here in the sense of "constraint-satisfaction", not in the model theoretic sense. This follows from the view that the theory, *Th*, is a collection of constraints, and we are seeking some instantiation of the "to be satisfied proposition" which is consistent with these constraints.)

CTruep is more restrictive. We now characterize the types of conjectures it allows.

Leaf Residue: Subsection 5.3.2 mentioned that some conjectures seem "bigger" than others, pointing out that the proposition $KK(t, c)$ can be "decomposed" into the "smaller" pair, *Kirchoff1*(*t*) and *Kirchoff2*(*c*). Stated informally, *CTruep* never conjectures a "decomposable" fact, such as the $KK(t, c)$ proposition. It, instead, conjectures only its component parts, here *Kirchoff1*(*t*) or *Kirchoff2*(*c*) (or, if necessary, both of these facts).

This applies recursively: to satisfy a RKK query, *CTruep* does not consider conjecturing the full RKK proposition. Instead, it breaks RKK into its component clauses, the *Ohms*, *ConservedThru* and *KK* facts. Rather than stop here, it further splinters *KK* into *Kirchoff1* and *Kirchoff2*, as shown above.

The only propositions *CTruep* considers conjecturing are these "leaf propositions",⁶ clauses which are not composed of other "smaller" ones. This prevents it from simply conjecturing the goal query or from postulating some full RKK clause. (This is particularly useful if some component part, e.g., *Kirchoff1*(*t*), is known to hold.)

⁶A nicer name for these would be "atomic propositions", as this conveys the sense of "not decomposable". However, that term was already busy.

These are the intuitions. Stating them more formally is not so simple. Section 5.3 already discussed a semantic way of describing how “big” a proposition is; here we are seeking a syntactic way of approximating this criterion. We begin by defining *leaf propositions*, and use this to define a *leaf decomposition* of a relation. This, in turn, leads to our definition of a *leaf residue*, which is the only type of residue which *CTruep* considers.

First, we use $leaf_{IM}(\delta, Th)$ to mean that the proposition δ is a leaf proposition with respect to a theory, Th , and inference mechanism, IM :

Definition 19 $leaf_{IM}(\delta, Th)$ iff δ has no non-trivial IM -based derivation in Th .

When IM is a backward-chaining engine, $leaf_{IM}(\delta, Th)$ means that Th contains no rules whose conclusion matches the proposition δ .

Basically, this refers to propositions which have no non-trivial proofs. The intuition below conveys the general idea:

Intuition 2 $leaf_{IM}(\delta, Th) \iff [\Delta^{IM}(Th, \delta, \Psi) \Leftrightarrow \Psi = \{\delta\}]$.

(This Δ^{IM} relation is a slight variant of the Δ defined by Definition 13: While Δ is based on logical inference, \models , this modified Δ^{IM} is based on the (possibly incomplete) IM inference scheme.)

This means that these leaf propositions are the smallest units possible, for this given theory and inference mechanism. We can use them to define the *leaf decomposition* of a sentence σ with respect to the theory Th :

Definition 20 $LD(Th, \sigma, \{\delta_j\}) \iff$

$$\sigma \equiv \bigwedge_j \delta_j(x_1, \dots, x_n) \ \& \ \forall \delta_j \ leaf_{IM}(\delta_j, Th).$$

A given relation R may have several distinct (even disjoint) leaf decompositions.

In our system, **RKK** has the single decomposition:

$$LD(Th, \mathbf{RKK}(t, c, r, l), \left. \begin{array}{l} \text{Kirchoff1}(t) \\ \text{Kirchoff2}(c) \\ \text{ConservedThru}(t, l) \\ \text{Ohms}(t, c, r, l) \end{array} \right\}) \quad (6.1)$$

We now have the machinery needed to define the second part *CTruep*'s output: it is always a leaf residue. Given $R(x_1, \dots, x_n)$ as input, *CTruep* returns a binding list $[a_1, \dots, a_n]$ and a leaf residue, $\{\delta_j\}$, where each δ_j is a ground clause which is a member of $R(a_1, \dots, a_n)$'s leaf decomposition.

To state this formally, $\{\delta_j\}$ is the *leaf residue* of the sentence σ and theory Th , written $\Delta_{LR}(Th, \sigma, \{\delta_j\})$, if it is the minimal subset of σ 's leaf decomposition which must be added to Th (i.e., which needs to be conjectured) to derive σ :

Definition 21 $\Delta_{LR}(Th, \sigma, \Psi) \implies [\Delta(Th, \sigma, \Psi) \ \& \ \exists \Gamma (LD(Th, \sigma, \Gamma) \ \& \ \Psi \subseteq \Gamma)]$

By minimal, we mean that if there is another Ψ' which satisfies the right hand side of Definition 21, then $\Psi \subset \Psi'$. Hence, this Ψ is the minimal such set of leaf propositions which is needed to satisfy σ (i.e., $Th \cup \Psi \models \sigma$) and which is consistent with Th (i.e., $Consistent(Th \cup \Psi)$).

Two final comments:

[1] In what follows, we pretend that this Δ_{LR} relation is a function, mapping a theory and a sentence onto a set of sentences:

$$\Delta_{LR} : 2^S \times S \mapsto 2^S,$$

where S is the set of all legal sentences.

In general, a relation R may have many independent leaf decompositions, leading to many distinct possible leaf residues. As this does not happen in our examples, I was able to exploit this functional description.

[2] There are two potential problems with this definition of leaf decomposition: First, while it works fine for conjunctions and for universally quantified variables, disjunctions and existential variables can be quite problematic. (Reiter calls these

“indefinite terms”, and comments on their complexities in [Rei78].) This particular issue never came up in my system. This is because *NLAG* only considers conjecturing abstractions, and abstractions tend to be conjuncts of universally qualified clauses.

Second, this definition can lead to an infinite loop if the proof for some relation involves the same relation. (E.g., (if (and (parent \$x \$y) (ancestor \$y \$z)) (ancestor \$x \$z)).) We remove this complication by “taking the fix point” during the proof. This has the effect of omitting the recursive clauses: e.g., clauses in the antecedent of a rule which match the rule’s conclusion. (This means that $LD(Th, \text{ancestor}(a, z))$ is a set of parent clauses, as expected.) This issue, also, never occurred. Once again, this is because *NLAG* only considers conjecturing abstractions, and abstractions tend to be well-behaved — here, they tend not to be recursive.

*C*Truiep’s Algorithm

This portion describes *C*Truiep’s actual algorithm. For efficiency reasons, it performs this task in two stages: The first substep, *C*Truiep-1, attempts to satisfy the partially instantiated abstraction instance, using existential variable “place holders” as needed. The second substep, *Resolve-Exists*, then seeks consistent bindings for those existential variables.

*C*Truiep-1 begins by seeking all deducible bindings; that is, all $[a_1, \dots a_n]$ such that $Th \models S(a_1, \dots A, \dots a_n)$. It then finds the bindings which only satisfy the formula $S(?_1, \dots A, \dots ?_n)$. (This operating heuristic — find instantiations which require zero conjectures before ones requires more than zero conjectures — follows immediately from the H_{FC} rule. See Subsection 5.3.5.)

When satisfying the formula, *C*Truiep-1 ignores the issue of variable bindings when conjecturing a new leaf proposition (here, some δ_j), choosing instead to simply use an existential variable for each underdetermined parameter. This leads to a dummy binding list, which assigns each variable to an distinct existential variable, $?_i$.

This substep also returns the leaf residue associated with this bindings; that

is, a set $\{\delta_j\}$ which satisfy $\Delta_{LR}(Th, \mathbf{S}(?_1, \dots, A, \dots, ?_n), \{\delta_j\})$. (As the previous portion explained, insisting that each δ_j is a leaf proposition means that *CTruep-1* includes a proposition, δ_j , as a conjecture only if that axiom could not have been derived from other propositions via an already present rule.)

Hence, given **RKK**'s definition (Figure 4-3) and a starting theory *Th* essentially devoid of any FS facts, *CTruep-1* would determine

$$\Delta_{LR}(Th, \mathbf{RKK}(\text{FlowRate}, ?_c, ?_r, ?_l), \left\{ \begin{array}{l} \text{Kirchoff1}(\text{FlowRate}) \\ \text{Kirchoff2}(?_c) \\ \text{ConservedThru}(\text{FlowRate}, ?_l) \\ \text{Ohms}(\text{FlowRate}, ?_c, ?_r, ?_l) \end{array} \right\}), \quad (6.2)$$

where these $?_c$, $?_r$ and $?_l$ symbols are existential variables. (Notice that the non-analogue terms do not need to be existential variables. For example, if *Th* could prove that $\text{ConservedThru}(\text{FlowRate}, \text{Pipes})$ held and that ConservedThru is a function, it would use the *Pipes* binding rather than the $?_l$ shown above. Here, the leaf residue would be

$$\Delta_{LR}(Th, \mathbf{RKK}(\text{FlowRate}, ?_c, ?_r, \text{Pipes}), \left\{ \begin{array}{l} \text{Kirchoff1}(\text{FlowRate}) \\ \text{Kirchoff2}(?_c) \\ \text{Ohms}(\text{FlowRate}, ?_c, ?_r, \text{Pipes}) \end{array} \right\}). \quad (6.3)$$

However, as ConservedThru is not a function, this did not happen.)

This concludes *CTruep-1*'s operation. Its underlying goal is to find the sets $\{\delta_j\}$ s which satisfy $\Delta_{LR}(Th, \mathbf{S}(?_1, \dots, A, \dots, ?_n), \{\delta_j\})$. This $\{\delta_j\}$ set provides constraints on the legal values for these $?_i$. These variables, $[?_1, \dots, ?_n]$, and their associated constraints, $\{\delta_j\}$, are then given to *Resolve-Exists*.

This second substep seeks bindings for these variables, $\{?1 \mapsto a_1, \dots, ?n \mapsto a_n\}$, which satisfy this *Th* theory. (Of course, this substep is vacuous when there are no existential variables.) The current algorithm seeks only symbols already present in the \mathcal{L} , the language of *Th*: i.e., each $a_j \in \mathcal{L}$ for each j . (*N.b.*, this means that *NLAG* does not generate new terms. See Subsection 5.3.6 and Point N4 in Section 6.3.)

To find possible candidates, *Resolve-Exists* determines all of the facts known about each of the variables by forward chaining from their respective defining

clauses. For example, “ $?_c$ ” is defined by $\text{Kirchoff2}(?_c)$ and $\text{Ohms}(\text{FlowRate}, ?_c, ?3, ?4)$. These, in turn, lead to facts like $?_c \in \text{Functions}$, etc. (See Figure 5-5 in Section 5.3.6.) *Resolve-Exists* generates such a list for each variable and uses this explicit list of propositions to constrain the allowed values for that variable. This process then splits each list into two parts: those which deal only with this specific variable (here $\Pi^1(?_c) = \{\text{“Kirchoff2}(?_c)\text{”}, \text{“}?_c \in \text{Functions}\text{”}, \dots\}$) and those which involve more than one variable (here $\Pi^2(?_c) = \{\text{“Ohms}(\text{FlowRate}, ?_c, ?3, ?4)\text{”}, \dots\}$).

For each variable $?i$, *Resolve-Exists* then uses a set of heuristics to order the first sublist, $\Pi^1(?i)$. The goal is to place the maximally constraining clauses first. For notion, assume $\Pi^1(?i) = \langle \rho_0^i, \dots, \rho_m^i \rangle$. The first member of this list, ρ_0^i , is used as a generator, which (we pretend) generates all the possible candidates for this variable.⁷ This means the possible referents of this $?i$ variable are the bindings of this first proposition, as returned by (*TRUEPS* ρ_0^i). (The *H_{FT}* rule allows us to make this simplifying assumption.) For notation, assume the legal values for the variable $?i$ are $\{x_i^1, \dots, x_i^{k_i}\}$.

The remaining members of this first sublist $\Pi^1(?i)$, viz., $\langle \rho_i^1, \dots, \rho_i^m \rangle$, become the first wave of pruners for this variable: the value a_i^j is an acceptable value for $?i$ only if every one of these $\rho_i^k[?i/a_i^j]$ propositions is consistent with *Th*, for all k .

This, *Resolve-Exists* “1-Consist” test, removes some of the initial candidates. Without loss of generality, assume that $\{a_i^0 \dots a_i^{k_i}\}$ remain for each $?i$. We now have to find the ones which satisfy the other constraints, the $\Pi^2(?i)$ s. *Resolve-Exists* merges these propositions together, forming $\Pi^2 = \cup_{?i} \Pi^2(?i)$.

Now consider the cross-product of these sets of possible values, $\{a_i^j\}$. Each

⁷If $\Pi^1(?i)$ is empty, the generator is taken from $\Pi^2(?i)$. As this latter set is guaranteed to be non-empty, we will always find some generator. However, there are times when this clause cannot be used as a generator. (This often happens when *CTruep* is used solo: see Run#3-5 in Section 7.5.) To avoid returning no answers here, the *Resolve-Exists* system has a built-in heuristic: as a last resort, it uses the domain information associated with the underlying relation (screened with defined-below data, if present) to produce a set of possible values. This is as if there was a rule of the form:

(if (and (domain \$r \$i \$s) (\$r ... ?x ...)) (mem ?x \$s))

when the $?x$ is the i^{th} argument of the proposition. (The domain and defined-below relations are discussed in Section 7.2.)

member of this cross-product,

$$\langle a_1^{j_1}, \dots, a_n^{j_n} \rangle \in \{a_1^{j_1}\} \times \dots \times \{a_n^{j_n}\},$$

is then screened against the Π^2 list of propositions: *Resolve-Exists* applies the $\{?1 \mapsto a_1^{j_1}, \dots, ?n \mapsto a_n^{j_n}\}$ substitution to each member of this list $\pi \in \Pi^2$, and tests if this $\pi[?1/a_1^{j_1}, \dots, ?n/a_n^{j_n}]$ is consistent with *Th*.⁸ Those n -tuples which pass this second test are returned as *Resolve-Exists*' results. This binding, together with the related leaf residue (which is the instantiation of each member of $\Delta_{LR}(Th, S(?_1, \dots, A, \dots, ?_n), \{\delta_j\})$) is the result of the *CTruep* process.

To tie this back with the *NLAG* system: this instantiation and residue pair is then returned as the result of the final *Inst-Target* step, and is therefore returned as the result of the overall *ComAbs* process.

We close this description with some final notes: First, this facility can be very inefficient. One way of improving the situation is by "supercaching" various results. Point MRS1 discusses how to do this. Second, finding the appropriate leaf decomposition of a sentence often involves more than simple backward chaining. Point MRS3 discusses one useful embellishment.

The final issue deals with two decisions left implicit in the above discussion: [1] how we order the various clauses (i.e., should (mem ?3 functions) be considered before (domain ?3 3 2 numbers)), and [2] how to order the sets of consistent values (i.e., should the [flow-rate pressure-drop cross-section pipes] instantiation be considered before [flow-rate pressure-drop cd-of tubes]). While the general \parallel_{PT} process is silent on both issues, the particular *NLAG* process uses a set of *ad hoc* rules to provide an ordering.

How should we order the various clauses? One approach would involve using the most constraining clause as the generator. Here we would need to find the maximally constraining clause. ([SG85] describes ways of doing this.) At the other extreme, a more cautious and thorough method would advocate finding every possible abstraction instantiation H_{FT} would allow. To be totally comprehensive, we would have to use *all* of the clauses as generators.

⁸This is a quick test. The *Verify* module later tests each proposition against that theory unioned with the other propositions proposed: see Subsection 6.1.1.

NLAG, instead, employs the simpler convention of using a set of inexpensive (and changeable) meta-rules to select which clause should be the generator. One version insists that the generator is of the form (mem ?i *(class)*). Another version uses the relation (InTheory ?i theory_{Abst}), where theory_{Abst} is the theory associated with the abstraction, *Abst*. Both have been implemented; others may be proposed and tested. (These are described in Section 7.2.)

These operations lead to a set of consistent values, each a potential instantiation of the abstraction in the target domain. The second decision is how these possible instantiations should be ordered: *i.e.*, in what order should they be given to *Verify* for further testing and possible presentation to the user? *NLAG* assigns a value to each *n*-tuple of entries. The tuple with the highest value is returned first, then the second highest, and so on. This value is computed as follows:⁹

- **Lookup Clauses:**

Associated with each existential variable is a set of constraints used to generate and prune the legal values for this variable. (For example, both (mem ?3 functions) and (domain ?3 3 2 numbers) are associated with the variable ?3.) Now consider some particular instantiation, *e.g.*, {?2 ↦ pressure-drop, ?3 ↦ cross-section, ?4 ↦ pipes}. (This corresponds to the target conjecture **RKK**(flow-rate pressure-drop cross-section pipes).) In how many of the above clauses does this proposed value for ?3, cross-section, “primitively participate”? For example, are either (mem cross-section functions) or (domain cross-section 2 numbers) primitively stored in the initial theory? *NLAG* gives the proposed instantiation one point for each such primitively stored fact, for each existential variable. This means it adds to this value the number of ?2-related facts which contain the pressure-drop term and the number of ?4-related facts which contain pipes. As it is adding the number of primitively stored facts associated with each variable, finding that (domain cross-section 1 pipes) is primitively stored counts as two — one for ?3 ↦ cross-section and one for ?4 ↦ pipes.

This contribution is computed by the *Weight-By-Lookup* clause.

⁹The final rule presented in SubAppendix B.1 shows the actual code used.

- **Associated Terms:**

Consider the symbols lexically contained in the query (here (flow-rate j-wc-a pipe1 s0 \$fr)). Now find all the primitively stored facts which include any of these symbols, and then consider the set of symbols contained in these facts. (This describes a two step spreading-activation process.) For example, the term pipes leads to the sentence (mem pipe1 pipes), and this provides the associated term, pipes. This search quickly finds the relevant hierarchy information. (E.g., we see how pipe1 points to pipes.) Any candidate which includes any of these resulting terms is given an additional five points.

This is computed by the *Weight-By-Assoc* clause.

- **Common Theory:**

This begins by considering the theories associated the symbols of the target query. (E.g., pipe1 is associated with the theory wc1.¹⁰) Any candidate instantiation which includes a term associated with any of these theories is awarded an additional ten points. This measure does not consider the hierarchically included theories. Hence, a term immediately associated with LELS receives no points.

This is computed by *Weight-By-CommonTheory* clause.

6.2.4 Summary of MRS Additions

NLAG necessitated several minor embellishments to the MRS system, in addition to the *FC-Find* and *CTruep* facilities discussed above. This subsection presents a list of some of these additions. (Some of these facilities have already been incorporated into the core MRS system and others are available as loadable packages. The points below indicate the status of each package.)

MRS1. "SuperCaching":

After *TRUEP*ing a proposition ρ , it would be desirable to be able to re-use those answers during a later computation. The current caching facility does some of

¹⁰SubAppendix B.2.1 sketches the relevant theories.

this by storing intermediate results, knowing that subsequent *TRUEP* calls will find those stored values first. If MRS knows that these stored answers are the only ones, it can simply return these answers; otherwise it is forced to consider back-chaining to find other answers. Unfortunately, MRS seldom knows the number of solutions to a query; in fact, the only times it knows it can stop are when a query can return only a single answer. This means that caching is ineffective whenever a query might return more than one answer or no answer. Here, as MRS does not know that these stored answers are the only ones, it will continue backtracking, albeit needlessly.

This *SuperCache* facility addresses this problem. Once all solutions to the query ρ have been produced, the *SuperCaching* mechanism stores those values. A subsequent (*TRUEP* ρ) invocation uses this information; *TRUEP* now knows it can simply stop and return just those values, rather than continue backward-chaining.

This requires explicitly associating those answers with the query ρ . As an example, imagine that query (Dog \$x)¹¹ returned the two answers (Dog Fido) and (Dog Shep). This *SuperCache* package then stores the fact (BCS-of (Dog \$x) *ts* (Dog Fido) (Dog Shep)), where *ts* is a time stamp. The upgraded *TRUEP*ing mechanism is smart enough to use these values the next time this query (or any specialization) is sought. This means that (*TRUEP* '(Dog \$y)) will use this BCS-of fact to return (Dog Fido) and (Dog Shep) without any additional backward chaining. Furthermore, this same BCS-of fact tells *TRUEP* that the query (*TRUEP* '(Dog Felix)) should fail.

This *SuperCache* facility can be embellished to worry about the repercussions of additional user-input facts. This is why the *ts* time-stamp is included.

CTruep made constant use of this facility. For example, when computing the residue for the **RKK** abstraction, it found, as an intermediate result, the residue associated with the **KK** abstraction. This *SuperCache* facility allowed this computation to be re-used during *CTruep*'s subsequent dealings with **KK**,

¹¹Variables in queries are prefaced with a dollar sign, "\$".

both directly and indirectly (e.g., when considering the CKK abstraction.) (See SubAppendix B.3.1, especially page A-89.)

As a final note, the astute reader might imagine a slightly simpler process, one which simply stored the number of solutions to a query, e.g., ($\#BCS \rho \langle n \rangle$), as well as caching those $\langle n \rangle$ answers. In the above example, this would mean storing ($\#BCS (\text{Dog } \$x) 2$) in addition to caching (Dog Fido) and (Dog Shep). While this would work with the general *BC* backward chainer, it is not enough for the *CTruep* residue process. For this, we need to store the actual conjectured propositions, as shown above.

MRS2. Use of Constraints:

In the standard MRS system, the only direct way to compute the value of a parameter is by deducing its value from other terms. While this works well in many situations, it is sometimes more natural to describe the interconnections between various parameter in terms of *constraints* on their mutual values. This can be awkward to write using a strictly rule-based system. For example, the typical way of describing Ohm's Law is by stating that the product of the current and the resistance of a resistor is the voltage drop through that device. This is far more succinct than the alternative of stating three rules:¹²

1. to compute the voltage drop: determine the resistance and the current, then find their product;
2. to compute the resistance: determine the voltage drop and the current, then find their quotient;
3. to compute the current: determine the voltage drop and the resistance, then find their quotient.

Similarly, it is easier to state that the voltage drop between two points is defined as the difference between the voltages of those points, rather than as a triplet of specific rules.

¹²In its full generality, this could require having one rule for every one of the 2^n possible subsets of values. Each such rule would be responsible for deducing or constraining one of the parameters based on the values of the other $m < n$ known values.

The situation gets even messier when the only way to compute one quantity is to compute a value of a related term; and this related value is, in turn, dependent, on the initial quantity. Clearly, the way to solve such problems is by satisfying constraints rather than by pure backward-chaining.

This motivated various researchers to consider a system of constraints. (*Cf.*, [SS77] and [SGLS80].) For this circuit work, I too designed a simple constraint propagation system, one capable of both describing constraints which interrelate various values and of symbolically using this intentional information to solve problems.

This point describes the culmination of this work, the constraint package. First, we need to define the format of a constraint. By example, the Ohm's Law constraint is written

```
(constraint (* $c $r $vd)
            (and (port $x 1 $j1)
                 (mem $x elect-devices)
                 (port $x 2 $j2))
            (resistance $x $r)
            (voltage-drop $j1 $j2 $s $vd)
            (current $j1 $x $s $c))
```

The top line means that this fact describes a multiplication constraint. The next s-expression is a screening clause, which defines when this rule may be pertinent and establishes some bindings. Here, it specifies that the variable \$x must be bound to an electric device, whose two ports are (the values of) \$j1 and \$j2. The remaining clauses determine the other bindings, in particular, for the variables in the "header", (* \$c \$r \$vd). (The difference between the first clause — here "(and (port ...) ...)" — and the others is that the first one is *LOOKUPed* while the others are *TRUEPed*.) The other clauses state that \$r is bound to the resistance of the device \$x, \$vd, to the voltage drop across the two ports associated with that device (recall these ports are the values of \$j1 and \$j2)

during the situation $\$s$, and $\$c$, to the current at the junction $\$j1$ through the same device during the same situation.

Many of these constraints may come into play when solving a complex problem. The *Solve* subroutine attempts to find solutions to such systems of equations.¹³

This overall facility is tied into MRS's basic backward chaining system, meaning that the query (*TRUEP* '(current ...)) will use the Ohm's Law constraint shown above, if necessary.¹⁴ These constraints have also been integrated with the "Iff" facility described below in Point MRS3 below.

MRS3. Double Backward Chaining: "Iff":

When a typical backward chainer (e.g., MRS's *BC*) attempts to prove a proposition, ρ , it finds rules of the form (if σ ρ), and then considers σ to be a subgoal. However, the initial *Th_{CF}* knowledge base included many rules of the form (if τ (if σ ρ)): for example, (if (kirchoff1 $\$t$) (if (...) ($\t $\$j$ $\$d$ $\$value$))). Here, even though both τ and σ may be provable, the standard backward-chainer would not use this indirect rule to prove ρ . This point describes the "double back chain" facility, which addresses this limitation.

In the example suggested above, when trying to prove a clause which matched ρ , this "Iff" addition would try to prove τ . If that succeeded, it would cache the rule (if σ ρ). (Technically, proving τ leads to a binding list, *al*. The rule stored is (if σ' ρ'), where σ' (respectively ρ') is the result of applying the substitution *al* to σ (respectively ρ).) This new rule can then be used toward proving our initial goal, ρ .

Three final comments:

[1] This addition is fully recursive. It can use a rule of the form

$$(\text{if } \tau_1 (\text{if } \tau_2 (\text{if } \dots (\text{if } \tau_n \rho) \dots)))$$

¹³In certain complex situations, this constraint solving system simply asks the user for a binding to the relevant variables. See Note *Trace3* in SubAppendix B.3.1. (After all, this research is on analogical reasoning, not algebraic simplification and symbolic evaluation.)

¹⁴This mechanism gives rise to the mysterious *useconstraints* clauses which appear throughout SubAppendix B.3.3.

to prove a query ρ .

[2] This “find-embedded-methods” facility is fully integrated with the Constraint package described in Point MRS2 above. That is, the rule (if (kirchoff1 \$t) (constraint ... (\$t ...) ...)) is considered when trying to prove something matching (\$t ...).

[3] This facility is only active when the IfIfP flag is nonNIL. As this operation is expensive, we only turn this flag on when we want the system to try as hard as possible to tease everything possible out of the available facts. It was never used with *NLAG*, since this system finds the facts it needs by working in the forward direction. The only time it was used was when *CTruep* was working by itself. (See Run#3-5 in Section 7.5.)

MRS4. Generator:

Consider the problem of deducing the possible answers to a conjunction of clauses. The standard approach is to produce all possible answers for the first clause, and then use the remaining clauses as filters. Unfortunately, that first query might produce a large, or even infinite, number of possible answers.

In these situations, a better approach involves generating one value at a time. This single value can then be tested against the other clauses. Only if that first entry fails does the generator then produce another possible answer, and so on.

Unfortunately, MRS did not provide a general facility for this. I wrote such a system, one which included the hooks needed to associate LISP procedures which returns these one-at-a-time answers.

(This module is now a loadable package.)

MRS5. Existential Variables:

The initial MRS system has no mechanism for dealing with existential variables. That is, there is no way of stating that we do not yet know the precise value of some argument of a formula, one which may be co-referential with some known term. This was required by the *CTruep* subroutine: see its description in Subsection 6.2.3.

MRS6. REPN and REPN-METHOD:

These MRS relations allow the user to easily declare the representation to be used for a class of propositions. I also added an assortment of new representations which *NLAG* required.

(This facility has been incorporated into MRS's starting code.)

MRS7. FUNPROC and RELNPROC:

These MRS relations allow the user to add a procedural attachment to an MRS function or relation.

(This facility has been incorporated into MRS's starting code.)

MRS8. Rule direction information:

This package allows the user to declare which way a rule should be invoked: exclusively in a forward direction, exclusively in a backward direction, or both (the default). This was implemented by storing a meta-level tag with each rule.

(This was motivated by Novak's GLISP work [Nov83].)

(This facility has been incorporated into MRS's starting code.)

6.3 Requirements of the *NLAG* System

We claimed earlier that the *NLAG* algorithm is a faithful model of the theoretical \Vdash_{PT} process: i.e., that it does return the same answers \Vdash_{PT} would. This section proves this claim. In particular, it supplies an explicit description of what is necessary for the *NLAG* algorithm to work effectively. These features constrain both the source and target domains and the underlying computer language in which an *NLAG*-like system can be written.

First, just what does *NLAG* return? By construction, the *ComAbs* algorithm produces only abstractions common to both analogues. For each such result to pass *Verify's* test, it must lead to an answer to the initial *PT* query. Hence, *NLAG* only returns useful common abstractions. It may, however, not be the correct abstraction: that is, this common abstraction may lead to an incorrect answer to *PT*. All we can guarantee is that any new conjecture is "reasonable"; that it can

be justified given these inputs and the starting knowledge base, Th . (See Note:1-2.) This shows that *NLAG*'s algorithm is sufficient; that it never produces useless or unreasonable common abstractions.

Now for the other direction: Will *NLAG* always find the correct answer, eventually? The answer is a qualified "Yes". The *NLAG* algorithm can only find an abstraction (and hence, can only find the desired abstraction) when the following conditions are met:

N1. The starting theory, Th must include a definition of the relevant abstraction.

This means a fact of the form " $\forall x_i \text{ RKK}(x_1, \dots, x_n) \Leftrightarrow (\& \dots)$ " must be included in Th , for the needed abstraction. (See Figure 4-3 on page 52.) The theory must also include a fact of the form $\text{Abstraction}(\text{RKK})$.

N2. The source instantiation of the abstraction must be derivable for the initial theory, and the target instantiation must be consistent. That is, $Th \models \text{S}(b_1, \dots, B, \dots, b_n)$, and $Th \not\models \neg \text{S}(a_1, \dots, A, \dots, a_n)$. Notice this means that *NLAG* does not have to conjecture any facts to see that $\text{S}(b_1, \dots, B, \dots, b_n)$.

N3. A full instantiation of the appropriate abstraction must be sufficient to solve the target problem — i.e., we assume that nothing else needs to be conjectured. This is what we mean by $Th + \text{S}(a_1, \dots, A, \dots, a_n) \models PT$.

This turned out to be a non-trivial requirement. For example, imagine [1] we used heuristic H_{JK} to determine the analogous source domain problem $PS =$ "Find the current" from the given target problem "Find the flowrate", [2] that PS 's solution depended critically on some source fact which involved Voltage, $\phi(\text{Voltage})$, and [3] that the corresponding target conjecture $\phi(\text{Pressure})$ is required to solve PT . Now recall that the source abstraction instantiation dealt with VoltageDrop, not Voltage. This means we are viewing the concept Pressure as an unseen dependent of this **RKK** abstraction, with respect to this formula ϕ .

We are making the strong assumption that we can derive all such dependent statements, given only the newly conjectured abstraction instance. Here, this

means that

$$Th + \mathbf{RKK}(\text{FlowRate}, \text{PressureDrop}, \text{PipeCharacter}, \text{Pipes}) \models \phi(\text{Pressure}).$$

Hence, the only fact which needs to be conjectured is this abstraction instance; everything else needed must follow from it.

N4. All the concepts needed to instantiate the abstraction in target domain must already be included in \mathcal{L} , the language of Th . This means each instantiating concept must already be reified as a pre-defined symbol. For example, *NLAG* would fail if \mathcal{L} did not already include a symbol for the *PressureDrop* concept which is needed to fill the second position of the **RKK** abstraction. This does *not* mean that Th must include all the relevant facts about this *PressureDrop* concept. Indeed, we may know arbitrarily little about it: *NLAG* can find this *PressureDrop* symbol even if it was included in only a single Th fact (the one needed to make this *PressureDrop* symbol findable in the sense required by the H_{FT} rule: see Subsection 5.3.6).

N5. Each term used to instantiate the abstraction in the target domain must be findable. To describe this, we need to use the material implications from **S**, $M = MI(\mathbf{S}(\?_1, \dots, A, \dots, \?_n))$. (See Definition 18.) For each $\?_j$ in $[\?_1, \dots, \?_n]$, we can find a subset of M which deal with this j^{th} variable. Using the rules described on page 127, we can order this into the list $\langle \rho_j^0 \dots \rho_j^{n_j} \rangle$, where this symbol $\?_j$ is lexically included in each ρ_j^k and each ρ_j^k is in M .

For each j , assume the desired value of each $\?_j$ is a_j . For this value to be found, $(\mathbf{TRUEPS} \rho_j^{j_0})$ must return a binding list which includes $\?_j \mapsto a_j$.

(Each $\rho_j^k[\?_j/a_j]$ must also be consistent with Th , but Point N2 already enforces this constraint.)

If all the above conditions are satisfied, the *NLAG* system is guaranteed to produce that correct answer, eventually. It may, however, find other answers, and may find those other answers first. That is, (i) *NLAG* may first find a different common abstraction or (ii) even if it finds the correct abstraction, it may first

find another instantiation. This depends heavily on the heuristics discussed in the previous two chapters and their interactions with the particular facts in the initial theory, Th . The intuitions below approximate the operations of this $ComAbs$ generator:

E1. To find the correct instance of the abstraction in the source domain *early*:

That instance should be “close” to the kernel facts — *i.e.*, either explicitly mentioned or quickly found by a forward chaining process. That is, the proof of $S(b_1, \dots, B, \dots, b_n)$ from the kernel facts should be short.

Note:6-1 quantifies what it means to for a proposition, σ (here, the source abstraction instance $S(b_1, \dots, B, \dots, b_n)$), to be “close” to a particular set of facts, Φ , (here, the kernel) with respect to the theory, Th .

E2. To find the correct instance of the abstraction in the target domain *early*:

There should be enough facts in the target domain that the correct bindings for the remaining variables of the abstraction are found soon.

This is easy to quantify in terms of the Δ_{LR} relation defined in Definition 21: namely, the sentence φ will be found before the sentence ρ whenever

$$\|\Delta_{LR}(Th, \varphi)\| < \|\Delta_{LR}(Th, \rho)\| \quad (6.4)$$

(This simply paraphrases the H'_{FC} heuristic, using Δ_{LR} *qua* function.)

In case of ties, $\|\sim_{PT}$ does not specify an order. Our particular $NLAG$ implementation prefers those instantiations whose derivations require fewer steps. (This can be quantified using an obvious variant of the K_i relation discussed in Note:6-1.)

Chapter 7

Experimental Results

This dissertation has provided a particular definition of analogical inference, based on finding common abstractions. The previous chapter described the *NLAG* routine which implemented this \Vdash_{PT} process. This chapter discusses the utility of these (up until now, just ethereal) ideas, demonstrating that they are not only intuitive,¹ but realizable and general as well.

The first section sets the stage, describing which facts, rules and abstractions are, and are not, present in the initial knowledge base. The next four sections present a series of demonstrations and experiments using this data, designed to demonstrate the viability of this abstraction-based analogical inference process. Table 7-1 provides a preliminary overview of these tests. (Many of the terms used in this sketch are clarified in the various sections.) Each deals with the specific example presented in Section 1.3, or a closely related variant, and all use the *NLAG* routine. Section 7.6 then analyzes this data and proposes conjectures on the nature and utility of abstractions in general.

Two other preliminary comments: Two other analogy situations were considered and partially implemented. The first, dealing with the **Group** abstraction, is sketched in Note:7-2. The other, based on the Programmer's Apprentice work, appears in Note:7-3. Second, we make frequent references to the algorithm presented

¹Note:7-1 addresses this claim, discussing some of the ways this model of analogy fits the standard, pre-theoretical sense of analogy.

1. Functionality — Section 7.2

- Run#1-1: Using Generator#1
- Run#1-2: Using Generator#2

2. Sensitivity

(a) Initial Theory -- Section 7.3

- Run#2a-1: Add Irrelevant Facts
- Run#2a-2: Delete Relevant Source Facts
- Run#2a-3: Add Directly Relevant Target Facts
- Run#2a-4: Add Indirectly Relevant Target Facts

(b) Abstractions — Section 7.4

- Run#2b-1: Add Irrelevant Abstractions
- Run#2b-2: Delete Relevant Abstractions
- Run#2b-3: Add Relevant Abstractions

3. Ablation of “Analogy” -- Section 7.5

- Runs#3-2a, #3-2b: \neg Maximally General
- Runs#3-3a, #3-3b: \neg Abstraction
- Runs#3-4a, #3-4b: \neg Maximally General, \neg Abstraction
- Runs#3-5a, #3-5b: *CTruep* alone

Table 7-1: Overview of Tests Run

in Chapter 6 in describing and analyzing the experiments. The reader is advised to that chapter a strong prerequisite for this one.

7.1 Initial Set Up

This section quickly sketches the basic set up, corresponding to the “Find the flowrate” problem first presented in Section 1.3. (SubAppendix B.2 describes this information in detail.) Each of the experiments described below either uses this exact set of facts, or some slight variant.

To be compatible with the actual data, as shown in Appendix B, the expressions appearing in this chapter are the ones actually used by the *NLAG* program. This

means that the expressions are written in standard LISP notation (i.e., (kirchoff1 current) rather than Kirchoff1(Current)) and everything will be in the same font and lower case.

First, NLAG's specific inputs were:

```
Query:          (flow-rate j-wc-a pipe1 s0 $fr)
Analogical Hint: ((current . flow-rate))2
```

The initial knowledge base, Th_{CF} contained the following four lumped element linear system abstractions:

kk which consists of Kirchoff's two laws.

rkk which represents resistor circuits. It includes the resistance law (aka Ohms Law) and conservation rule for the through-term (e.g., flow-rate) through the load element (e.g., pipes), in addition to Kirchoff's Laws. This is exactly the relation shown in Figure 4-3 in Section 4.1.

ckk which represents capacitor circuits in the same way rkk represents resistor circuits; i.e., it includes the capacitance law in addition to Kirchoff's Laws.

lkk which represents inductor circuits in the same way rkk represents resistor circuits; i.e., it includes the inductance law in addition to Kirchoff's Laws.

(These, and all other abstractions, are described in the glossary, Appendix C. Their actual MRS code appears in SubAppendix B.2.1. That subappendix also discusses the other general information included in the initial theory, Th_{CF} — these are the facts which are in common to the two subtheories, EC and FS.)

Th_{CF} also includes a fairly complete collection of electricity facts (EC), including the needed $LD(Th_{CF}, (rkk \text{ current voltage-drop resistance elect-devices}))$ ³

²This ((current . flow-rate)) notation encodes the same information as the {FlowRate \mapsto Current} form shown in the previous chapter.

³The class of elect-devices consists of both resistors and wires.

clauses:

(kirchoff1 current)
 (kirchoff2 voltage-drop)
 (conserved-thru current elect-devices)
 (ohmslaw current voltage-drop resistance elect-devices)

It also includes domain specification for the variety of functions, including
 capacitance current inductance resistance voltage voltage-
 drop volt-batt ...

As one example,

(mem voltage-drop functions)
 (domain-spec voltage-drop junctions junctions states numbers)

means that voltage-drop is a function of three arguments — the first two are
 junctions, the third is a state, and the result (the final argument of the relation), is
 a number.

It also includes facts about classes of EC objects, including

batteries capacitors elect-devices inductors resistors wires
 ...

as well as things like units,

amperes colombs ohms volts ...

(Full details of this EC subtheory appear in SubAppendix B.2.2.)

On the other hand, there were relatively few facts about hydraulics systems
 (FS). This FS subtheory contained little beyond the information which could be
 inferred from the Section 1.3's Figure 1-1: just domain specification facts, about
 the functions

cd-of cross-section flow-rate pipe-charact pipe-shape pres-
 sure pressure-drop pump-press water-height ...

and the classes

orifices pipes tanks thin-pipes tubes water-devices ...

In particular, it included none of the clauses in

`(ThCF, (rkk flow-rate pressure-drop pipe-charact pipes)),`

e.g., it did not include `(kirchoff1 flow-rate)`. (SubAppendix B.2.3 explicates this set of facts.)

The problem statement contains descriptions of the devices and junctions,

`pump1 pipe1 pipe2 j-wc-a j-wc-b`

e.g.,

`(mem j-wc-a junctions)`

`(port pipe1 1 j-wc-a)`

`(flow-rate j-wc-a pump1 s0 -50.0)`

`(cross-section pipe2 1000.0)`

(Full details appear in SubAppendix B.2.4.)

7.2 NLAG's Functionality — Runs#1

Given all of this data, did *NLAG* work? This first demonstration (Run#1-1) shows that the answer is “Yes”. This run demonstrates the *NLAG* system in operation, describing how it solves the “Find the flowrate” problem.

NLAG did find the correct answer: the target abstraction instance,

`(rkk flow-rate pressure-drop pipe-charact pipes),` (7.1)

via the source abstraction instance,

`(rkk current voltage-drop resistance elect-devices),` (7.2)

This required conjecturing all of

`(kk flow-rate pressure-drop)`

`(kirchoff1 flow-rate)`

`(kirchoff2 pressure-drop)` (7.3)

`(conserved-thru flow-rate pipes)`

`(ohmslaw flow-rate pressure-drop pipe-charact pipes).`

That information, in turn, led to the rules and constraints needed to solve the problem. An actual trace of *NLAG*'s behavior when performing Run#1-1 appears in SubAppendix B.3.1. Various peripheral information appear in nearby subappendices; in particular, SubAppendix B.3.3 explicates the specific set of proposed conjectures.

As Section 2.5 mentioned, the analogy problem is underconstrained. *NLAG* also considered several other possible formulae and various other possible instantiations. One was

$$(\text{rkk flow-rate pressure-drop cross-section pipes}). \quad (7.4)$$

Based on information *NLAG* had (just domain specifications), this cross-section binding looked perfect; it just happened to be wrong.

In fact, *NLAG* considered a variety of different abstraction instances. After summarizing this set, this preliminary subsection concludes by discussing the size of search space.

NLAG explicitly considered ten different possible target instantiations of (rkk flow-rate ?2 ?3 ?4), where each ?i is an existential variable. These are shown in SubAppendix B.3.4. Five of these were useful; i.e., lead to an answer to the target problem.

NLAG also considered each of the three other abstractions. Thanks to H_{JK} 's advice, the next abstraction considered was kk. Its only proposed instantiation was (kk flow-rate pressure-drop). This was rejected since it was not a useful analogy: i.e., this conjecture does not lead to a solution to the "Find the flowrate" problem. (In SubAppendix B.3.1, this analogy corresponds to the ##11th analogy candidate.)

NLAG next considers ckk. Here, too, there is but a single possible target instantiation: (ckk flow-rate pressure-drop water-height tanks). (The water-height term satisfied what turned out to be the most constraining condition associated with ckk's third argument, viz., that it be a function of *three* arguments.) As this conjecture does not solve the "Find the flowrate" problem, it too was discarded.

The final abstraction, lkk, also had but a single possible instantiation, (lkk

flow-rate pressure-drop water-height tanks). (This abstraction also imposed the same ternary-function condition on its third argument.) SubAppendix B.3.1 shows that *NLAG* is unable to find any other common abstractions when this final candidate fails, and gives up. (Of course, *NLAG* has already found the desired answer, the (rkk flow-rate pressure-drop pipe-charact pipes) one shown above. It was vetoed to further exercise *NLAG*.)

All told, *NLAG* found these thirteen proposed \approx analogies. Only five of these were \approx_{PT} analogies; i.e., only five target abstraction instances produced an answer to the “Find the flowrate” query. (Tables 7-3 and 7-4 summarize this information.)

How big is the actual search space? The MRS environment contained some 636 distinct symbols.⁴ Even assuming we knew the common abstraction would be rkk, there are still some $636^3 = 257,259,456$ different possible instantiations. For all four possible abstractions, the total is 771,779,004.

How many of these instantiations were actually considered? This number depends on which proposition played the rôle of generator for each existential variable. As the H_{FT} rule leaves this question open (see Subsection 5.3.6), I experimented with different generators.

Recall Subsection 6.2.3’s description of the *CTruep* algorithm: *NLAG* determines, for each variable, which associated proposition had the highest score. This proposition is used as the generator. Here, this ranking of atomic propositions is based on a user-defined ordering of their respective relations. The next subsections describe two different ways of ranking these relations; these systems select different propositions as generators.

7.2.1 Using Generator#1

Using the “Generator#1” system, the preferred generator is the set membership test: i.e., this version defines the eligible constants for each existential variable as the members of a particular set.

⁴This includes the virgin MRS system, plus miscellaneous additional facts which describe hierarchy, etc. Of the total of 1,357 facts (which included some 128 rules), only the facts listed in SubAppendix B.2 are directly used.

| Var | Generator | Initial | Final |
|-----|------------------------|---------|-------|
| ?2 | (mem ?2 functions) | 16 | 1 |
| ?3 | (mem ?3 functions) | 16 | 4 |
| ?4 | (subclass* ?4 devices) | 8 | 8 |

Table 7-2: Generated and Accepted Values for Variables

Its relation-ranking function considers the following relations to be special, and ranked them in the order:

$$\text{subclass* mem arity domain} \tag{7.5}$$

The other relations were ranked equally, below these four. (The one exception is the defined-below relation, which associates a symbol with some theory. That relation was simply ignored.)

Table 7-2 shows the three propositions which served as generators for the three existential variables. (These can be seen in SubAppendix B.3.1's trace.) We see that there are 16 qualifying functions and 8 known subclasses of devices.⁵ The other constraints quickly honed these values to 1, 4 and 8. (The corresponding data for the other three abstractions appear in SubAppendix B.3.1.)

As per Subsection 6.2.3's description, *CTruep* prunes these candidate entries in two phases. Table 7-2 shows only the first, which uses the single-variable clauses. (For example, the second row shows there are 16 possible values generated for ?3, generated using the clause, (mem ?3 function). Only 4 of these values pass the 1-Consist test; i.e., only these satisfy all of the $\Pi^1(?3)$ propositions.) The far right column corresponds to the "After 1-Consist:" row in Table 7-3. The second phase involves all *n*-tuples of acceptable entries; Table 7-3's "After *n*-Consist:" row shows that only 10 of the 32 possible entries pass this test. Notice that the space of a quarter of a billion possible rkk abstraction instances is cut to about 2,000 by the use of a good generator, and then to a mere ten by these consistency considerations.

⁵These numbers would probably be much larger in a general data base: i.e., there may be hundreds of known functions. Of course, most would be quickly pruned by the subsequent filters. This was one reason I considered the other possible generator. See Run#1-2, discussed in Subsection 7.2.2.

| | rkk | kk | ckk | lkk | ALL |
|--------------------------------------|--------------------|-----|--------------------|--------------------|--------------------|
| Generated: | 2,048 ¹ | 16 | 2,048 ¹ | 2,048 ¹ | 6,160 |
| After 1-Consist: | 32 ² | 1 | 16 ³ | 16 ³ | 65 |
| After <i>n</i> -Consist: | 10 | 1 | 1 | 1 | 13 |
| Useful: | 5 | 0 | 0 | 0 | 5 ⁴ |
| | # of Deductions | | | | |
| <i>Find-Kernel</i> : | | | | | 33 |
| <i>Inst-Source</i> : | 14 | 18 | 10 | 7 | 50 ⁵ |
| <i>CTruep-1</i> : ⁶ | 25 | 13 | 18 | 18 | 74 |
| <i>Resolve-Exists</i> : ⁷ | 401 | 118 | 315 | 316 | 1,150 |
| <i>Verify</i> : ⁷ | 284 | 75 | 17 | 17 | 393 |
| TOTALS: | 724 | 224 | 360 | 358 | 1,708 ⁸ |
| (% <i>Resolve-Exists</i> : | 55% | 53% | 88% | 88% | 67%) |

Notes:

- 1 In all three cases, this 2,048 was derived as the product of $16 * 16 * 8$; see Table 7-2.
- 2 This 32 is $1 * 4 * 8 = 32$; see Table 7-2.
- 3 In both cases, this 16 is $1 * 2 * 8$.
- 4 Entry number ##4 is the "correct" answer. It required some $535 + 4 = 539$ steps.
- 5 This total of 50 reflects the 49 deductions shown on this row, plus 1 additional entry, run after the final abstraction (lkk) was considered. (Here, *Inst-Source* determines that there were no other abstractions to consider.)
- 6 This refers to the first part of *Inst-Target*, up until the call to *Resolve-Exists*. Hence the total time spent in *Inst-Target* is the sum of this row and the next (the one labeled *Resolve-Exists*).
- 7 Tables B-1 and B-2 (in Point Trace4 on page A-94) derive these estimates for both *Resolve-Exists* and *Verify*.
- 8 This figure reflects the other numbers shown above (and to the left); plus an additional 8 steps: 4 to initialize the *NLAG* system, and 4 to terminate it.

Table 7-3: Data for Run#1-1 (using Generator #1)

The subsequent usefulness test weeds out half of the candidates, leaving only five possible useful analogical inferences. Of these, the correct (rkk flow-rate pressure-drop pipe-charact pipes) answer is the 4th useful analogy found. (It is also the #4th overall analogy considered, as all three earlier analogies were also useful.)

(Once again, the trace in SubAppendix B.3.1 describes corresponding figures for the other abstractions. This section only presents the quick summary shown in Table 7-3.)

Table 7-3 also presents timing information. Each of these numbers corresponds to the number of deductions required. Hence they include all backward and forward chaining steps and all conjecturing steps, as well as all of the start up time. (Certain control-related tasks present unavoidable fence-post problems when determining these figures. For example, there is no systematic way of determining which module is responsible for the tasks involved in returning an answer. So these numbers, while approximately correct, may be off by 1 or 2 in either direction.)

These figures are broken down in terms of the module involved. The *Find-Kernel* number is independent of the abstractions, as is the time spend in various overhead tasks. Each of the other figures can be attributed to one of the abstractions. The *Inst-Target* module is split into its two components, *CTruep-1* and *Resolve-Exists*.

There are several important bits of information. The two most relevant are: [1] This *NLAG* process is non-trivial: Its task, of finding the possible abstractions and then reducing the space from about a billion possible instantiations down to the five shown to the user, required over 1,700 deductions. [2] The *Resolve-Exists* module took a large chunk of *NLAG*'s time. Based on this example, it required about two-thirds of the processing time. (Section 7.6 later returns to this observation.)

Answers not found:

We see that a lot of analogies are found. There are other possible abstraction instantiations (which might suggest other analogies) which are not found. For example, the type information alone reduced the space to the 6,160 potential abstraction instances which *NLAG* generated. For example, when considering the value to use for *rkk*'s second argument, *NLAG* only considered functions (and not classes nor units

nor people). Of those, it further narrowed down the possibilities by considering only functions of three arguments.

The next sections consider how additional information restricts this selection even further; and how *NLAG* reacts to the addition of both relevant and irrelevant abstractions.

We close this section by considering the importance of the generator. The current hierarchy-based generator did amazingly well. Table 7-3 demonstrates that an abnormally high number of its suggestions were right on the money and relatively few were wasted. This is unusual, even for this generator. In general, we might expect to have a knowledge base with, for example, hundreds of functions and dozens of devices. (This related to Footnote 5 on page 145.) For these reasons, I tested *NLAG* with a less smart, but still meaningful, generator. This is discussed in the next subsection.

7.2.2 Using Generator#2

The only difference between Run#1-2 and Run#1-1 is that Run#1-2 used a different function to rank the propositions. This lead to a different set of generators for the various existential variables. The ranking function embodied by the Generator#2 system made the context information highest.⁶ This system considers only symbols which are associated with the same (MRS)theory as the abstraction being considered. For example, the *rkk* abstraction is affiliated with the *LELS* lumped element linear system domain. This means that Generator#2's generators would produce only terms similarly affiliated with *LELS* and its contained theories, e.g., *wc1*. This is implemented by elevating the rank of the defined-below relation, placing it higher than any of the relations shown in Equation 7.5. (Recall that this relation was totally ignored in the Generator#1 system.)

Table 7-4 shows the results of Run#1-2, which uses the Generator#2 system. Its top part is a close match to Table 7-3, showing that Generator#2 is *functionally* virtually identical to Generator#1. SubAppendix B.4.1 shows that it produces the

⁶This was partially motivated by the common context heuristic, *HCC*.

| | rkk | kk | ckk | lkk | ALL |
|-------------------------------------|----------------------|-----|----------------------|----------------------|--------------------|
| Generated: | 195,112 ¹ | 58 | 195,112 ¹ | 195,112 ¹ | 585,394 |
| After 1-Consist: | 32 ² | 1 | 16 ³ | 16 ³ | 65 |
| After n-Consist: | 10 | 1 | 1 | 1 | 13 |
| Useful: | 5 ⁴ | 0 | 0 | 0 | 5 |
| # of Deductions | | | | | |
| <i>Find-Kernel:</i> | | | | | 33 |
| <i>Inst-Source:</i> | 14 | 18 | 10 | 7 | 50 ⁵ |
| <i>CTruep-1:</i> | 25 | 13 | 18 | 18 | 74 |
| <i>Resolve-Exists:</i> ⁶ | 554 | 181 | 468 | 469 | 1,672 |
| <i>Verify:</i> ⁶ | 284 | 75 | 17 | 17 | 393 |
| TOTALS: | 877 | 287 | 513 | 511 | 2,230 ⁷ |
| (% <i>Resolve-Exists:</i> | 63% | 63% | 91% | 92% | 75%) |

Notes:

- 1 In all three cases, this 195,112 is derived as the product of $58 * 58 * 58$.
- 2 This 32 is $1 * 4 * 8$.
- 3 In both cases, this 16 is $1 * 2 * 8$.
- 4 Entry number ##5 is the "correct" answer. It requires some $703 + 4 = 707$ steps.
- 5 Once again, this total of 50 reflects the 49 deductions shown on this row, plus 1 additional entry, run after the final abstraction (lkk) was considered.
- 6 Table B-3 derives the estimates for these two rows.
- 7 Once again, this figure includes an additional 4 steps to start the system and another 4 steps to stop it.

Table 7-4: Data for Run#1-2 (using Generator #2)

exact same answers, but in a slightly different order. The only important difference is the number of terms generated; here, there were 58 members in the LELS context. (These are shown in SubAppendix B.4.1.) This leads to a space almost 100 times larger than the one shown in Table 7-3 — 585,394 versus 6,160.

Notice, however, how close the “# of Deductions” figures are! In fact, the *Resolve-Exists* row is the only (primitive) row in this cluster which differs from its correspondent in Table 7-3. Even here, the difference is not very large, from 1,150 to 1,672. This means that *Resolve-Exists* now required about 75% of the deductions rather than 67%.

This run convinced me of the advantages of the first generator. *N.b.*, Generator#1 is used in all of the subsequent tests.

7.3 NLAG’s Sensitivity to Initial Theory—Runs#2a

This set of demonstrations uses the basic theory described in Section 7.1, with minor modifications. First, in Run#2a-1, I added various irrelevant facts: here facts about numbers, e.g., (associative +) and (mem 0 reals). There was no change in functionality (*i.e.*, the same set of answers were returned) and no change in timing or ordering. That is, as we expected, *NLAG* simply ignored this information.

As a second trivial demonstration, Run#2a-2 involved deleting some relevant source facts. Removing just

$$(\text{kirchoff2 voltage-drop}) \tag{7.6}$$

was sufficient. This meant that no source abstraction instances were derivable — *i.e.*, $Th \not\vdash S(a_1, \dots, \text{Current}, \dots, a_n)$, for all known abstractions, S , and all sets of other terms, $\{a_j\}$. As this means there are no possible common abstractions, *NLAG* returned no answer.

The final two sub-experiments involve adding some relevant target facts. This led to two experiments. Run#2a-3 involved adding some of the facts from

$$LD(Th_{CF}, (\text{rkk flow-rate pressure-drop pipe-charact pipes})).$$

In particular, I added

```
(kirchoff2 pressure-drop)
  (conserved-thru flow-rate pipes)                                (7.7)
  (ohmslaw flow-rate pressure-drop pipe-charact pipes)
```

together with their entailments, to the starting theory. (Notice that these suggest a binding for all of rkk's arguments. Subsection 7.6.1 comments on this point.) Here, establishing the correct analogy required postulating only the single conjecture, (kirchoff1 flow-rate).

Here, we found no change in functionality, but an improvement in the time required to find the "correct" answer, (rkk flow-rate pressure-drop pipe-charact pipes). Thanks to the fewest conjecture heuristic H_{FC} , NLAG is able to zero in to this instantiation much faster, finding it first, not fourth. (It required only 180 deductions, not the earlier 539.) Full details appear in Table B-4 in SubAppendix B.4.

The effects of Run#2a-4 are more subtle. Here, I added information which indirectly helped to prune the space:

```
(if (ohmslaw $tt $ct $rt $load)
    (monotonic $tt $rt decrease))

(monotonic flow-rate pipe-charact decrease)

(monotonic flow-rate cross-section increase)

(function monotonic)
```

This first rule states that the through term ($\$tt$) decreases monotonically as the resistance term ($\$rt$) increases. The second and third statements assert that flow-rate decreases monotonically as pipe-charact increases, but that flow-rate increases as cross-section increases. The fourth asserts that this monotonic relation is a function.

Once again, this new information lets NLAG focus in on the desired pipe-charact much more effectively. Here, it again finds the correct abstraction instance first, requiring only 412 steps. (SubAppendix B.4.2's Table B-4 summarizes the time required by the various submodules in finding this answer.) NLAG's functionality

on this run is slightly different, though, as the (rkk flow-rate pressure-drop cross-section pipes) instance is now eliminated, along with the other three instantiations which involve cross-section. Table B-5, also in SubAppendix B.4.2 shows the full list of answers.

The message from both Run#2a-3 and Run#2a-4 is that additional facts can be used to focus the search, by (indirectly) providing both ordering and pruning information. This means that additional knowledge eliminates search, as desired: the more one knows, in particular about the target analogue, the less is left to random factors. This results in a more directed search, and hence to superior analogies, faster.

7.4 NLAG's Sensitivity to Abstractions — Runs#2b

These three runs involve adding and removing abstractions. For Run#2b-1, I added irrelevant abstractions: *viz.*, ones which deal with historical relationships and abstract algebra relations. These are listed below (and are fully defined in SubAppendix B.4.3):

historical-event

monoid, group, abelian-group

ring, commutative-ring, ring-id, commutative-ring-id, field

There were no surprises: we found no change in functionality and no change in timing. That is, NLAG simply ignored these, as we wished. (See also Note:7-4.)

The second sub-demonstration, Run#2b-2, involved deleting the relevant abstraction, here rkk. Again the results are as expected: As NLAG did not have the needed abstraction, it returned no answer. (Here, ComAbs did consider some common abstractions, but Verify rejected them all as useless — *i.e.*, none led to a solution to the problem.)

The third experiment, Run#2b-3, involved adding other abstractions to the system, ones which could be relevant to this target problem and so might be confusing to the system as a whole. These are:

lrkk which is the “union” of the **rkk** and **lkk** abstractions: it includes information which describes both resistance (Ohm's Law) and inductance.

rckk which is the “union” of the **rkk** and **ckk** abstractions: it includes information which describes both resistance and capacitance.

lckk which is the “union” of the **lkk** and **ckk** abstractions: it includes information which describes both inductance and capacitance.

rlckk which is “union” of the **rkk**, **lkk** and **ckk** abstractions: it includes information which describes resistance, inductance and capacitance.

(These are fully defined in SubAppendix B.4.4.)

As expected, these additional abstractions gives the overall system more breadth: this run found a great many more analogies. (The first table in SubAppendix B.4.5 provides a summary of all 44 analogies *NLAG* produced.)

The other important feature is the order in which these answers are produced: in particular, **rkk** is still the first abstraction considered. In fact, the “correct” conjecture, (**rkk** flow-rate pressure-drop pipe-charact pipes), took the same number of steps now as it did in Run#1-1, 539. Hence, these additional abstractions did not lengthen the time required to find the “correct” conjecture.

This shows one benefit of the H_{MGA} heuristic: without it, *NLAG* might have first considered a more specific, and costlier, abstraction. This heuristic is especially worthwhile in situation like this, when the needed information is included in a relatively general abstraction.

Of course, this is not always the case. Imagine the desired common abstraction was a more specific one, e.g., **rlckk**. In a theory cluttered with more general abstractions — like **rlkk**, **rkk** and **kk** — *NLAG* could take longer to reach the desired answer, since it now has to walk through these more general possibilities before finding the desired one.

At first blush, one might consider the H_{MGA} heuristic to be inappropriate in these situations. I have two responses to this charge. The first is fairly minor: clever programming tricks like the *SuperCache* facility partially recaps this loss.

(This facility is especially useful in situations where the abstractions fit into a hierarchy, as they do in this case.)

The second response is more direct: H_{MGA} is not $NLAG$'s only source of advice when deciding which abstraction to consider; the H_{JK} rule also provides input. Assuming a well-worded target problem, this H_{JK} rule will select the most likely abstractions, even ones which are relatively specific. (Recall that rkk is considered before the more general kk in each of these examples: this is H_{JK} 's doing.) If $rlckk$ is the appropriate abstraction for some problem, we feel safe in similarly assuming that that problem would do its share by providing H_{JK} with the "support set" information it needs to select this $rlckk$ abstraction over the more general ones.

Table 7-5 summarizes the basic results of this run. The abstractions are listed, from left to right, in the order in which they were selected.

7.5 Ablation of Analogy — Runs#3

This is the interesting experiment. Based on the description given so far, the way to infer the new facts needed to solve the problem PT is to find a Maximally-General Common Useful Abstraction. (See Figure 4-5 in Section 4.2 and Subsection 5.4.1.)

This uses

- [Com] $\varphi(A)$ if $\varphi(B)$
- [Abs] $AbstForm(\varphi)$
- [M G] φ_1 before φ_2 if $\varphi_2 \Rightarrow \varphi_1$ ⁷
- [Use] $Th + \varphi(A) \models PT$

Were all of these characteristics essential? To find out, I subtracted out various attributes from this set. To avoid meaningless results, I only considered useful

⁷Hence, this [M G] label refers only a subset of the I_{Least} least constraining maxim. In particular, it deals only with the H_{MGA} rule. As we discussed at the end of Subsection 5.4.2, it is difficult to imagine any reason to violate the other two I_{Least} rules, H_{FC} and H_{FT} .

| | rkk | rlkk | rckk | rlckk | kk | ckk | lkk | lckk | ALL |
|------------|-------|----------------------|----------------------|-------------------------|----|-------|-------|----------------------|------------|
| Generated: | 2,048 | 262,144 ¹ | 262,144 ² | 33,554,432 ³ | 16 | 2,048 | 2,048 | 262,144 ⁴ | 34,347,024 |
| 1-Consist: | 32 | 512 ¹ | 512 ² | 8,196 ³ | 1 | 16 | 16 | 256 ⁴ | 9,541 |
| n-Consist: | 10 | 10 ⁵ | 10 ⁵ | 10 ⁵ | 1 | 1 | 1 | 1 | 44 |
| Useful: | 5 | 5 ⁵ | 5 ⁵ | 5 ⁵ | 0 | 0 | 0 | 0 | 20 |

Notes

- 1 The chore here is to instantiate (rlkk flow-rate ?2 ?3 ?4 ?5 ?6). The table below shows the number of values for each existential variable:

| Var | ?2 | ?3 | ?4 | ?5 | ?6 | Total |
|---------|----|----|----|----|----|---------|
| Initial | 16 | 16 | 8 | 16 | 8 | 262,144 |
| Final | 1 | 4 | 8 | 2 | 8 | 512 |

- 2 The chore here is to instantiate (rckk flow-rate ?2 ?3 ?4 ?5 ?6). The resulting table is identical to the one above.

- 3 The chore here is to instantiate (rlckk flow-rate ?2 ?3 ?4 ?5 ?6 ?7 ?8). The table below shows the number of values for each existential variable:

| Var | ?2 | ?3 | ?4 | ?5 | ?6 | ?7 | ?8 | Total |
|---------|----|----|----|----|----|----|----|------------|
| Initial | 16 | 16 | 8 | 16 | 8 | 16 | 8 | 33,554,432 |
| Final | 1 | 4 | 8 | 2 | 8 | 2 | 8 | 8,196 |

- 4 The chore here is to instantiate (lckk flow-rate ?2 ?3 ?4 ?5 ?6). The table below shows the number of values for each existential variable:

| Var | ?2 | ?3 | ?4 | ?5 | ?6 | Total |
|---------|----|----|----|----|----|---------|
| Initial | 16 | 16 | 8 | 16 | 8 | 262,144 |
| Final | 1 | 2 | 8 | 2 | 8 | 256 |

- 5 Each of these answers corresponds to some answer to the rkk abstraction, extended with as many meaningless extra arguments as the more specific abstraction requires.

Table 7-5: Data for Run#2b-3

systems, ones which produced an answer to the given problem. This meant I did not subtract the [Use] property. This suggests the five pairs of experiments described below.

1. - [] Useful Maximally-General Common Abstraction: Runs#1-1 and #2b-3
 $(\varphi_1 \text{ before } \varphi_2 \text{ if } \varphi_2 \Rightarrow \varphi_1; \varphi(A) \text{ if } \varphi(B); \text{AbstForm}(\varphi))$
 (These are the base cases, described in Sections 7.2 and 7.4)
2. - [M G] Useful Common Abstraction: Runs#3-2a and #3-2b
 $(\varphi(A) \text{ if } \varphi(B); \text{AbstForm}(\varphi))$ ⁸
3. - [Abs] Useful Maximally-General Common Relation: Runs#3-3a and #3-3b
 $(\varphi_1 \text{ before } \varphi_2 \text{ if } \varphi_2 \Rightarrow \varphi_1; \varphi(A) \text{ if } \varphi(B))$
4. - [Abs]-[M G] Useful Common Relation: Runs#3-4a and #3-4b
 $(\varphi(A) \text{ if } \varphi(B))$
5. - [Com]-[Abs]-[M G] Useful Relation: Runs#3-5a and #3-5b
 (CTruep alone,⁹ i.e., this system simply conjectures any leaf clause it thinks might help.)

For each of these situations, I tested to see how many deductions were required to reach the desired abstraction instance. I ran each ablated variant of *NLAG* on two data sets. The cases suffixed with an “a” used only the initial four abstractions; those suffixed with a “b”, all thirteen abstractions including all eight lumped element linear system abstractions. Table 7-6 summarizes the results in terms of number of deductions. For cases #3-1 and #3-3, when H_{MGA} was present, the results were identical. (I.e., Run#3-3a and Run#3-3b required the same amount of

⁸Here, the modified system did not insist on finding the most general abstraction — in fact, it actually sought the most specific abstraction. This corresponds to the H_{MSA} rule, defined by Heuristic 7 in Subsection 9.2.2.

⁹I actually used a slightly modified *CTruep* routine, changed to search harder for relevant rules. This required modifying MRS’s backward chaining subroutine to find and use rules of the form (If $\langle ante_1 \rangle$ (If $\langle ante_2 \rangle$ $\langle concl \rangle$)) when trying to derive some proposition which matched $\langle concl \rangle$. See Points MRS2 and MRS3 in Section 6.2.4.

| Run# | Com | Abs | M G | Deductions |
|------------|-----|-----|-----|----------------------|
| 1-1a, 2b-3 | + | + | + | 539 |
| 3-2a | + | + | - | 540 ¹ |
| 3-2b | + | + | - | 8,649 ² |
| 3-3a,b | + | - | + | 816 ³ |
| 3-4a | + | - | - | 540 ¹ |
| 3-4b | + | - | - | 8,649 ² |
| 3-5a,b | - | - | - | >25,000 ⁴ |

Notes:

- 1 Why is this number different the reading from Run#1-1, shown on the line above? This is due to a minor side effect of “turning off” the H_{MGA} rule: now certain additional overhead inferences must be taken.
- 2 These numbers reflect the number of deductions required to conjecture all four rkk-related sentences; namely, the ones shown in Equation 7.3. In both cases, this involved the r1cck abstraction, meaning that other (incorrect) conjectures were postulated as well. Had I insisted that *only* the four rkk conjectures be postulated, the wait would have been much longer: In each case, the correct rkk abstraction instance appears on the 21,506th deduction. The full set of answers for Run#3-2b appears in the second table in SubAppendix B.4.5.
- 3 Here, $NLAG$ had to consider various other relations, including conserved-thru and kirchoff1, before finding the desired rkk.
- 4 Here, Franz Lisp ran out of atom name space.

Table 7-6: Ablation of Analogy Data

time.) Notice also that the runs in the first row (which should be labeled “3-1x”) corresponds to Run#1-1 (which would be Run#3-1a) and Run#2b-3 (which would be Run#3-1b).

7.6 Analysis

This section analyzes these experiments, concentrating on the data shown in Table 7-6. At the end, it summarizes the results of all four sets of experiments.

7.6.1 Utility of Analogies

Consider first Table 7-6's bottom line. It was nice to see that our *NLAG* system was something more than just a smart residue-system: here we see that it did find the correct answer much more efficiently than the simple *CTruep* alone.

This had been a sincere worry: the initial theory includes a lot of information in the form (if (kirchoff1 \$x) (rule)); furthermore, we built *CTruep* to exploit just such information. Did we still need the analogizing process? As a strawman, why not just let *CTruep* alone solve the problem, and allow it to conjecture, willy nilly?¹⁰

While *CTruep* would eventually find the answer, its unguided search is far too slow. In each of the two runs, *CTruep* performed over 25,000 deductions when Franz Lisp ran out of atom name space. While it was definitely searching correct space (and so would have found the correct answer, eventually), it took too long to instantiate the variables of non-ground clauses.

Given such a clause (e.g., (kirchoff2 ?2)), *CTruep* could do nothing better than use it (plus derivable information, such as domain specification) as a generator, to find all derivable terms which could instantiate it. This would sprout k new subtasks, one for each proposed binding. As the derivation continued, each of these tasks could encounter yet other non-ground clauses (e.g., the ?4 of (conserved-thru flow-rate ?4)) which would each, in turn, sprout off its own k' subtasks, each corresponding to some proposed instantiation of that second clause. This soon leads to a combinatorial explosion.

As an example, imagine one of the k candidates for ?2 was PressureDrop. When the task "containing" this ?2 \mapsto pressure-drop binding encounters the conserved-thru clause, k' new tasks would be sprouted, each corresponding to some instantiation of ?4 consistent with this second clause. A similar set of k' related tasks would be generated by each other proposed instantiation of ?2, leading to $k*k'$ tasks. When encountering a third clause (e.g., (ohmslaw flow-rate, ?2, ?3, ?4)),

¹⁰*CTruep*'s first try actually failed, as the rules were too directional. It took some effort to make the rules equi-directional. (In particular, this prompted the "IfIf" facility described in Subsection 6.2.4's Point MRS3.) These additions did cause some degradation in the performance of the original system but fair's fair.

another k'' possible instantiations need to be considered, *etc.*

What we have described so far is the best possible situation, when *CTruep* happens to select the proper clauses to consider. Notice it is already combinatorial. In general, there may be other, spurious clauses mixed in. Not guided by abstractions, nothing prevents *CTruep* from considering a cost-per-hour or function clause during this derivation, which would add an additional multiplicative factor of k''' to the already huge space.

The *NLAG* system, on the other hand, used abstractions. This allowed it to consider many clauses at once, using all m statements which derive from \mathbf{S} , *i.e.*, all of $MI(\mathbf{S})$. This meant that, for each variable, *NLAG* could

1. Choose the best — *i.e.*, *most constraining* — clause to be the generator, and
2. Use the other $m - 1$ clauses to quickly prune many of those candidates.

That is, these m -tuples could work synergistically, helping each other prune the space.

This insight is consistent with several other observations. First, much of *NLAG*'s time (over 65%) was spent finding instantiations for variables — *i.e.*, was spent in the *Resolve-Exists* part of the *CTruep* process.

Second, as noted in Run#2a-3, the search is significantly faster when *NLAG* starts with additional facts which removed the need for a conjecture, and especially when that conjecture suggested a bindings for some variable.

I also ran *CTruep* after loading in the same "*LD* facts". Instantiation is not a problem in this easier case. Not only did *CTruep* find the correct answer, but it did so in only 94 steps, faster than *NLAG*'s 180 steps. (This is because *CTruep* has much less overhead, not needing to derive any instantiation in the source domain, *etc.*)

Third, all of these results continued to hold when I tested *NLAG* in other domains: *e.g.*, the two suggested by the comments in Note:7-2 and Note:7-3.

This evidence suggests that this abstraction-based approach is particularly useful (*i.e.*, better than *CTruep*) when attempting to flesh out a sparse system, one with many "existential" variables to fill in. Hence, the biggest payoff occurs when the

knowledge base designer knows the “shape of the space”, corresponding to the structures imposed by the abstractions, and in particular, when he can identify where the big “holes” of the knowledge base are. (For example, the linear system abstractions provide the basic shape of the (initially empty) hydraulics knowledge base. Its initial absence of the “*LD* facts” constitute its major “hole”.) This example shows how the appropriate abstractions helps to fill such gaps.

7.6.2 Utility of Maximal Generality

Now consider Table 7-6’s other data points. The Maximally General rule clearly made a difference; observe the effect of turning it off in Run#3-2b and Run#3-4b! When the knowledge base has a lot of specific abstractions, the *NLAG*-ish system found the desired answer over 15 times faster when running with this rule than without it. Furthermore, the performance does not degrade when the knowledge base has relatively few specific abstractions: in fact, it is virtually the same. (E.g., notice we did not need distinct “a” and “b” subcases for Run#3-1 and Run#3-3.)

This reinforces the arguments first made in Section 3.4 and repeated as conjectures in Section 4.1 and Section 5.3: The “find maximally general abstraction approach” is advantageous. This follows from the “completeness” of each abstraction, and means that it is sufficient to propose the conjectures associated with any relevant abstraction, and nothing more. (Of course, we still need to find the appropriate abstraction. Here, we rely on H_{JK} to provide the information needed to home in on the sufficiently specific abstraction — see the discussion at the end of Section 7.4.)

Why is it detrimental to chose a too specific abstraction first? The arguments appearing in Subsection 7.6.1 still apply: it is very expensive to instantiate the other, unneeded variables. We also saw that the less we know about a variable, the more expensive this instantiation process is. As we know almost nothing about these unnecessary variables, they become a considerable burden.

Two final comments: First, Section 9.2 describes this alternate approach, of seeking the maximally specific formula, as an incarnation of Section 3.4’s I_{Most} intuition. Subsection 9.2.3 provides a variety of reasons against this position. Second,

we saw that adding additional relevant abstractions did not slow the system down at all, for this particular set of experiments. In general, though, adding such abstractions could force *NLAG* to consider these other possible analogies. This slight degradation of speed is the price of the greater coverage afforded by including these additional generalizations.

7.6.3 Questionable Utility of Abstraction Label

“A rose, by any other name, would still smell as sweet...”

Romeo and Juliet, Shakespeare

Now compare the results of the systems which differ only in terms of the [Abs] attribute: *i.e.*, compare Run#3-1 with Run#3-3, Run#3-2a with Run#3-4a and even Run#3-2b with Run#3-4b. Notice that their results were about the same. This was disappointing at first, since it seems to suggest that abstractions did not make very much difference. After all, we apparently did just about as well with common *relations* as with common *abstractions*.

Seeing that arbitrary relations worked about as well as abstractions seemed to falsify Chapter 4’s central claim. Then I remembered that the actual claim: *viz.*, that abstractions were superior to *arbitrary formulae*. Of course, the final experiment was not comparing abstractions with arbitrary formulae; rather, it was comparing them with *pre-defined* relations — which the ancient scholars, and others, have defined and named earlier. This data shows that these relations do correspond to useful collections of facts, which in turn suggests that people do a pretty good job of recording only things which are useful. On further thought, this made sense: Why would anyone bother to reify a worthless formula? We tend only to have names for formulae which occur naturally; that is, for conjuncts of clauses which co-occur when solving some problems. But this is precisely Section 4.1’s defining criterion for abstractions! This leads to the punch line:

Forget about the abstraction label!

as essentially any pre-defined relation probably qualifies.

This claim is not asserting that the idea of abstractions is irrelevant, but only that this notion has been *linguistically trivialized*: almost any relation which has been named probably corresponds to a concept worth using again. Hence, it seems the class of abstractions is probably not that much smaller than the class of relations. (This was reflected in Figure 4-4, way back in Chapter 4: the space of Abstractions was intentionally drawn only slightly smaller than the space of Relations.)¹¹

7.6.4 Summary

We can summarize the analyses of this section by stating that this analogical inference process is an effective mechanism for finding useful new conjectures, and, within this system, the abstraction label is not that important (as ordinary relations work almost as well) but the maximally general rule is. This suggests that future analogy systems should seek the maximally general common relation, like the ablated version used for Run#3-3. (Such systems would differ from *NLAG* only by accepting arbitrary relations for its common formula, rather than insist on abstractions.)

Table 7-7 summarizes the findings of all of the experiments. (To complement this empirical validation, see also Note:7-1, which addresses the claim that this model of analogy does match our intuitions.)

¹¹Note:10-1 discusses for a slight refinement of the idea that all relations are abstractions, proposing a non-boolean measure for “abstraction-ness” (read “reusability”).

- *NLAG* does propose Useful Conjectures, efficiently
- Matched intuitions
 - Add irrelevant facts \mapsto no difference
 - Add irrelevant abstractions \mapsto no difference
 - Add pertinent target facts \mapsto faster
 - Add pertinent abstractions \mapsto greater coverage (but could slow down system)
 - Delete needed abstraction or source facts \mapsto no answer
- Useful maximally-general common relations sufficient?

Table 7-7: Summary of Findings

Chapter 8

Semantic Definition of Analogy

This chapter provides a semantic account of analogy. Section 8.1 motivates the utility of a semantic definition and provides the intuitions that the rest of this chapter will formalize. Section 8.2 argues that the standard Tarskian notion of semantics is inadequate and presents an alternative semantic system. Sections 8.3 and 8.4 use this framework to define the analogy relation and analogical inference process, respectively; each also demonstrates that its semantic definition is equivalent to the corresponding syntactic one presented in Chapter 2. Section 8.4 also provides semantic versions of many of the heuristics shown in Chapters 4 and 5.

8.1 Motivation and Intuitions

There are several reasons to seek a semantic (as opposed to purely syntactic) basis for analogy. First, a semantic criterion is more general: As it reflects the world itself, it is not tied to the particulars of any formalism. This means its single definition covers the processes of adding a new sentence to an incomplete theory, setting the value of a some unit's slot, or generating a new rule for a rule base. The second reason stems from the view that the analogy relationship is inherently semantic: that it pertains to objects in the world rather than descriptions of those objects. Once again, we want to describe properties of the world, not those of some

particular representation.¹ Third, these semantic considerations suggest ways of pruning and ordering the space of legal analogies. Several of these insights have been incorporated into the heuristics discussed in the previous chapters.

Of course, as one cannot put (the actual referents of) pipes and water into a computer, a semantic definition is, in and of itself, unusable. However, once these semantic \models -based relations are re-expressed within a syntactic \vdash -based framework, this formulation can be implemented and used. (Chapter 2 supplies this syntactic formulation, Chapter 6, the implementation, and Chapter 7, the validation.)

We can now repeat some of the questions posed earlier, and supply semantic, rather than syntactic, answer. First, what is an analogy? Intuitively, claiming that two concepts are analogous means that they (or actually, their referents in the world) belong to the same class of objects. Not every class of objects should qualify: in particular, finding that both objects are members of the entire universe of discourse is meaningless. Section 8.3 characterizes which classes are legal: *i.e.*, which classes can correspond to non-trivial analogies. There are also ways of ranking these eligible classes (*cf.*, Chapters 4 and 5). This ordering is especially important when forming an analogy, that is, during the process of analogical inference.

What is this analogical inference process? Chapter 2 claimed it was a type of learning, where learning means acquiring independent facts.² This syntactic process translates into the semantic process of reducing the total set of possible interpretations. An analogical inference imposes certain constraints on which interpretations should be eliminated. Loosely stated, an analogical inference further specifies the target analogue \boxed{A} .³ After an analogical inference, this object is “localized” to some set, one which is known to include the source analogue \boxed{B} .

To make this precise requires defining a particular notion of semantics, one

¹This position seems contrary to the apparent view of most AI researchers, who appear to view analogy in purely syntactic terms. This may reflect a confusion between the analogy relationship and the learning step labeled as analogical inference.

²This, of course, refers only to “*Learn*₁” processes, as defined in Note:2-2. Many of these points, especially about the inapplicability of Tarskian semantics, also apply to *Learn*₂ activities *a fortiori*.

³This chapter distinguishes symbols from their (real world) denotations by $\boxed{\text{boxing}}$ the latter. A complete synopsis of the fonts and notational conventions employed in this chapter appears at the end of SubAppendix C.1.

which can encode the partial knowledge known to the learner. Section 8.2 explains why the standard Tarskian notion of semantics is inadequate and presents a formal definition of *Partial Interpretation Semantics*.

8.2 Partial Interpretations

This section answers three questions: [1] why we are not using the standard notion of Tarskian semantics, [2] what is *partial interpretation semantics*, and [3] how this semantic framework can be used to describe a learning process. The next sections apply this systems to the specific task of analogy.

8.2.1 Why Not Use Tarskian Semantics

There are three reasons for not using the standard Tarskian semantics ([Tar52]). Note:1-2 presents the first two, arguing that this notion of truth is neither attainable nor desirable for the analogy relationship. The third reason is that Tarskian semantics, in the strictest sense, is unable to describe any learning process, and hence is an inappropriate formalism for describing analogical inference. Why? Learning is the process of embellishing an ignorant system, one which has only partial knowledge of some concepts. Unfortunately, because the Tarskian framework deals only with complete models, it cannot describe such partial knowledge. Thus, it cannot be used to describe learning in general, nor analogical inference in particular.

The semantic system described in this section can express such partial information. It stems from the same set of needs which led to the “conversational semantics” discussed in [BP83] and to the possible worlds semantics of [Kri80], [Moo80], *et al.* (Page 177 ties my idea of partial interpretations to some of these other systems.)

8.2.2 Partial Interpretation Semantics

This subsection has several parts. After presenting the basic framework, it defines the notion of the *partial extention* of a relation, and then extends this to a larger collection of labels. This leads to the notion of a partial interpretation. Second, it

discusses satisfaction within this system of partial interpretation. Third, it presents a particular way of restricting the set of allowed interpretations; this is used to form the basis of our semantics. The next subsection then discusses how this system could be used to describe learning.

Partial Interpretation Semantics is a modification of the Tarskian notion of semantics. As with the standard Tarskian framework, we are given a set of linguistic symbols, \mathcal{L} , a universe of objects \mathcal{U} , and typing information which specifies the arity of each relation. We consider functions and constants to be special kinds of relations.

A standard Tarskian interpretation (which we will call a “total interpretation”) is a function which maps each linguistic symbol onto some set of tuples of elements of the universe. In particular, the total interpretation I maps each n -ary relation symbol, R , to some set of n -tuples, $I[R] \subseteq \mathcal{U}^n$. For example, I maps the constant symbol c onto the singleton set consisting of a single 1-tuple, $I[c] = \{ \langle \boxed{c} \rangle \}$.

Partial Extensions: Unlike total interpretations, a partial interpretation might not completely specify the extensions⁴ of its symbols. We define $\text{+}R^I$ to be the (positive) extension of the relation symbol R under the interpretation I . $\text{-}R^I$ is the complement of that set: this is the set of n -tuples which are known not to belong to the R relation. When I is total, these sets are both complementary and disjoint:

$$\text{+}R^I \cap \text{-}R^I = \{ \} \quad (8.1)$$

$$\text{+}R^I \cup \text{-}R^I = \mathcal{U}^n \quad (8.2)$$

A partial interpretation may violate Equation 8.2, the second constraint above. This means that a given partial interpretation, \mathcal{P} , may not have complete knowledge of some of the relations. We define $\text{-}R^{\mathcal{P}}$ as the unknown members of this relationship:

$$\text{-}R^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{U}^n - \text{+}R^{\mathcal{P}} - \text{-}R^{\mathcal{P}} \quad (8.3)$$

Both extreme points are relevant: $\text{-}R^{\mathcal{P}} = \mathcal{U}^n$ means that \mathcal{P} knows nothing at all about R . The other extreme, $\text{-}R^{\mathcal{P}} = \{ \}$, means that \mathcal{P} has *totally specified* this relation R .

⁴This chapter uses *extension* with a “s” when referring to the result of enlarging (extending) a set, and *extension* with a “t” when discussing the real world object corresponding to a symbol. See [Hin62].

This is why Tarskian interpretations are called “total interpretations”: Saying that the interpretation I is total means that every relation is totally specified; i.e., $R^I = \{\}$ for every relation symbol R .

For notation, the term “interpretation” refers to any partial interpretation. (The phrase “total interpretation” refers to only Tarskian interpretations.) We define the (partial) extension of R (under the interpretation \mathcal{P}), written $R^{\mathcal{P}}$, as the pair of known positive and negative instances of the relation R , $R^{\mathcal{P}} = \langle \text{+}R^{\mathcal{P}} \text{ -}R^{\mathcal{P}} \rangle$. When the interpretation is obvious from context, or irrelevant, we omit the \mathcal{P} superscript and write simply R .

As desired, this partial interpretation model of semantics allows us to specify the learner’s ignorance. For example, in the same manner that $\langle \text{Abe} \rangle \in \text{+Dogs}^{\mathcal{P}}$ means that Abe is a dog and $\langle \text{Beth} \rangle \in \text{-Dogs}^{\mathcal{P}}$ indicates that Beth is not a dog, we can use $\langle \text{Shep} \rangle \in \text{Dogs}^{\mathcal{P}}$ to indicate that we simply do not know whether Shep is a dog — i.e., $\langle \text{Shep} \rangle \notin \text{+Dogs}^{\mathcal{P}}$ and $\langle \text{Shep} \rangle \notin \text{-Dogs}^{\mathcal{P}}$.

Using more relevant examples, we may know that $\langle \text{R} \text{ + } \text{O} \rangle$ is in $\text{+Group}^{\mathcal{P}}$, but have $\langle \text{R}_0 \text{ * } \text{1} \rangle$ only in $\text{-Group}^{\mathcal{P}}$; or that the RKK relation includes

$\langle \text{Current} \text{ VoltageDrop} \text{ Resistance} \text{ Resistors} \rangle$

as a positive instance and

$\langle \text{FlowRate} \text{ PressureDrop} \text{ Cost} \text{ Pipes} \rangle$

as a negative instance, and does not know about

$\langle \text{FlowRate} \text{ PressureDrop} \text{ PipeCharacter} \text{ Pipes} \rangle$.

That is,

$\langle \text{Current} \text{ VoltageDrop} \text{ Resistance} \text{ Resistors} \rangle \in \text{+RKK}^{\mathcal{P}}$
 $\langle \text{FlowRate} \text{ PressureDrop} \text{ Cost} \text{ Pipes} \rangle \in \text{-RKK}^{\mathcal{P}}$
 $\langle \text{FlowRate} \text{ PressureDrop} \text{ PipeCharacter} \text{ Pipes} \rangle \in \text{RKK}^{\mathcal{P}}$.

Constructible Labels: While this subsection has discussed partial extensions of relations alone, this same notation can be used for a larger class of syntactic labels.

This class of labels contains all relation symbols, and is extended by (recursively) combining existing labels using the operations of intersection, union, complement, cross-product and projection. The equation below defines the partial extension of each such constructible label:

$$\begin{aligned}
 \underline{C_1 \cap C_2} &= \langle \text{+}C_1 \cap \text{+}C_2 & \text{-}C_1 \cup \text{-}C_2 & \rangle \\
 \underline{C_1 \cup C_2} &= \langle \text{+}C_1 \cup \text{+}C_2 & \text{-}C_1 \cap \text{-}C_2 & \rangle \\
 \underline{\neg C_1} &= \langle \text{-}C_1 & \text{+}C_1 & \rangle \\
 \underline{C_1 \times C_2} &= \langle \text{+}C_1 \times \text{+}C_2 & \text{-}C_1 \times \mathcal{U}^n \cup \mathcal{U}^m \times \text{-}C_2 & \rangle \\
 \underline{C_1|_i} &= \langle \text{+}C_1|_i & \text{-}C_1|_i & \rangle
 \end{aligned} \tag{8.4}$$

When necessary, we use $CL_{\mathcal{L}}(C)$ to indicate that C labels a constructible term. (Hence, the set $CL_{\mathcal{L}}$ is the smallest class which includes the relation symbols in the language \mathcal{L} and which is closed under intersection, union, complement, cross-product and projection.) This uses \cap and \cup to represent the standard set operators of intersection and union. The binary operator \times takes two sets and returns their cross-product. If we assume that $\text{+}C_1$ is a set of m -tuples and $\text{+}C_2$ is a set of n -tuples, then each member of $\text{+}C_1 \times \text{+}C_2$ is an $m + n$ -tuple, formed by joining some m -tuple from $\text{+}C_1$ with an n -tuple from $\text{+}C_2$.

The projection operator, $|_i$, is defined by

Definition 22 $R|_i = \{ \boxed{c} \mid \exists \boxed{d_{j \neq i}}. \langle \boxed{d_1} \cdots \boxed{c} \cdots \boxed{d_n} \rangle \in R \}$

(This corresponds to the related syntactic operator shown in Equation 2.7 on page 15.) In all cases, \mathcal{C} is defined as the set of remaining elements; à la Equation 8.3.

We can define other connectives using these in the obvious manner; e.g., $\underline{C_1 \Rightarrow C_2}$ corresponds to $\underline{\neg C_1 \cup C_2}$.

Every constructible set is labeled by the description used to generate it. For example, another label for the set of all fathers is “ $\text{Parent} \cap (\text{Male} \times \mathcal{U})$ ”. This means that $\underline{\text{Parent} \cap (\text{Male} \times \mathcal{U})}$ encodes the pair

$$\langle \text{+Parent} \cap (\text{+Male} \times \mathcal{U}^1) \quad \text{-Parent} \cup (\text{-Male} \times \mathcal{U}^1) \rangle.$$

(This uses $\underline{U} = \langle U^1 \ \{\} \rangle$ where U^1 is the set of 1-tuples taken from the universe U .)

We refer to the positive extension of each constructible label as a “constructible set” or a “bucket”.

Satisfaction: In standard Tarskian semantics, the notation

$$\models_I \sigma \tag{8.5}$$

means that the interpretation I satisfies the sentence σ . This same notation can be used for partial interpretations as well. As the base case, consider only atomic formulae, i.e., relationships. The syntactic statement $\exists x R(x)$ means that there is an object in the universe which is a known positive instance of the R relation. Hence, $\models_I \exists x R(x)$ means there is some $\boxed{d} \in \mathcal{R}^I$, which means that $\|\mathcal{R}^I\| > 0$.

Universal quantification is also with respect to the known positive instances of the relation; $\forall x R(x)$ means that \mathcal{R}^I includes every object in the universe: $\models_I \forall x R(x)$ is true when $\mathcal{R}^I = U^1$. This means that R 's allowable domain contains all possible objects. (This sounds like a semantic analogue to the syntactic *Trivial* relation, and inspires Definition 26.)

So much for the base case, of quantified variables in simple relations. The inductive step uses the translation which maps the composition processes of set union, set intersection and set complement into formula disjunction, conjunction and negation, respectively. If we use cross-product and projection appropriately, we can find a constructible label C which corresponds to any formula φ_C . (E.g., the formula $\text{Parent}(x, y) \ \& \ (\text{Male}(x) \ \& \ y = y)$ corresponds to the label $\text{Parent} \cap (\text{Male} \times U)$.) As above, $\models_I \exists x \varphi_C(x)$ means that $\|\mathcal{C}^I\| > 0$ and $\models_I \forall x \varphi_C(x)$ means that $\mathcal{C}^I = U$. A simple induction proof shows that this extends to formulae with more than one variable.

The only remaining task is to define the truth conditions for instantiating a formula with a constant. If we think of a constant as a unary relation, then $\phi(d)$ translates to $\forall x d(x) \Rightarrow \phi(x)$. (We also know that $\exists x d(x)$, but that is a separate consideration.) This suggests that

$$\models_I \phi(d) \quad \text{holds when} \quad \underline{d}^I \sqsubseteq \underline{\phi}^I. \tag{8.6}$$

This uses the extension-subset relation \sqsubseteq , defined by

Definition 23 $\underline{R}^I \sqsubseteq \underline{S}^I \iff \text{+}R^I \subseteq \text{+}S^I \ \& \ \text{-}R^I \supseteq \text{-}S^I$.

Unlike the Tarskian interpretation, satisfaction using these partial interpretations is not complete; there can be sentences σ for which neither $\models_I \sigma$ nor $\models_I \neg\sigma$ hold. For example, using the interpretation shown above, $\not\models_p \text{Dogs}(\text{Fido})$ and $\not\models_p \neg\text{Dogs}(\text{Fido})$.

Allowed Interpretations: We can easily generate a gigantic space of possible interpretations: each n -ary label C can induce some $3^{|\mathcal{U}|^n}$ different interpretations, based on whether each n -tuple in \mathcal{U}^n is assigned to $\text{+}C$, $\text{-}C$ or ? . There are usually some constraints on which interpretations should be allowed. For example, knowing that **Fido** is a constant, we should insist that +Fido^I contain at most one entry. (That is, we reject any interpretation I in which $\|\text{+Fido}^I\| > 1$.) We may also want to connect different relations to one another: for example, by insisting that all dogs are animals, i.e., $\underline{\text{Dogs}}^I \sqsubseteq \underline{\text{Animals}}^I$. As a third situation, we may know precisely the extension of **Fido**, and want to consider only those interpretations which map **Fido** onto this particular real world object, $\boxed{\text{Fido}}$. (Technically, this means I must satisfy $\underline{\text{Fido}}^I = \langle\langle \boxed{\text{Fido}} \rangle\rangle$.)

Our goal now is to specify the particular collection of partial interpretations which satisfy these types of restrictions. We achieve this by defining the set as those interpretations which satisfy three types of requirements. First, the interpretation I must match a particular domain and range specification: viz., I must map from the given language ($\text{Dom}[I] = \mathcal{L}$), onto the given universe of discourse ($|I| = \mathcal{U}$), using the given typing information. We use the notation $\text{Eligible}(I)$ to mean that I satisfies these constraint. (The particular fixed language, universe and type information is implicit in the *Eligible* definition.)

Secondly, I must satisfy a particular theory. To deal with the first two restrictions mentioned above, the theory can include constraints like

$$\exists x [\text{Fido}(x) \ \& \ \forall y \text{Fido}(y) \Rightarrow y = x] \tag{8.7}$$

$$\forall x \text{Dogs}(x) \Rightarrow \text{Animals}(x) \tag{8.8}$$

This satisfaction requirement, even when coupled with the *Eligible* constraint, is still not enough to precisely specify the extension of a symbol: i.e., to handle the third situation. This means we need yet a third type of specification. We can achieve this objective by considering only those interpretations which agree with a certain base interpretation. That is, viewing each partial interpretation as a mapping, we only consider those interpretations which consistently *extend* the assignments of that base partial interpretation.

We say that one interpretation *extends* another, written $I \overset{E}{\supseteq} J$, if the interpretation I consistently *extends* what J knows about each relation:

$$I \overset{E}{\supseteq} J \iff \forall R \in \mathcal{L} \left[\begin{array}{l} \text{+}R^I \supseteq \text{+}R^J \quad \& \\ \text{-}R^I \supseteq \text{-}R^J \end{array} \right] \quad (8.9)$$

We can think of an interpretation as “expressing an opinion” about the extension of the symbol R whenever it includes an object in either $\text{+}R$ or $\text{-}R$. In this framework, $I_1 \overset{E}{\supseteq} I_2$ means that I_1 agrees with all of I_2 's opinions. Of course, I_1 may have other opinions as well.

This is the third component needed to constrain the allowed interpretations. We use the term $Allowed(J, Th)$ to specify the set of allowed interpretations. Each $I \in Allowed(J, Th)$ must be an eligible interpretation, must model the theory Th and must extend J 's assignments. That is,

Definition 24 $Allowed(J, Th) = \{ I \mid Eligible(I) \ \& \models_I Th \ \& \ I \overset{E}{\supseteq} J \}$

This *Allowed* relation is the cornerstone of our notion of semantics. At any instant, we consider only those legal interpretations which model a given theory and extend a particular privileged partial interpretation, \mathcal{RW} .⁵ That is, rather than allow arbitrary interpretations of the symbols, this system allows only those which belong to $Allowed(\mathcal{RW}, Th)$. We refer to Th as the *underlying theory* and \mathcal{RW} as the *base interpretation*.

This means that certain symbols are “grounded”: that is, have a particular fixed interpretation. This follows from Equation 8.9's monotonicity conditions, which

⁵I think of \mathcal{RW} as encoding one's knowledge of the Real World.

guarantee that every n -tuple in $\mathcal{R}^{\mathcal{RW}}$ will remain in \mathcal{R}^I in every allowed I ; this holds for each n -tuple in $\mathcal{R}^{\mathcal{RW}}$ as well. In many cases, a constant symbol c will have a unique interpretation in \mathcal{RW} — that is, $\|\underline{c}^{\mathcal{RW}}\| = 1$ and $\|\mathcal{C}^{\mathcal{RW}}\| = 0$. In such cases, \underline{c} refers to this same real world object in every allowed interpretation. (By convention, we can refer to this symbol as \boxed{c} .)

We conclude this subsection with some quick observations. The next subsection uses this partial interpretation model to describe the phenomenon of learning.

PI1. Both Semantic and Syntactic Constraints: This *Allowed* relation requires both a semantic \mathcal{RW} and a syntactic Th . Why do we need both? We already discussed why Th alone is inadequate. On the other hand, the base interpretation \mathcal{RW} does seem sufficient to describe the current situation. However, the next subsection shows that the intentional relationships defined by Th 's sentences are essential when describing a dynamic learning process.

PI2. Need for Partial Interpretations: Notice that everything stated above (and below) still holds if we just modify the *Allowed* set to include just those interpretation which are total: that is, we could instead use the *Allowed'* relationship, where

$$Allowed'(\mathcal{RW}, Th) = \{ I \in Allowed(\mathcal{RW}, Th) \mid \forall R \in \mathcal{L} \mathcal{R}^I = \{\}. \} \quad (8.10)$$

Of course, the base interpretation \mathcal{RW} is still partial. As we have to deal with partial interpretations anyway, I choose to use *Allowed* over this alternative *Allowed'* version.

PI3. Density of Allowed: We require that this *Allowed* set of interpretations exhibit a certain density property: Given any $\boxed{c} \in \mathcal{R}^{\mathcal{RW}}$, $Allowed(\mathcal{RW}, Th)$ must include a pair of interpretations I_1 and I_2 such that $\boxed{c} \in \mathcal{R}^{I_1}$ and $\boxed{c} \in \mathcal{R}^{I_2}$. (This \boxed{c} can be an n -tuple of constants.) *N.b.*, this does not mean that semantic assignments are totally arbitrary: the $\boxed{\text{Fido}} \in \mathcal{Dogs} \Rightarrow \boxed{\text{Fido}} \in \mathcal{Animals}$ example shows that an assignment to one symbol (\mathcal{Dogs}) can force some assignment to another symbol ($\mathcal{Animals}$).

This means that any set of eligible interpretations uniquely determines its base interpretation: it is precisely the pointwise intersection of those interpretations. That is, given the set S of eligible interpretations and theory Th , such that $\models_I Th$ holds for each $I \in S$, we can define the base interpretation, \mathcal{B} , of this S set as

$$\mathcal{B} = \bigcap_{I \in S} I \quad (8.11)$$

To confirm that \mathcal{B} , observe that $S = Allowed(\mathcal{B}, Th)$. The next point defines what it means to intersect two or more interpretations.

PI4. Intersection of Interpretations: Equation 8.11 is based on intersecting interpretations. Intuitively, the intersection of the interpretations I and J is the *most complete* interpretation which can be extended to both I and J . That is, $\mathcal{K} = I \cap J$ if \mathcal{K} consists of exactly the opinions shared by I and J ; i.e.,

$$\mathcal{K} = I \cap J \text{ implies } I \overset{\text{E}}{\supseteq} \mathcal{K} \text{ and } J \overset{\text{E}}{\supseteq} \mathcal{K}; \quad (8.12)$$

and if, for any other \mathcal{K}' which satisfies Equation 8.12's right-hand side, $\mathcal{K}' \overset{\text{E}}{\supset} \mathcal{K}$.

We can realize this by defining \mathcal{K} 's extension of each symbol as the (pointwise) intersection of I 's and J 's values for that symbol:

$$\mathcal{K} = I \cap J \iff \forall R \in \mathcal{L} \left[\begin{array}{l} \text{+}R^I \cap \text{+}R^J = \text{+}R^K \quad \& \\ \text{-}R^I \cap \text{-}R^J = \text{-}R^K \end{array} \right] \quad (8.13)$$

As this operator is obviously associative and commutative, its n -ary form \bigcap_i is well defined.

8.2.3 Using Partial Interpretations to Describe Learning

At any instant, this semantic system considers only some of the eligible interpretations: *viz.*, the members of the set $Allowed(\mathcal{RW}, Th)$. This set of allowed interpretations may change over time. In particular, the more the system learns, the fewer interpretations remain. If *Before* is the initial collection of allowed interpretations, and *After*, the set of interpretations which remain after the learning step, we claim that *After* is a proper subset of *Before*, i.e., $After \subset Before$.

For our current purposes, we consider only a particular type of learning: *viz.*, the process which “fills in” some unknown entries — e.g., by adding $\langle \boxed{\text{Shep}} \rangle$ to Dogs , or $\langle \boxed{\text{FlowRate}} \boxed{\text{PressureDrop}} \boxed{\text{PipeCharacter}} \boxed{\text{Pipes}} \rangle$ to RKK . In particular, if we consider the process of learning that $\boxed{\text{Shep}}$ is a *Dogs* (sic), we see that

$$\text{After} = \{I \in \text{Before} \mid \langle \boxed{\text{Shep}} \rangle \in \text{Dogs}^I\} \quad (8.14)$$

We were able to use the *Allowed* relation to specify *Before*, as $\text{Before} = \text{Allowed}(\mathcal{RW}, Th)$. This subsection shows that we can find a similar definition for *After*, defining it as the set of interpretations allowed from some more refined base interpretation \mathcal{RW}' : i.e., $\text{After} = \text{Allowed}(\mathcal{RW}', Th)$. Hence, this type of learning reduces to the process of extending the initial base interpretation to a more complete \mathcal{RW}' . This \mathcal{RW}' partial interpretation is the *minimum extension* of \mathcal{RW} in which some tuple of objects (e.g., $\langle \boxed{\text{Shep}} \rangle$) is in some labeled “bucket” of tuples (e.g., *Dogs*).

What can we say about this \mathcal{RW}' ? Notice we can *not* define \mathcal{RW}' as

$$\mathcal{RW}'[x] = \begin{cases} \langle \text{Dogs}^{\mathcal{RW}} \cup \{ \langle \boxed{\text{Shep}} \rangle \} & \text{Dogs}^{\mathcal{RW}} \rangle & \text{when } x = \text{Dogs} \\ \mathcal{RW}[x] & \text{otherwise} \end{cases} \quad (8.15)$$

for all symbols $x \in \mathcal{L}$. Here, knowing that dogs are animals, an interpretation I should be declared illegal unless $\text{Dogs}^I \sqsubseteq \text{Animals}^I$. This means $\langle \boxed{\text{Shep}} \rangle$ should now be in $\text{Animals}^{\mathcal{RW}'}$, contrary to Equation 8.15’s simplistic claim. (This is because \mathcal{RW} did not know that $\boxed{\text{Shep}}$ was an animal: i.e., nothing prevents us from assuming that $\langle \boxed{\text{Shep}} \rangle \in \text{Animals}^{\mathcal{RW}}$.) This shows that \mathcal{RW} ’s extensional information is not enough. We need the additional, intentional information that relates dogs to animals.

The syntactic *Th* portion of $\text{Allowed}(\mathcal{RW}, Th)$ provides this connection. In particular, we can utilize *Th*’s intentional content by considering the full set of *Allowed* interpretations, since this, by construction, “embodies” *Th*. Using Equations 8.11 and 8.14, we can define \mathcal{RW}' to be the intersection of the subset of currently allowed interpretations in which *Shep* is known to be a dog:

$$\mathcal{RW}' = \text{MinExt}(\text{Allowed}(\mathcal{RW}, Th), \langle \boxed{\text{Shep}} \rangle, \text{Dogs}), \quad (8.16)$$

where

Definition 25 $MinExt(S, \boxed{A}, c) = \bigcap \{I \in S \mid \boxed{A} \in {}_+c^I\}.$

As implied by this usage, *MinExt*'s first argument is a set of interpretations, its second argument is a particular n -tuple of objects, and its third is a label which denotes a "bucket" into which to deposit this n -tuple.

Definition 25's intersection ranges only over allowed interpretations. This means that whenever interpretation I is in this set, if $\langle \boxed{Shep} \rangle$ is in ${}_+Dogs^I$, we know that $\langle \boxed{Shep} \rangle$ is in ${}_+Animals^I$ as well. This guarantees that $\langle \boxed{Shep} \rangle$ is in ${}_+Animals^{\mathcal{R}\mathcal{W}'}$, as desired. This solves the problem mentioned above. We also know that this embellished $\mathcal{R}\mathcal{W}'$ will be an eligible partial interpretation, guaranteeing that the range of this new interpretation is the same universe \mathcal{U} used above (i.e., the world is not generating new objects).

Commentary: This ends our definition of *Partial Interpretation Semantics* in general. Before using this system to discuss analogies, it is worth digressing to discuss some relevant issues.

First, why should we use this particular notion of semantics? While Subsection 8.2.1 already argued that Tarskian semantics is inappropriate, it did not specifically promote this particular framework. I chose this semantic system because it describes a fairly typical situation, one which occurs whenever people converse. Namely, both speaker and hearer usually know a great deal in common. In particular, they will agree on both the referents of many of the terms used in the discourse, and on some of the extensions of some of the relations. This partial interpretation semantics framework captures this situation very well.

Secondly, we have discussed only one type of learning, one implemented by this single means of embellishing the base interpretation by using the *MinExt* relation.⁶ We might also consider other ways of modifying the *Allowed* set. For example, we may want to manipulate classes of object rather than just single elements. This

⁶This system of partial interpretations can be used to formally describe various aspects of learning. For example, Note:8-1 uses it to define the claim that some sentence is novel with respect to a particular concept.

While this definition is worded in terms of partial interpretations, it holds for the Tarskian sense of semantics as well. (One need only replace each occurrence of “ $\mathcal{C}^{\mathcal{R}\mathcal{W}}$ ” with the usual Tarskian extension.) This partial interpretation form is used to be consistent with the rest of this chapter, and in particular, with Section 8.4’s semantic definition of analogical inference.

Analogy_{sem} is equivalent to *Analogy_F*:

This *Analogy_{sem}* is, by construction, equivalent to *Analogy_F*, modulo the obvious identification of *Analogy_F*’s initial theory *Th* to *Analogy_{sem}*’s underlying theory *Th* and base interpretation $\mathcal{R}\mathcal{W}$. The actual proof begins by mapping each relation symbol *R* into the corresponding extension, $\underline{R}^{\mathcal{R}\mathcal{W}}$. As each constant symbol *c* is assumed fixed, we can identify \underline{c} with the single real world object, \boxed{c} . This means that $\langle \boxed{a_1} \cdots \boxed{A} \cdots \boxed{a_n} \rangle \in \mathcal{C}^{\mathcal{R}\mathcal{W}}$ whenever $Th \models \varphi_c(a_1, \dots, A, \dots, a_n)$ and $\langle \boxed{b_1} \cdots \boxed{B} \cdots \boxed{b_n} \rangle \in \mathcal{C}^{\mathcal{R}\mathcal{W}}$ whenever $Th \models \varphi_c(b_1, \dots, B, \dots, b_n)$, using the formula φ_c which corresponds to the set labeled *C*. As $\neg Trivial_{sem}(C|_i, \mathcal{R}\mathcal{W})$ is equivalent to $\neg Trivial(\varphi_c|_i, Th)$, we are done.

8.4 Semantic Definition of Analogical Inference

This section reconsiders the syntactic view of analogical inference presented in Section 2.1 within Section 8.2’s semantic framework. Here, the syntactic process of adding an independent fact corresponds to the process of reducing the set of possible interpretations.⁸ This involves resetting the base interpretation, $\mathcal{R}\mathcal{W}$, to a more refined $\mathcal{R}\mathcal{W}'$, and thenceafter only considering comparable extensions of this $\mathcal{R}\mathcal{W}'$ interpretation.

An analogical inference induces a certain type of restriction on the possible interpretations. Namely, the $\boxed{A} \sim \boxed{B}$ analogical inference selects a certain subset of the possible interpretations. As with the syntactic situation, the driving intuition is that an analogical inference completes an analogy (see Section 2.3).

This section formalizes this idea. For pedagogical reasons, it begins with a

⁸This is also true with the standard possible worlds semantics. It follows immediately from Gödel’s Completeness Proof; see [End72].

simplified version, which is later embellished to describe the full nature of analogical inference.

Simple Analogical Inference: Each $\boxed{A} \sim \boxed{B}$ analogical inference reduces the space of possible interpretations. In particular, the interpretation I remains only if $\boxed{A} \in \text{+}C^I$, where this class C is a constructible set that is known to include \boxed{B} and initially does not include \boxed{A} — i.e., $\boxed{B} \in \text{+}C^{\mathcal{RW}}$ and $\boxed{A} \in \text{?}C^{\mathcal{RW}}$.⁹ Looking at it the other way, this inference eliminates all interpretations, J , in which either $\boxed{A} \in \text{?}C^J$ or $\boxed{A} \in \text{-}C^J$. This means we are defining a new base interpretation, \mathcal{RW}' , as the intersection of the remaining I s. Using Definition 25,

$$\mathcal{RW}' = \text{MinExt}(\text{Allowed}(\mathcal{RW}, Th), \boxed{A}, C). \tag{8.18}$$

The object \boxed{A} becomes more “localized” with respect to this more constraining \mathcal{RW}' : it now has to participate in the known extension of C , i.e., $\boxed{A} \in \text{+}C^{\mathcal{RW}'}$.

After the inference, \mathcal{RW}' becomes the new base interpretation. This means that the set of allowed interpretations is reduced; it now includes only those interpretations in $\text{Allowed}(\mathcal{RW}', Th)$. (Since $\mathcal{RW}' \stackrel{E}{\supset} \mathcal{RW}$, we know that $\text{Allowed}(\mathcal{RW}', Th) \subset \text{Allowed}(\mathcal{RW}, Th)$.) Hence, subsequent learning steps are with respect to this interpretation, and further reduce the set of allowed interpretations associated with this base interpretation \mathcal{RW}' .

Notice that changing the base interpretation has no effect on the extension of any totally specified relation. Because we assumed that all constants are totally specified, this means that constants are unaffected: both $\underline{A}^{\mathcal{RW}'}$ and $\underline{A}^{\mathcal{RW}}$ refer to the same object. However, C 's extension has changed: in particular, $\text{+}C^{\mathcal{RW}'} \supset \text{+}C^{\mathcal{RW}}$.

General Analogical Inference: This above description corresponds to the simple form of analogical inference, whose syntactic version was shown in Equation 2.1. The general process is slightly more complex. We need to consider sets of n -tuples rather than only sets of singletons. This means we must deal with some set C which satisfies the requirements shown in Figure 8-1, using some pair of sets $\{\boxed{a_j}\}$ and $\{\boxed{b_j}\}$ taken of objects in the universe, \mathcal{U} .

⁹Of course, the $\boxed{A} \sim \boxed{B}$ analogical statement may lead to several different analogical inferences, each corresponding to a different common set (labeled C). We consider only one of these.

$$\begin{array}{l}
 \text{Allowed}(\mathcal{RW}, Th), \quad \boxed{A} \sim \boxed{B} \approx \langle \boxed{a_1} \cdots \boxed{A} \cdots \boxed{a_n} \rangle \in {}_+ \mathcal{C} \\
 \text{where Common:} \quad \langle \boxed{b_1} \cdots \boxed{B} \cdots \boxed{b_n} \rangle \in {}_+ \mathcal{C}^{\mathcal{RW}} \\
 \text{Independent:} \quad \langle \boxed{a_1} \cdots \boxed{A} \cdots \boxed{a_n} \rangle \in \not\mathcal{C}^{\mathcal{RW}} \\
 \text{NonTrivial:} \quad \neg \text{Trivial}_{\text{Sem}}(\mathcal{C}|_i, \mathcal{RW})
 \end{array}$$

Figure 8-1: Semantic Definition of (General) Analogical Inference: \approx

To read the notation, the \approx process generates a new set of allowed interpretations, $\text{Allowed}(\mathcal{RW}', Th)$, based on a new base interpretation, \mathcal{RW}' .¹⁰ We require that \mathcal{RW}' extends the earlier base interpretation \mathcal{RW} and satisfies the right-hand side of the \approx form: i.e., $\langle \boxed{a_1} \cdots \boxed{A} \cdots \boxed{a_n} \rangle \in {}_+ \mathcal{C}^{\mathcal{RW}'}$. As we saw earlier, this means

$$\mathcal{RW}' = \text{MinExt}(\text{Allowed}(\mathcal{RW}, Th), \langle \boxed{a_1} \cdots \boxed{A} \cdots \boxed{a_n} \rangle, \mathcal{C}). \quad (8.19)$$

We conclude this section with some final comments.

Sem1. Comparison with \vdash :

It is easy to see that Figure 8-1's semantic definition corresponds to Figure 2-1's syntactic \vdash . The two criteria with the same name (**Common** and **NonTrivial**) match directly and \approx 's **Independent** condition matches both \vdash 's **Unknown** and **Consistent**.

Sem2. Extension to correspond to \vdash_{PT} :

Figure 8-1 corresponds to *general* analogical inference. To extend it to correspond to useful analogical inference (whose syntactic form is seen in Figure 3-1), we need only add the semantic analogue to the **Useful** criterion. As before, an analogy is deemed useful if, in the resultant situation (be it a new theory Th' or a new

¹⁰This corresponds to the claim that $Th, A \sim B \vdash \varphi(A)$ means that a new theory, Th' , is generated, where this Th' extends the earlier theory Th and contains the sentence on the right-hand side, $\varphi(A)$.

base interpretation, $\mathcal{R}\mathcal{W}'$), the target problem PT has some solution. Here, this means that

$$+_PT^{\mathcal{R}\mathcal{W}'} \neq \{\}. \quad (8.20)$$

All free variables lexically within this PT are assumed existentially bound for this query. See also the discussion on page 170.

(We can assume that $+_PT^{\mathcal{R}\mathcal{W}} = \{\}$ with the original interpretation; otherwise, there was no need to acquire this new information.)

Sem3. Connection to $Analogy_{Sem}$:

As suggested above, this analogical inference usually establishes an $Analogy_{Sem}$ connecting the analogues,

$$Analogy_{Sem}(\boxed{A}, \boxed{B}, \mathcal{R}\mathcal{W}', \langle C \langle \boxed{a_1} \dots \boxed{A} \dots \boxed{a_n} \rangle \langle \boxed{b_1} \dots \boxed{B} \dots \boxed{b_n} \rangle \rangle) \quad (8.21)$$

(Notice this uses the new base interpretation, $\mathcal{R}\mathcal{W}'$.)

Unfortunately, this is not quite guaranteed. We saw above that the \approx inference changes what we know about C : i.e., $\underline{C}^{\mathcal{R}\mathcal{W}'}$ is different from $\underline{C}^{\mathcal{R}\mathcal{W}}$. In particular, it is possible that $+_C^{\mathcal{R}\mathcal{W}'} = \mathcal{U}$, which would mean that $Trivial_{Sem}(C|_i, \mathcal{R}\mathcal{W}')$. This parallels Point TR5 in Section 2.4.

Once again, we could sidestep this problem by defining a stronger version of non-triviality:

$$\text{Definition 28 } StronglyNonTrivial_{Sem}(\phi, I) \iff _ \phi^I \neq \{\}$$

Here, Equation 8.9's monotonicity conditions assure us that ϕ stays *StronglyNon- $Trivial_{Sem}$* as we embellish this interpretation.

Sem4. Semantics of $Analogy_{CA}$ and \Vdash_{PT} :

Chapter 4 refines the syntactic notion of analogical inference by allowing only certain formulae (*viz.*, abstraction formulae) to be eligible as analogy formulae. We can trivially express this additional requirement semantically, by allowing only certain sets to be eligible for the common set status. That is, the syntactic

constraint that only abstraction formulae qualify translates into the semantic requirement that only certain buckets qualify: *viz.*, \Vdash_{PT} considers only those buckets which correspond to (the extension of) some relation, and furthermore, only of those relations which have been tagged as abstractions.

This means the abstraction-based analogical inference process deposits an n -tuples of objects into some bucket, where this bucket corresponds to some abstraction.

(While we can state this H_{Abst} rule semantically, we have no *semantic justification* for why it works. See Note:4-4.)

Sem5. Further Refinements of \Vdash_{PT} , expressed Semantically:

Parts of Chapter 5 already used this semantic account as a basis by which to rank legal analogies. In particular, Section 5.3 discussed how to find the least constraining inference. This I_{Least} maxim prefers the analogical inference whose set of remaining interpretations is the largest. To state this formally: we may be considering several analogical inferences. Let S_i be the set of interpretations which remain after the i^{th} possible analogical inference. Of these, I_{Least} advocates the j^{th} inference, where $\|S_j\| = \max_i \{\|S_i\|\}$.¹¹ (It is difficult to express semantically the particular syntactic heuristics which operationalize this maxim. In particular, the interactions between H_{MGA} and H'_{FC} might lead to a violation of this maxim, and there seems no semantic way of expressing the inherently syntactic H_{FT} rule.)

The other abstraction-based rules, H_{JK} and H_{CC} , are also virtually impossible to express semantically, as each specifically refers to some lexical terms. *e.g.*, the way H_{JK} lexically generates the source problem PS from the target problem PT . We could handle this H_{CC} rule by using the collection of `SameTheory n` buckets: *e.g.*, by insisting that $RKK \sqsubseteq \underline{\text{SameTheory4}}$, where `SameTheory4` represents the set of all 4-tuples which belong to the same theory. This is quite cumbersome.

¹¹To see that there is an upper bound, notice that the `Unknown` condition guarantees that $S_i \subset \text{Allowed}(RW, Th)$ for all i .

Chapter 9

Standard Definitions of Analogies

“There is no word which is used more loosely or in a greater variety of senses, than Analogy.”

A System of Logic, J. S. Mill (1900)

This dissertation has presented a particular definition of analogical inference, one which differs from many standard views of analogy in both major and minor ways. This chapter discusses several of these differences and explains my position.

The first part of this chapter provides a series of specific analyses. Section 9.1 provides a quick preface, summarizing the major issues. The next two sections focus on the two general categories in which my approach differ from others. Section 9.2 emphasizes how the *NLAG* analogical inference process ($\|_{PT}$) differs from many others. Section 9.3 is concerned with how *NLAG*'s underlying definition of analogy, *Analogy_F*, differs from the traditional view.

These contrasts may seem vague in the abstract. Section 9.4 makes these differences more concrete by describing other related research in the field of artificial intelligence, covering both general learning and analogy *per se*. This survey highlights the key distinctions between these systems and mine.

Several other parts of this dissertation have provided brief literature surveys and comparisons. In particular, Note:2-2 contrasts various types of learning programs, Note:2-5 mentions two minor ways in which my view of the analogy relation differs from the standard view, Section 3.1 discusses many other research projects

which examine how to “learn for a purpose”, Section 4.5 compares my model of abstractions with various related notions, and Section 8.2.3 compares my “Partial Interpretation Semantics” with other flavors of possible worlds semantics.

9.1 Common Views of Analogy

Different people have very different notions of what an analogy is. This preliminary section characterizes the general issues and outlines the basic differences between the traditional view and mine. The summary at the end serves as an outline into the next two sections.

The relevant questions are (i) what is an analogy and (ii) how can an analogy be found and then used in learning. The accepted view holds that an analogy is a symbol-to-symbol mapping and that the learning by analogy process uses this mapping to map existing source sentences into new target sentences; *cf.* [Hes66], [Eva68], [Kli71], [NM73], [DM76], [DM77], [MU77], [McD79], [Bro79], [Win80], [Gen80b], [Cle81], [Car81a], [Len82a], [Hob83a,Hob83b], [Bur83b], [Hof84], ...¹ My view is basically compatible with these positions. (This may be more apparent using the *Analogy_T* framework described Note:2-4. We can use the realization that these two formulations are equivalent to include other analogy systems — those based on a common relation — within this fold; *cf.*, [Ari52], [Pol54], [Thi77], [Bla62] [Pai79], [Gen80a], [Gen83] and [Mar84].)

The next relevant question is how to guide the search for good analogies. Repeating Figure 2-4, this requires finding first the relevant source sentences and then the appropriate symbol-symbol mapping. Here we start to see some differences. Most systems begin by establishing that some source terms correspond to analogous target terms. This suggests a sentence-to-sentence mapping, pairing together source and target sentences which contain these (respective) symbols. These other systems then follow a bottom-up, spreading-activation procedure in the source domain to find other connected source facts. These derived source sentences are mapped over

¹Some of this research describes the related linguistic phenomenon of metaphor. The distinction between metaphor and analogy is not relevant to this dissertation in general, nor to this discussion in particular.

to propose new sentences in the target domain. These new sentence-to-sentence correspondences may suggest a new set of analogous terms; and the process continues.

While accepting the utility of the above process, *NLAG* is based on a slightly different approach. I assume that a wealth of relevant connections have already been “cached” into what I call “abstractions”. This lets *NLAG* follow a top-down strategy: after finding an abstraction which holds in the source domain, it tries to instantiate this same abstraction in the target domain. This leads to the view that most useful analogies are common abstractions.

This is the major difference between *NLAG*'s \Vdash_{PT} process and most other systems. It, in turn, induces many of the other differences. These distinctions, as well as several others, are summarized in the list below. The parenthetical note following each point states my particular slant on that issue.

- Useful Analogy
(The result of an analogical inference should be a collection of useful new conjectures. This is opposed to the view that analogies must be “maximal”.)
- Exactness of Analogy
(It is possible to state, statically, what an analogy is. This is opposed to the procedural view, which holds that analogies are inherently “slippery” and inexact.)
- Explicit Model of User
(Different people understand different analogies. This translates into an awareness that the contents of the underlying theory is important.)
- Explicit Description of Analogical Connection
(Certain properties vary from analogue to analogue while others remain invariant. This information should be explicit.)
- Semantic Relation
(Analogy deals with real world objects, rather than their representations. Hence the form of that underlying theory is not important.)

The first two points deal mainly with the analogical inference process and are discussed in Section 9.2. There we see how they each closely tie in with *NLAG*'s model-based approach. Section 9.3 addresses the latter three points, which are concerned with the basic definition of an analogy. That is, they pertain more to the analogy relation than to the analogical inference process.

9.2 How $\|\sim_{PT}$ Differs from Other Versions of Analogical Inference

Many analogy seeking systems consider only “novel” analogies: *i.e.*, analogies which must be “deduced” on the fly. The *NLAG* system, on the other hand, deals with less novel situations. In particular, it assumes that the commonality between the analogues is already known.² Its objective is to utilize this general information in “fleshing out” useful additional facts about the target analogue.

This section compares *NLAG*'s flavor of analogical inference with many of the others. Subsection 9.2.1 discusses the prevalent bottom-up view of analogy and shows how this leads to view that equates an analogy with “maximum commonality”. It also ties this view to each of the first two issues of the previous section. Subsection 9.2.2 follows this intuitive definition of maximum commonality with a more rigorous one. Subsection 9.2.3 describes limitations inherent in the “maximum commonality” approach.

9.2.1 View that Analogies must be Constructed

Much of the current research in analogy is based on the premise that an analogy must be assembled as needed: that is, each analogy is constructed piecemeal. This, in turn, suggests the bottom-up approach to analogical inference sketched above: First establish some kernel of correspondence between the analogues. New facts

²The common abstraction encodes this commonality, alternatively called “ground” ([Pai79], [Ric36]) or “common” [Bla62]. Also, Point Lit10 in Section 9.4 returns to this issue of how much is known initially.

are then “analogically inferred” by extending this kernel in all possible ways. This suggests that an analogy is inherently the union of all such *commonable features*, where a feature is *commonable* if it is shared initially by both analogues or if it can be shared by both analogues by adding additional facts. The first category, of initially common features, corresponds to the starting kernel, and the full set of features (including also the features which can be made common), to the full analogy. Since this process seeks every possible commonality, I refer to it as the “maximal commonality view of analogy”.

We still have to define what a “common feature” is: i.e., what it means to claim that “a feature is shared by the two analogues”. In its full generality, this refers to some formula which can be instantiated by either analogue. (This is consistent with the definitions appearing throughout this dissertation.) Many analogy systems impose other restrictions on what qualifies as a commonable feature: for example, some consider only formulae which can be reached from some initially common formulae using only “causality” links. The precise definition of a commonable feature is not that important here. Instead, the critical observations are that these systems work bottom-up, starting from all initial common features, and they attempt to generate the full set of all commonable features. This means that an analogy is not considered complete until all of these commonable features have been found.

My view of analogy, as embodied in the *NLAG* subroutines, shares some similarities with this “maximal commonality” position. In particular, my approach also involves first finding and then embellishing an initial kernel of correspondence. The important distinction is that *NLAG* does *not* seek all commonable features connecting the two analogues. Instead, it seeks only the minimal connection, under the additional constraint that this connection be useful. (Section 5.3 discussed this sense of minimality and Chapter 3 defined usefulness.) This leads to my model-based orientation: *NLAG* only searches for certain specific clusters of connections — those corresponding to an abstraction — and stops as soon as it has found such a set.

This distinction — between my top-down *NLAG* system and the traditional bottom-up approach — ties in with the first two questions posed on page 186:

9.2. HOW $\|\sim_{PT}$ DIFFERS FROM OTHER VERSIONS OF ANALOGICAL INFERENCE

First, this distinction arises from the different purposes of our respective systems. While most analogy systems attempt to find the greatest similarity connecting the analogues, *NLAG*'s objective is to find an analogy which suggests additional new facts which can be used in solving some problem.

Second, this maximum commonality approach invites the view that analogy is a *slippery* and inexact phenomenon.³ Almost all researchers appear to agree with this position. For example, some computational linguists view metaphor interpretation as a process of successively lifting constraints. (See [Rus76] and [Lev77].) This view is implicit in other systems which suggest that an analogy must be the result of continuing modifications, that it must "grow" dynamically or change continuously (*cf.*, [Eva68], [Hes66], [Win80], [Hob83a], *etc.*). It also underlies the view that an analogy is inherently "non-decomposable", *cf.*, [Bla62], [Boy79] and [Sea79].

How is this related to the maximum commonality maxim? This maxim asserts that an analogy gets better as more common features are found. Unfortunately, there are always more commonable features. Furthermore, there are many different sets of additional postulates, and these additions may be incompatible with one another. They consider an analogy "slippery" because there are always additional, incompatible ways in which the analogues can be placed in correspondence. (Subsection 9.2.2 expresses this formally, noting both that one can generate many diverse common formulae and that a given formula may have different (and incompatible) instantiations in the target domain.)

"Inexactness" is related to slipperiness. Any process is inexact unless there is an effective *a priori* way of deciding both which of the distinct branches to take and when to stop seeking additional information. This means most analogical inference processes are inexact. (Note:9-2 describes some other independent factors which reinforce the contention that the process of understanding an analogy is inexact.)

NLAG addresses both of these issues by using abstractions. By describing what type of new information is required, they make *NLAG* a (more nearly) exact and less slippery process. (Going back on further level, we can attribute this solution to *NLAG*'s goal of finding just *useful* conjectures, as this led to its use of abstractions.)

³Hofstadter coined the particular phrase, "slippery" [Hof84].

The rest of this section defines more precisely the notion of “maximal commonality” and presents arguments which first support and then refute it.

9.2.2 Definition of Maximal Commonality

As suggested above, many analogy systems seek *the maximally common connection* between the two analogues. This maxim, first expressed as the I_{Most} intuition in Section 3.4, holds that

Intuition 3 *Better analogies express more in common between the analogues.*

This subsection provides a semantic criterion for *commonality* and tells how it can be measured. (Note:9-3 proposes various lexical measures of commonality and shows that all are inadequate.) We can use this measure to define the analogy which expresses the maximal commonality of the analogues. The next subsection argues why we may not want to use this maximally common analogy.

Consider a simple example. Assume you know nothing about Duke, and are told that $Duke \sim Fido$. Knowing that $Fido \in Dogs$ and $Dogs \subset Animals$, we might analogically infer either $Duke \in Dogs$ or $Duke \in Animals$.

Which of those two says more about Duke? Clearly the first. It leads to many more conjecture about Duke — e.g., that he has a cold nose, probably enjoys digging holes, *etc.*, in addition to all the things you know about animals in general. Thinking semantically, there are fewer extensions of that first common formula $\varphi_D(x) \Leftrightarrow x \in Dogs$ than the second $\varphi_A(x) \Leftrightarrow x \in Animals$: that is, $\{x \mid x \in Dogs\} \subset \{x \mid x \in Animals\}$.

Figure 9-1 shows this graphically. Claiming that Duke is a dog “specializes” him to be in the inner Dogs box, which means that everything known to be true about dogs in general holds for Duke. On the other hand, knowing only that Duke is an animal allows him to roam anywhere within the larger Animals box, meaning that only those facts which are true for all animals are guaranteed to hold for Duke. This means there must be fewer new things which we can conjecture about Duke in this latter case.

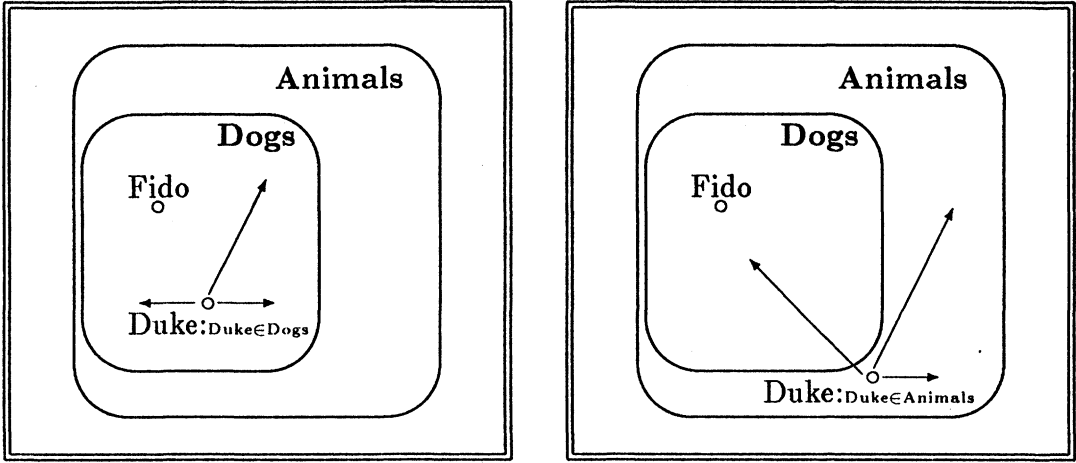


Figure 9-1: Why $Duke \in Dogs$ is more specific than $Duke \in Animals$.

This motivates the claim that *fewer extensions is better*, which suggests seeking the analogy which minimizes the number of extensions. The argument goes: the fewer the number of extensions of the common formula, the more constraining that relation is. That means it is more special to see that both analogues qualify — it is, *a priori*, less likely. (This resonates with Shannon’s concept of information theory: that information is inversely related to the number of values [Gal78].) This view holds that better analogies are based on more specific formulae.

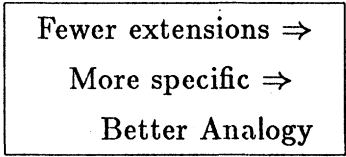


Figure 9-2: Commonality Intuition

Definition 29 states this formally, defining what it means to claim that φ_1 is *more specific* than φ_2 , written $\varphi_1 \overset{s}{>}_{[Th]} \varphi_2$:

Definition 29 $\varphi_1 \overset{s}{>}_{[Th]} \varphi_2 \iff \|\{x \mid Th \models \varphi_1(x)\}\| < \|\{x \mid Th \models \varphi_2(x)\}\|$

This describes only unary formulae. What about n -ary formulae, for $n > 1$? One option is to consider the total number of extensions of all arguments, *i.e.*, all $\langle d \text{ op } id \rangle$ triples which satisfy Group. This still favors Group over Monoid, and either of these (and almost anything else) over $\tau(x, y) \Leftrightarrow (x = x \ \& \ y = y)$. It,

however, has the problem that irrelevant arguments can interfere. For example, we could define

$$\text{SGroup}(d, op, id, x) \stackrel{\text{defn}}{\iff} \text{Group}(d, op, id) \ \& \ (x = x). \tag{9.1}$$

Notice that the number of $\langle d \ op \ id \ x \rangle$ quadruples that satisfy this SGroup relation far exceeds the number of $\langle d \ op \ id \rangle$ triples that satisfy Group. This says nothing, however, about the specificity associated with the second argument: that is, + and * express as much commonality by participating in the SGroup relation as they do by participating in the Group relation. The fact that every possible concept could qualify for SGroup’s final position is irrelevant.

This suggests we “existentially quantify away” the other $n - 1$ arguments and define specificity in terms of the domain of the particular argument of the formula. Now we can compare two analogical formulae φ_1 and φ_2 , which have the analogues in the i^{th} and j^{th} position, respectively.

Definition 30 $\varphi_1|_i \stackrel{s}{>}_{[Th]} \varphi_2|_j \iff \|\{x \mid Th \models \varphi_1|_i(x)\}\| < \|\{x \mid Th \models \varphi_2|_j(x)\}\|$

If we use standard Tarskian semantics, these sets are usually infinite.⁴ We can, in any case, easily catch logical implication — $\forall x \varphi_1(\dots x, \dots) \Rightarrow \varphi_2(\dots x, \dots)$ — and use this measure to rank the analogies. This means that I_{Most} would prefer Group over Monoid, and the through term, “t”, in $\mathbf{RKK}(t, c, r, 1)$ over $\mathbf{KK}(t, c)$. (As these examples suggest, abstractions often fit into such an implication-based hierarchy. This would allow us to make good use of this measure; see Note:9-1.)

This leads to the “use most specific formula” heuristic, H_{MSA} :

⁴However, the model of semantics presented in Chapter 8 leads to sets which can be finite in size. Here, this “<” comparison is meaningful. Notice also that Definition 30 falls out of our semantic considerations, as $\mathcal{W}[\text{SGroup}|_2(*)] = \mathcal{W}[\text{Group}|_2(*)]$.

9.2. HOW \sim_{PT} DIFFERS FROM OTHER VERSIONS OF ANALOGICAL INFERENCE

Heuristic 7 H_{MSA} : If $Th, A \sim B \vdash \varphi^1(a_1^1, \dots, A, \dots, a_n^1)$
 and $Th, A \sim B \vdash \varphi^2(a_1^2, \dots, A, \dots, a_m^2)$
 (involving the analogues in the i^{th} and j^{th} positions, resp.)
 and $\varphi^1|_i \stackrel{s}{\succ}_{[Th]} \varphi^2|_j$,
 Then prefer φ^1 over φ^2 .

To help understand this H_{MSA} rule, consider the extreme formula

$$\phi(x) \Leftrightarrow [x = \text{FlowRate} \vee x = \text{Current}]. \quad (9.2)$$

This is clearly the most specific common formula which can join FlowRate and Current, since it has only these two extensions.⁵

As another source of examples, recall that we can generate a new analogy from any pair of analogies, using the fact that the conjunction of any pair of analogical inferences is also a legal analogical inference (see Section 2.5):

$$\begin{array}{l} Th, A \sim B \vdash \varphi_1(A) \\ Th, A \sim B \vdash \varphi_2(A) \\ \hline Th, A \sim B \vdash \varphi_1(A) \& \varphi_2(A) \end{array} \quad (9.3)$$

Now consider the resulting $\psi(x) \Leftrightarrow \varphi_1(x) \& \varphi_2(x)$. Notice that H_{MSA} prefers this conjunct over its components: $\psi \stackrel{s}{\geq}_{[Th]} \varphi_1$ and $\psi \stackrel{s}{\geq}_{[Th]} \varphi_2$.

We can generalize this: assume that each formula in $\{\sigma_i\}_{i=1}^m$ qualifies as an analogy formula joining A and B. This means that the conjunction of any possible subset of this set is also a viable analogical formulae.⁶ Since this specificity measure prefers any m -tuple of conjuncts to any of its $m-1$ -tuple children, we see that H_{MSA} prefers the "largest" formula, $\varphi_{All} = \bigwedge_i \sigma_i$.

⁵The claim that this is actually maximal does depend on the expressiveness of the language. If the language is completely expressible, one could define a yet more specific formula, one satisfied only by a single object. Note:9-4 explains why this does not imply that A and B are co-referential.

⁶This description is overly simplistic as certain instantiations of these σ_i may be incompatible; see Point NotMS3 in Subsection 9.2.3. Also, this does not exhaust the set of all possible analogies; see Section 2.5 and Note:2-7.

The points in the next subsection critique this approach, arguing against using this φ_{All} formula. We close this subsection with two comments.

First, this H_{MSA} rule, as described above, is only a way of comparing analogy formulae: e.g., it tells us to prefer **Group** over **Monoid**. If we view it as an instance of the I_{Most} maxim (see Section 5.3), it can be extended to determine how to instantiate a formula with free variables. (E.g., it could suggest which term should be used to instantiate the $?_1$ in **Monoid**($?_1, +, 0$.)

Second, this commonality measure sheds light on the *Triviality* condition: Figure 3-1's **NonTrivial** constraint eliminates the minimal elements of this commonality space, allowing us to discard those formulae whose argument could accept 100% of the possible values. The motivation is that such formulae express no commonality and so would be worthless as an analogy.

(To state this semantically: we claim that a formula ϕ is trivial whenever its extension includes the full universe of possible values: i.e., $\phi = \mathcal{U}$. This is reflected in Definition 27.)

9.2.3 Limitations of the H_{MSA} Rule

Many other analogy system seek the maximally specific formula that joins the two analogues, based on the implicit assumption that this is both desirable and sufficient. This subsection presents five arguments against this H_{MSA} position. The three main ones are: [1] it does not lead to natural (or coherent or useful) analogies, [2] it is not efficient and [3] it may still miss the correct answer. Two less important points are: [4] it ignores the *a priori* knowledge (and so violates the least constraint principle) and [5] it leads to formulae with limited applicability. (Note:9-5 presents yet another reason, describing situations where the user might not like the analogies suggested by this rule.) After presenting these arguments in order, this subsection concludes by discussing the trade-off between H_{MGA} and H_{MSA} .

NotMS1. H_{MSA} Leads to Unnatural Analogies:

This point addresses the specific claim that H_{MSA} alone is a sufficient criterion on which to determine desirable analogies. (By contrast, the remaining points

9.2. HOW $\|\sim_{PT}$ DIFFERS FROM OTHER VERSIONS OF ANALOGICAL INFERENCE

discuss limitations of H_{MSA} which hold even when when this rule is coupled with H_{Abst} : i.e., even when only considering common abstractions.)

Consider first the ϕ formula defined in Equation 9.2. While it is maximally specific, it is neither more natural or more useful than the other formulae we have mentioned, in particular, Figure 2-3's φ_{RKK} . We could, perhaps, legislate away these problematic disjunctions. (See Note:2-3.) However, similar problems can occur even if we consider only conjunctive formulae. Consider, for example, the uninteresting formula

$$\begin{aligned} \theta(x, y, z, \dots) \leftrightarrow & \varphi_{RKK}(x, \dots) \ \& \\ & \text{CostPerHour}(x, y) \ \& \\ & \text{DiscoveredBy}(x, z) \ \& \\ & \dots \end{aligned} \tag{9.4}$$

This could be an analogy formula linking **FlowRate** to **Current**. While this strained θ formula expresses more in common than its component φ_{RKK} (i.e., it is more specific), it is certainly less intuitive.

As this example suggests, we can define (almost) arbitrary collections of facts, and get more and more information into successive formulae. While this certainly leads to *better similarities*, are they leading to more *usable* analogies? In particular, is φ_{All} , the aforementioned conjunction of the σ_i , the best possible analogy formula?

I argue the answer is "no"; that certain partial subsets are more meaningful than the full set. In particular, those which represent coherent clusters (and which give rise to a useful corpus of new facts) should be considered better than those which represent little beyond a chance co-occurrence. (Recall the θ shown above.) This means that the desired analogy formula need not be the maximal points in this space of possible analogy formulae.

The upshot of this point is that, while H_{MSA} does push toward greater similarity connecting the analogues, it is not pushing towards more coherent or more useful analogies. This is why this specificity criterion, by itself, is inadequate. In turn, this explains why *NLAG* is based on finding common abstractions, rather

than maximally specific common formulae. (The points below provide other arguments, which oppose even seeking the maximally specific common abstraction.)

NotMS2. H_{MSA} Can be Very Inefficient:

Point NotMS1 above argued that we do not necessarily need the most specific formula. However, it did not state any reason why we should not seek it anyway. This point and the next argue that this is a bad idea; each provides a reason for preferring some non-maximally-specific formula.

Consider again the set of possible analogy formulae, $\{\sigma_i\}_{i=1}^m$. If every σ_i is unary, this analogical inference requires only conjecturing some of the $\sigma_i(A)$ clauses. (I.e., we need only conjecture the sentence $\sigma_i(A)$ if it is independent of the starting theory).

In general, though, some formulae are non-unary; each σ_i is really a formula of n_i parameters. To simplify notation, assume that each is a formula with N arguments. This means we are really considering the analogy formula

$$\varphi_{All}(x_1, \dots, x_N) \equiv \bigwedge_i \sigma_i(x_1, \dots, x_N). \quad (9.5)$$

Finding the target instantiation of φ_{All} requires finding bindings for this N -tuple of variables. As we saw from Section 7.6, this can be quite expensive.

Now imagine that solving the problem requires only some subset of these σ_i s, say, $\{\sigma_i\}_{i=1}^p$. If we use φ_{All} , we have to deal with the other $m - p$ formulae which are not used in solving the problem. These needless clauses can prove quite expensive: First, simply conjecturing these auxiliary formulae is not free. The real cost, however, is in instantiating the unneeded variables which these unused formulae contribute.

As an example, recall that the **RKK** abstraction is sufficient to solve the "Find the flowrate" problem. If we erroneously used the **RLKK** abstraction, we would need to find instantiations for the two variables associated with the **Induct-Law** clause, `$indt` and `$ind-load`, in addition to the four variables used by **RKK**-related clauses. Of course, we never use the instantiation of these

additional variables. This means that the additional work of instantiating those unused variables is simply wasteful.

To generalize: one reason not to use the maximally specific formula φ_{All} is that, in general, only a subset of its component clauses are used to solve any given problem. Adding the additional conjectures is both unnecessary and expensive. The major additional cost is in instantiating the additional free variables, the ones associated only with clauses not used. This theoretical argument was empirically verified in Section 7.6, where we saw how enormous this additional expense can be.

NotMS3. H_{MSA} Is Not Guaranteed:

The point above discussed how expensive it can be to find a maximally specific analogy. This cost might be acceptable if the resulting analogy was guaranteed to be correct or at least as encompassing as possible. Unfortunately, this is not the case.

Consider, once again, the analogy formula $\varphi_{All}(x_1, \dots, x_N)$ formed as the conjunction of all of the $\{\sigma_i\}_{i=1}^m$. Now realize that there may be many distinct target instantiations for this formula, each corresponding to a legal analogical inference. This means that even finding a maximal element might not be correct: the desired answer might follow from a different instantiation. (I.e., this specificity measure is really just a partial ordering, which may have more than one maximum.)

In fact, this desired instantiation might force a less specific formula: Imagine that $\sigma_1(A, 7)$ is the only new fact needed to solve the given problem, and that $\sigma_2(A, 7)$ is provably false.⁷ This prevents us from analogically inferring $\varphi_{All}(A, 7)$. Hence, the desired analogy cannot use the maximal formula, φ_{All} , but must settle for a subset of it.

NotMS4. H_{MSA} Ignores Initial Knowledge:

Another objection to this H_{MSA} idea is that this measure often runs counter to the least constraint proposal discussed in Section 5.3, which is based on

⁷Of course, this σ_2 must still be a legal analogical formula joining A and B; but that connection may be "witnessed" by a different instantiation, e.g., *Th*, $A \sim B \vdash \sigma_2(A, 12)$.

the Section 3.4's I_{Least} intuition. The maximal specificity principle favors the most specific abstraction instance, even if it means postulating many additional conjectures. For example, H_{MSA} prefers conjecturing $\mathbf{Group}(\mathfrak{R}, *, 1)$ over $\mathbf{Monoid}(\mathfrak{R}, *, 1)$, even if $\mathbf{Invertible}(*, \mathfrak{R})$ is not derivable from Th . The "least constraint" maxim would suggest $\mathbf{Monoid}(\mathfrak{R}, *, 1)$ instead, as it is closer to the current world: that is, as it requires that fewer conjectures be made. (Note:9-5 further motivates this position by discussing additional situations where H_{MSA} 's choice may not be found appropriate. This also ties in with Point NotMS5 and is elaborated in the summary at the end of this subsection.)

NotMS5. More Clauses leads to Less Generality:

Each conjunct included in the analogy formula limits the applicability of that formula, since it means that fewer terms can qualify. As a trivial example, the terms which satisfy $\sigma_1(x) \& \sigma_2(x)$ is a subset of the ones which satisfy $\sigma_1(x)$ alone. Because more specific formulae hold less often, they have less potential usefulness. (Recall that the extreme formula, ϕ from Equation 9.2, applied only to this particular pair of analogues.) We prefer analogy formulae which pertain to a host of different pairs of analogues.

The *maximize applicability* maxim encourages re-usable analogy formulae: i.e., formulae like abstraction formulae which pertain to many diverse situations. This maxim favors using as few conjuncts as possible, since this leads to the formula with the greatest range of applicability. The summary below elaborates this point.

We can summarize these objections by discussing the tension between maximizing commonality and applicability, as suggested by various points above. We already discussed the positive features of these two end-points; there are objections to each position as well. The maximally specific point usually represents an arbitrary hodge-podge of facts, whose instantiation is needlessly expensive, i.e., is

| | | |
|---------------|-------|--|
| | < + > | |
| More Specific | ⇒ | more new facts can be conjectured |
| More General | ⇒ | more concepts can apply |
| | < - > | |
| More Specific | ⇒ | “too rich” more <i>irrelevant</i> facts must be conjectured |
| More General | ⇒ | “too impoverished” less information can be transferred from one analogue to the other |

Table 9-1: Tension between Commonality and Applicability

overkill for any problem. On the other hand, the maximally general formula is useless for a different reason: it says absolutely nothing.⁸ While we do not advocate the extreme point in either direction, we still feel the opposing pressures, respectively pulling us towards analogy formulae which approach each of these extrema. Table 9-1 summarizes these points, showing the major positive and negative factors for each position.

It should be apparent that this dichotomy is the (by now familiar) struggle between the I_{Most} and I_{Least} intuitions. The I_{Most} intuition (embodied in the H_{MSA} maximal specificity rule) examines only the conclusion of an analogical inference, all but ignoring the starting state of knowledge, Th . It basically tries to maximize the *a posteriori* commonality, maximizing the similarity between the two analogues *after* the analogical inference. The least constraint principle I_{Least} , on the other hand, uses the initial knowledge, to tell $NLAG$ when stop seeking yet other new facts. When combined with I_{Close} , it stops the search as soon as the batch of new conjectures, together with the initial facts, form the minimal, acceptable collection. In effect, I_{Least} deals with how common the two analogues are *a priori*, before adding

⁸It is, in fact, *Trivial* in the sense defined in Section 2.4. This means it would not even qualify as a legal \sim analogical inference.

any analogically conjectured facts. (Subsection 5.3.1 elaborates this argument.)

As an example, compare the possible abstraction formulae **Group** and **Monoid** (for the same argument), in an empty target theory. I_{Least} prefers **Monoid**, since it is more likely, *a priori*, to be true than **Group**. On the other hand, I_{Most} prefers **Group** because, once conjectured, the derived **Group** statement says more about that argument, and so seems more likely to be useful.

Each of these intuitions has factors which encourage, and discourage, its use. While this specificity measure might still be sufficient if our only goal was to find the maximal similarity joining the two analogues, our goal is different: the underlying purpose of the analogy is to (efficiently) conjecture the new facts needed to solve some problem. This is why we are seeking a common formula which leads to a useful analogy, as opposed to a maximally specific one.

Following Section 3.4's suggestion, we resolve the I_{Least} versus I_{Most} tension by using I_{Close} . Chapter 4 shows that this translates into finding a common abstraction. That is, it is sufficient to find that each analogue satisfies the same abstraction; *NLAG* then stops seeking additional facts.

9.3 How $Analogy_F$ Differs from Other Versions of Analogy

Most systems view an analogy as a symbol-symbol mapping, where every occurrence of a source symbol is mapped into a corresponding target symbol. This section describes the main ways in which my $Analogy_F$ approach extends the standard version.⁹ This preface first sketches the final three points of page 186 in order; it then summarizes these points and tells where the more elaborate arguments can be found.

The first two points deal with explicitness, first with respect to the model of the user and then with respect to the analogy itself.

An analogy is inherently a subjective phenomenon; in a given situation, different

⁹This section considers only the general $Analogy_F$ relationship; Chapter 4 has already motivated why this is specialized to the $Analogy_{CA}$ relation.

learners may “understand” the same analogy differently. I attribute this to the learner’s initial collection of facts: i.e., different learners reach different conclusions because they know different things initially. (Fraser provides some empirical data which corroborates this obvious claim [Fra79].) In many analogy systems, this theory is left implicit. We prefer to deal with the initial theory explicitly, having it available for inspection and evaluation. This is why it appears in both our notion of analogical inference, \vdash , and in our definition of the analogy relation. In all cases it is encoded as the *Th* argument.

Having stated that an analogy depends on this theory, we now need to specify how. This leads to the second point: what precisely is an analogical connection? To represent it explicitly, we have to specify both which sentences are in the domain of the sentence-to-sentence mapping and then how these source sentences are transformed into the desired target domain sentences. First, what are the relevant source sentences? Point NotStd1 in Note:2-5 demonstrated that an analogy may only deal with a subset of the facts known about the analogues.¹⁰

How is this source information mapped over to form target sentences? Subsection 9.3.1 discusses the importance of the *form* of these source facts, showing that different formulations of the same facts can lead to different analogies. This means that just describing the semantic content of these source sentences is not sufficient. (The “semantic relationship” point justifies our claim that all of these possible analogies should be found.)

There is another way to consider this second sub-issue: it forces us to state precisely what is common between the analogues and what changes. That is, the analogy must specify which “source” constants are invariant and which are mapped to new target terms. (Of course, the *Analogy_F* formalism makes this a non-issue: the φ common formula encodes precisely what remains invariant, and the different sets of arguments, what changes.)

¹⁰Subsection 9.2.3 extends that argument, explaining why the domain of the analogy should not even be the most specific analogy formula. Rewording this into *Analogy_T*’s terminology: consider all possible source sentences which do not lead to a contradiction. While the union (i.e., conjunction) of all of these is still a legal source domain for the analogy mapping, that subsection demonstrates that this may not be the best possible useful analogy.

The fact that we are considering syntactic manipulations does not mean that we view analogy as a syntactic phenomenon. On the contrary, we consider an analogy to be a semantic relationship, one which deals with a pair of objects “in the world” rather than with their representation. (This was first stated in Section 8.1.) This attitude justifies our freedom to reformulate the analogues as necessary: to consider implicit new sentences (see Point NotStd2 in Note:2-5) and even generate implicit new terms (see Subsection 9.3.2) when formulating the analogical correspondence.

The list below summarizes these basic differences and tells where these issues are discussed.

- **Explicit Model of Learner**

An analogy should vary from individual to individual, based on what the learner already knows.

- Explicate initial theory
(Section 2.3)

- **Explicit Description of Analogical Connection**

An analogy should explicate what features are invariant between the analogues and what changes.

- Explicate domain of sentence-to-sentence mapping
(Point NotStd1 in Note:2-5)
- Explicate what is common between the analogues and what changes.
(Subsection 9.3.1)

- **Semantic Relation**

An analogy is a semantic relationship, which deals with a pair of objects “in the world”, rather than with their representation.

- *Prima facie* facts are not enough
(Point NotStd2 in Note:2-5 and Subsection 9.2.3)
- Initial vocabulary is not enough
(Subsection 9.3.2)

9.3.1 Exact Formulation is Important

Essentially all analogy systems operate by transferring some information which pertains to the source analogue over to the target analogue. Some information is preserved over this mapping and some is altered from one analogue to the other. Unfortunately, many analogy formalisms do not explicate what is invariant and what is local to the analogues.

This subsection explains why this distinction is important by showing that the same source facts can lead to different target facts, depending on what information is invariant. After illustrating this point with an example, it closes with a statement of the general problem, showing that the form of the analogy formulae matters.

Imagine we want to understand how a dam's output is controlled and are told to consider dams as transistors.¹¹ Assume we know that $\text{CValve}(\text{Transistor}, \text{Current}, \text{Current})$, which means that a transistor is a controllable valve which regulates the flow of electrons (i.e., of *Current*), using energy supplied in the form of *Current*. This suggests that we consider the analogy formula,

$$\varphi_{\text{CV}}(d, t) \Leftrightarrow \text{CValve}(d, t, t). \quad (9.6)$$

As we know that a dam regulates the flow of water (i.e., of *FlowRate*), this seems perfect. The target instantiation is simply $\varphi_{\text{CV}}(\text{Dam}, \text{FlowRate})$, which is equivalent to $\text{CValve}(\text{Dam}, \text{FlowRate}, \text{FlowRate})$.

On the other hand, we might instead have used a slightly different formulation of the same facts, one based on the analogy formula,

$$\varphi_{\text{CBC}}(d, t) \Leftrightarrow \text{ControlledByCurrent}(d, t) \quad (9.7)$$

where

Definition 31 $\forall d, t. \text{ControlledByCurrent}(d, t) \iff \text{CValve}(d, t, \text{Current})$.

Here the resultant analogical inference would be $\text{ControlledByCurrent}(\text{Dams}, \text{FlowRate})$ which is equivalent to $\text{CValve}(\text{Dams}, \text{FlowRate}, \text{Current})$.

¹¹A possibly more natural example of this phenomenon appears in Note:9-6. We use this *Transistors~Dams* example here as it deals with *FlowRate* and *Current*.

Here, even though $\varphi_{CV}(\text{Transistor}, \text{Current})$ is equivalent to $\varphi_{CBC}(\text{Transistor}, \text{Current}, \text{Current})$, $\varphi_{CV}(\text{Dam}, \text{FlowRate})$ is not equivalent to $\varphi_{CBC}(\text{Dam}, \text{FlowRate}, \text{FlowRate})$. Which of these is the intended meaning of the Dams \sim Transistors hint? That is, should we assume that our teacher means

Dams are like Transistors in that

- each uses energy in the form of energy it is regulating,
(if so, we should conclude $\text{CValve}(\text{Dams}, \text{FlowRate}, \text{FlowRate})$)
or
- each regulates energy in the form of current,
(if so, we should conclude $\text{CValve}(\text{Dams}, \text{FlowRate}, \text{Current})$).

Figure 9-3: How are Dams like Transistors?

Both answers are reasonable. While the first alternative seems more probable than the second here, we certainly can imagine a dam whose output energy is regulated in a feed-back manner — using the flowrate out to open or close the valves. However, for another analogy, the second option may be more reasonable. For example, one may be told that “A Pressure Regulator is like a Transistor”, using the Pressure Regulator shown in Figure 9-4, for the purpose of communicating the (correct) target conjecture that the FlowRate through a PressureRegulator is controlled by some FlowRate — *i.e.*, $\text{CValve}(\text{PressureRegulator}, \text{FlowRate}, \text{FlowRate})$ — which follows from the source fact that the Current through a Transistor is controlled by some Current — *i.e.*, $\text{CValve}(\text{Transistor}, \text{Current}, \text{Current})$.

(Of course, this transistor analogue does not convey all the possible information about pressure regulator. In particular, as a transistor’s current is not used to control the current through the transistor, this PressureRegulator \sim Transistor analogy cannot be used to describe the way a pressure regulator’s flowrate controls the flowrate through the pressure regulator.)

This example shows there may be many different meanings for a given “A is

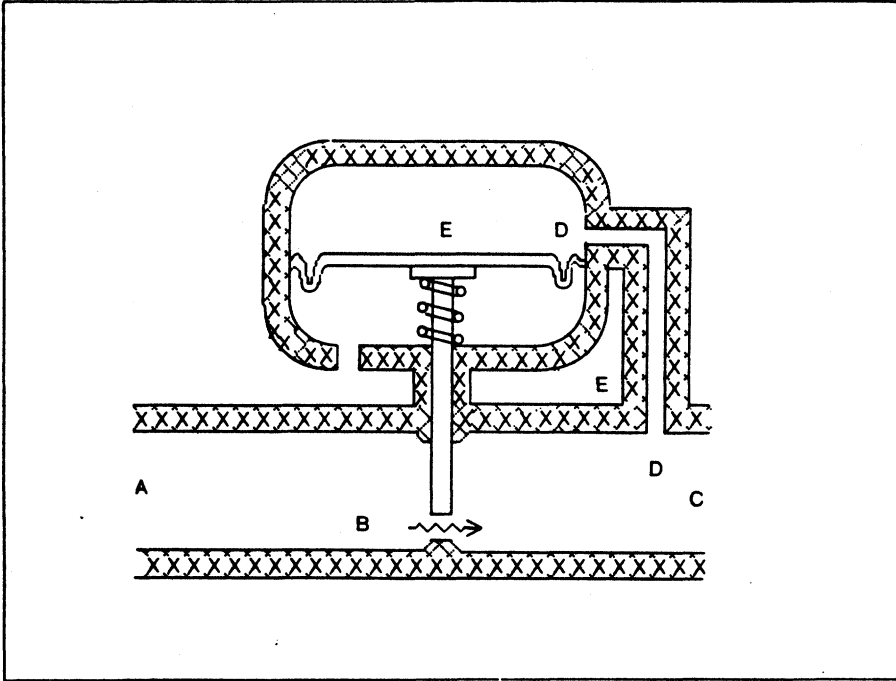


Figure 9-4: Simple Pressure Regulator
(from [dKB84, p9])

like B” hint, and which interpretation is selected depends not only on the information included in the source of the analogy (content) but on the formulation of that information as well (form). This, in turn, shows that the same information (here, $C_{\text{Valve}}(\text{Transistors}, \text{Current}, \text{Current})$) may lead to different meanings under different analogy contexts.

To state this issue more precisely: We may have different ways of expressing the same source facts: e.g., either as $\varphi(b_1, \dots, B, \dots, b_n)$ or as $\varphi'(b_1, \dots, B, \dots, b_n)$. Since they represent the same information, we of course have $\varphi(b_1, \dots, B, \dots, b_n) \equiv \varphi'(b_1, \dots, B, \dots, b_n)$. For example,

$$\varphi_{\text{CV}}(\text{Transistor}, \text{Current}) \equiv \varphi_{\text{CbC}}(\text{Transistor}, \text{Current}).$$

As shown above, we can derive different target facts depending of which formulation is used. Here,

$$\varphi_{\text{CV}}(\text{Dam}, \text{FlowRate}) \neq \varphi_{\text{CbC}}(\text{Dam}, \text{FlowRate}).$$

The general observation is,

$$[\varphi(b_1, \dots, B, \dots, b_n) \equiv \varphi'(b_1, \dots, B, \dots, b_n)] \not\equiv [\varphi(a_1, \dots, A, \dots, a_n) \equiv \varphi'(a_1, \dots, A, \dots, a_n)] \quad (9.8)$$

In essence, which target sentence is derived depends on what constants remain constants (these are the constants lexically contained in the analogy formula) and which correspond to a formal parameter.

This versatility suggests that an analogy system must be able to consider deductions which follow from the starting theory. That is, it should be able to produce either interpretation for the analogy, given only the above CValve fact and Definition 31's definition of `ControlledByCurrent` — even if this requires deriving the needed `ControlledByCurrent(Transistors, Current)`. (Note:2-5's Point NotStd2 reiterates this point.)

9.3.2 Need for Reformulation

Definition 33 in Note:2-4 suggested that an analogy is a simple parameter substitution. This certainly works when *Th* stores its facts correctly. (In this case, the analogy is considered degeneratively obvious — a trivial isomorphism.) Unfortunately, *Th* reserves the right to encode its contained information in completely arbitrary ways. The previous subsection demonstrated that different representations of the same information can lead to different analogies. The discussion below elaborates this point, showing how the nuances of a representation can hide an otherwise apparent analogies. Indeed, much of the complexity of our analogy system follows from our desire to recover such hidden analogies.

Why go through these contortions? Why should an analogizing system attempt to find such apocryphal connections? Our motivation stems from our belief that analogy should be a *semantic* relation, which depends on the analogues themselves, rather than on their representations. (See Section 8.1.) Wem unfortunately, must depend on a particular representation of these analogues: this is encoded in *Th*. While the content of this theory dictates which analogies can be found, we can

still ignore much of its form — this is why *Analogy_F* downplays the nuances of the representation as much as possible.

Focus now on the predicate calculus notation used throughout this dissertation. This partial theory encoding stores certain facts explicitly, (i.e., each $\rho \in Th$ is explicit to Th) and others implicitly (those $\sigma \notin Th$ such that $Th \models \sigma$).¹² Note:2-5's Point NotStd2 argues that we should ignore this distinction by dealing with $DC(Th)$, the deductive closure of the starting Th . This subsection shows that the desired language of the target instantiation may include symbols not in the language of Th . This means that finding a desired analogy from a fixed initial theory may involve adding new symbols to the initial language as well as new facts to the initial theory: incorporating these additions constitutes the learning process.

As an example, consider the arity of the `VoltageDrop` relation symbol. Kirchoff's Second Law, which states that the voltage drop between a pair of points is independent of the path taken, renders the third argument to `VoltageDrop` irrelevant. Knowing this, the Th theory might have used a binary `VoltageDrop2` function instead of the ternary `VoltageDrop` one shown throughout this dissertation. If Th continued to use the ternary `PressureDrop` function, we are in trouble: there is no single formula which, when instantiated with `VoltageDrop2`, produces

$$\forall loop. \sum_{\langle i,j \rangle \in loop} \text{VoltageDrop}_2(i, j) = 0, \quad (9.9)$$

and with `PressureDrop`,

$$\forall loop. \sum_{\langle i,j \rangle \in loop} \text{PressureDrop}(i, j, [x]) = 0. \quad (9.10)$$

A better way to find the desired correspondence between these theories begins by defining a binary `PressureDrop2` function,

Definition 32 $\forall j_1, j_2. \text{PressureDrop}_2(j_1, j_2) \stackrel{\text{def}}{=} \text{PressureDrop}(j_1, j_2, [\text{AnyPath}(j_1, j_2)])$,

¹²This distinction between explicit and implicit information is related to the ideas presented in [MG84].

where this *AnyPath* function returns an arbitrary path between its arguments. This can be used to derive a correspondent to Equation 9.9,

$$\forall loop. \sum_{\langle i,j \rangle \in loop} \text{PressureDrop}_2(i, j) = 0. \quad (9.11)$$

(Alternatively we could have derived a ternary *VoltageDrop*₃ function from the given *VoltageDrop*₂.¹³ In general, we prefer to change the representation of the less known *target* rather than the more established *source*. The abstractions discussed in Chapter 4 provide a top-down (and hence, better) criterion for determining which must be reformulated.)

This *PressureDrop*₂ from *PressureDrop* example provides one reason why new symbols may need to be added: *viz.*, to provide additional needed constants. Another reason is to hide certain values, leaving them invariant from one analogue to another.

Consider the *CValve* example presented in Subsection 9.3.1, and imagine we wanted the *Transistors~Dams* hint to communicate the message *CValve(Dams, FlowRate, Current)*. This is possible only if we have something like the *ControlledByCurrent* relation, which makes the second occurrence of *Current* one of the analogy's invariants. This means that if *Th* did not include this relation, we would have to generate (something like) it for this message to get through. (Note: 9-7 shows that we never need to generate a new symbol to expose a hidden value to some symbol-symbol transformation.)

In line with the theme which prefaces this subsection, an analogy should not depend on which particular concepts happen to be explicitly defined (*i.e.*, reified) in the source and target theories. This means that we may have to generate new symbols to find an analogy joining two concepts. That is, given an initial theory, we need something slightly stronger than its deductive closure when seeking an analogy: we may have to add definable new symbols to the language, as well as deducible new facts to the theory.

We close this subsection with some final comments on the nature of this implicit versus explicit issue.

¹³Yes, this *VoltageDrop*₃ is just another name for the *VoltageDrop* we used above.

EI1. Ties to Efficiency of Finding Analogy, Not Goodness of Analogy:

These representational nuances do determine how easy it is to find a particular analogy. While this does affect how easy it is to find an analogy, it should not determine whether it is an analogy, nor how good that analogy is. (See Note:3-1.)

EI2. Form, not Content:

It is the form of the representation which is irrelevant, not its content — which facts are included in $DC(Th)$ is important in determining the analogy, but how such facts are encoded (e.g., whether they are implicit or explicit) is not. For example, if (the deductive closure of) our theory did not include Ohm's Law, we should not be surprised if it does not lead to an analogy useful for the "Find the flowrate" problem.

EI3. Reformulation:

This implicit versus explicit issue ties in with the notion of reformulation, as discussed in [Ama68], [SGLS80], [Tap81] and [Low84]: in all of these cases, solving a problem requires (or is aided by) the addition and use of new terms.

EI4. Tie to Abstractions:

We can use abstractions to dictate which representations to seek. This information bounds both the search for new facts and for new symbols. (These abstractions can also serve to hide invariant values, the second rôle mentioned in above.)

EI5. Is Reformulation Ever Needed?

This final point addresses the claim that, perhaps, reformulation is never *really* needed. Some might argue that the Knowledge Base designer should know, at data input time, how the facts will be used and store them accordingly. As evidence, they cite the number of existent Knowledge Bases which use only *prima facie* facts, without the need for reformulation. I argue otherwise, claiming that a versatile learning system must be able to reformulate its facts as necessary, making this "reformulation pre-step" an important part. (See Note:3-1.)

Another way to think of this is in terms of biases (see [Mit83]). One way to bias the learning system is to force it to use only the initial representation. This is unnecessarily limiting. The use of abstractions is a more general biasing technique, allowing a collection of facts to have, in effect, many different representations. That is, as each abstraction provides a different “interpretation” of the initial data. (Of course, some form of bias is necessary: permitting arbitrary reformulations would lead to an infinite space. This use of abstractions provides the constraints needed to restrict this space, focusing it into a manageable size. See also Footnote 8 on page 63.)

Recall the caveat contained in Equation A.4: that any analogy could be expressed as a sentence-to-sentence mapping induced by some symbol-to-symbol mapping, *provided the analogues are represented appropriately*. The ability to add derivable new facts and create derivable new symbols as needed means that this analogy process can re-represent the initial facts, allowing it to learn any legal *Analogy_F*. This also explains the observation that *every analogy is obvious in retrospect*: understanding an analogy requires first finding, or deriving, the apt representation. Once one has the appropriate formula φ , the analogy itself become an obvious isomorphism, and the learning step reduces to a trivial parameter substitution, mapping each term in the source instantiation into a corresponding target term. This means that every analogy can be *reduced* to an isomorphism by representing the analogues appropriately.

9.4 Literature Survey

“The Lord is my shepherd, ...”

Psalm 23

This section provides a quick summary of how my model of analogy compares with those of other AI researchers, both in the area of analogy and learning.

Lit1. Non-AI Views of Analogy — Philosophy, Psychology, Linguistics ...

Many fields have tried to understand the phenomenon of analogy, often in terms

of its literary cousin, metaphor. The recent artificial intelligence interest in this subject has been pre-dated (often by millenia — *cf.*, Aristotle’s description of “educing the correlate” [Ari52, *Rhetoric*, III, iv, 1-3]) by the tremendous body of literature on this topic from philosophers, linguists, psychologists and poets, to name just a few. Ortony’s excellent anthology [Ort79] provides one smattering of the diverse fields interested; see also [Pol54], [Hes66], [Dar78], [Hru83], [Mar84], [She84], ...¹⁴ (Of course, this Artificial Intelligence dissertation is not the place to delve into a long description of the history of analogy and metaphor. The interested reader is referred to [Bea67] and [Dav85], for a summary of the core philosophical positions, and to [Fer67], for the basic theological positions.)

It is reassuring to find that researchers in other fields share a similar view of analogy; *i.e.*, their view is similar to one shown at the start of Section 9.1. Mill epitomizes the traditional approach, claiming that “... a fact m , known to be true of A , is more likely to be true of B if B agrees with A in some of its properties...” [Mil00, p394]. As a more contemporary example, Hesse, a philosopher of science, employs a similar description of analogy [Hes66]. She uses the term “positive analogy” to refer to the set of features which are known to match from analogue to analogue; this is what I have labeled the starting “kernel of correspondence”. One learns more about target analogue by investigating the “neutral analogy”; these are the features which are not known to correspond. Those features which are found to contradict are labeled the “negative analogy”, and are explicitly eliminated. (Harré slightly refines this description, describing different types of models which can be used to guide the process [Har72, esp p175-75]. Darden describes some of the obvious issues which arise when one attempts to use this formalization [Dar83].)

As an example more relevant to my research, it was Polya who first claimed that an analogy is a common abstraction [Pol54]. Much of this dissertation can be viewed as an attempt to formalize this general idea. (Indeed, much of AI can be viewed as attempt to formalize many of Polya’s insights...)

¹⁴In fact, the Stanford University libraries include several hundred citations under the entries “Analogy” and “Metaphor”.

Lit2. General Induction

There are many AI programs which search for a generalization of one or more instances. (These include [HM78], [And83] and [Mic83], as well as [FHN72] and [Sus75]. See [DLCD82, esp Section D] and [DM83] for a general overview.) The underlying analogy process is quite similar. (In fact, Anderson refers to the GRAPES system's generalization process as "analogy compiling" [And83].¹⁵) Each of these induction techniques is based on maximizing commonality. My approach is based on the realization that certain clusters of facts are *a priori* more useful than others. These coherent clusters are our "abstractions".

Of course, my approach only applies to domains and tasks which admit such clustering. In those cases, exploiting these biases (read "abstractions") has proven a useful way of constraining the search.

Lit3. Re-Using Past Problems

A great deal of recent research has been devoted to one particular way of generating (what I would call) abstractions: *viz.*, by watching the performance element solve some problems. The primary example is the excellent work on *chunking* by Rosenbloom, Laird, and Newell [Ros83,RN82]. Mitchell's recent [MMS85a] project is also in this camp, as is Carbonell's use of derivational analogies [Car81a,Car83b]. This clearly ties in with the general area of learning for a purpose; see the survey presented in Section 3.1.

Those works describe one way of generating abstractions; *NLAG* demonstrates that analogical inference is a real application of such compiled knowledge.¹⁶ As such, those methods are complementary to, but independent of, my *NLAG* research. We all agree that solutions to past problems are useful.

Lit4. Causal connection

¹⁵This apparently stems from the view that the variabilization process used here is a type of symbol-to-symbol mapping, and hence akin to the mapping used in its analogy processing.

¹⁶This also fits the definition of a heuristic presented in [Len82b]: a heuristic is information which, if only you had had earlier, would have helped to solve the problem.

Winston's sense of "causal connection" [Win79,Win80,Win81] is somewhat related to my use of abstractions: each is used as a way of conjecturing plausible propositions. There are several important differences, though. Winston's causal connections tend to be local and "bottom up": from one fact, postulate another. This new fact may then lead to other conjectures, and so on. My abstractions tend to be more global: each defines, *in toto*, the way in which a pair of analogues are usefully interrelated. While causal connections attempt to construct (possibly useful) similarities; my common abstraction approach, by its nature, leads to some collection of useful facts. (Section 9.2 already elaborated this point.)

Another difference is in terms of directionality: Causal connections are unidirectional, from one fact to another. We think of abstractions in terms of co-occurrence: as such, any subset of its clauses can suggest the others. This relates to the general issue of indexing: abstractions are more generally indexed, and so are accessible from any derivable term. (See Section 4.5.)

The final issue deals with explicitness of the information. Abstractions are very explicit about what remains invariant and what is expected to change: the variables change (by instantiation), and the rest is fixed. It is not as clear what is invariant in a causal connection. (See Subsection 9.3.1.)

Winston's most recent work ([WBKL83], in collaboration with Binford, Katz and Lowry) extends the earlier work in several significant ways. After providing a brief description of the [WBKL83] analogy process, this point discusses those extensions.

Its underlying task is to derive form from function: given a functional description of an object, the program proposes a physical description which can be used to identify that object. The approach exploits the observed co-occurrences of certain functionalities with certain physical properties: e.g., that "stable" objects tend to have flat bottoms.

As with the other systems, this program starts with a kernel correspondence between the source and target objects and then extends this to propose new properties of the target. Here, the kernel consists of the functional specification

common to both objects. (*E.g.*, the source “brick” and the target “cup” should each be stable.) This is extended by following causal links which tie those functional attributes to physical properties; this results in new conjectures about the target object. (Hence, the fact that bricks have a flat bottom is used to suggest that a cup should have a flat bottom.)

This work differs from Winston’s earlier systems in one important aspect, *viz*, it is motivated by a specific problem. Here, for example, the task is to build a cup-identifier. This means it can use partial matches. Rather than seeking all ways in which a cup is like a brick, it notes only that each is stable, and then considers the plausible entailments this suggests. (This also leads to another, albeit minor, difference: this analogy system may find and use many different source objects for a given target, each suggesting its own specific set of properties. In addition to using “bricks” to suggest the physical properties associated with stability, it uses “suitcase” to learn about liftability, “bowl” when considering how to contain liquids, *etc.* Kedar-Cabelli provides a further extension in this direction [Ked85].)

This reflects an awareness that an analogy should be “bounded”, focused to the needs of some problem. The resulting, more constrained approach is much more similar to my *NLAG* work, as well as many of the others discussed below.

Lit5. Selective Inference

Hobb’s research holds that an analogy is formed by first considering the facts which are known to be shared by both analogues, and then extending this connection by applying various legal inferences to posit additional facts about the target analogue [Hob83a,Hob83b]. Like my *NLAG* model, this perspective also considers the analogy process to be a form of inference. It differs by using many little steps to form the full analogical connection, rather than posit the full body of new information at once. (*I.e.*, it goes “bottom-up” rather than “top-down”.)

In this respect, it resembles Winston’s work, described in Point Lit4 above: in each, the analogy “spreads” from certain known connections to others. Once again, this leads to the maximal commonality view of analogy. (See Section 9.2.) In particular, the selective-inference-analogy draws all inferences which might be

acceptable: *i.e.*, it finds all connections which do not lead to a contradiction. Like Winston's latest system, Hobbs' selective inference model uses the context of a specific problem to help determine which inferences should be used.

Lit6. Concept Acquisition

Lenat [Len82a] and Burstein [Bur83a,Bur84] each discuss how one might learn new concepts by analogy. Lenat's system, AM, was able to use a noticed analogy between a pair of concepts to suggest values for some slots of the target concept. These new values would often require that some new concept be created. For example, the "Bag to Number" analogy was used to suggest that new "Number-operators" be generated to resemble known "Bag-operators". (See [Len82a, esp Section 5.2.6 and Appendix 3.2.4].)

Burstein uses several analogies to teach the computer operation of assignment to his CARL program: for example, one analogy links variables to boxes and another links the assignment operator to algebraic equality.

His papers explicitly discuss many of the same issues addressed in my dissertation, and it is encouraging to see that we reached very similar conclusions. In particular, he discusses why reformulation is necessary during the analogical learning process and uses this to explain why a top-down approach is needed to focus and delimit the search. (He also distinguishes between exhibiting an analogy and learning by analogy, and even uses the term "abstraction" to refer to the models CARL uses!)

Our approaches do differ. We had slightly different objectives: Since Burstein's work was partially motivated by psychological studies, his validation involves demonstrating that his CARL program would make the same errors students frequently made. (My interest is more theoretical — just what is an analogy, and how can they be used effectively.) A more significant difference is our respective views on the exactness of an analogy. While I maintain that an analogy can be exact (see Subsection 9.2.1), Burstein feels they are inherently approximate, claiming that the relationships in one domain are only similar (as opposed to

identical) to ones which hold in another. That is, he does not insist that a relation which holds in the source domain also holds in the target domain; instead, he claims that one may need a “more general version” of that source relation. (This may follow from his attempt to accurately model the limitations of his human subjects: in particular, this may accurately reflect their confusion concerning second-order relationships. See Note:9-2.) This leads to his interest in debugging incorrect inferences.

A final difference is our respective uses of the idea of abstractions. My abstractions are “bigger” than his: each of mine encodes the full way in which a pair of analogues are interrelated. This means that *NLAG* forms the analogical connection all at once, by instantiating a single abstraction. By contrast, the *CARL* system builds the analogical correspondences incrementally, using a variety of abstraction-ettes. (This means his use of abstraction resembles Winston’s use of “causal connection” or Hobbs’ “selective inferences”: each represents a single connection.)

Lit7. Invariance Hierarchy and Structure Mapping

Both Gentner and Carbonell make the claim that n -ary relations are more likely to be preserved than unary predicates ([Gen80b,Gen80c], [Car81b]). Why? One answer is in terms of commonality: Finding that a term — e.g., *hovercraft* — qualifies in some position of a relation — e.g., *transportation* — by itself, is sufficiently special to suggest other terms as well. This suggests that relationships are likely to be preserved, *modulo* substitution of terms.

This, of course, depends on how which relations were codified. Fortunately, people do quite well at finding natural and useful relations — see Subsection 7.6.3.

One way in which these versions of analogy resemble the \mathbb{H}_{PT} system is that they all deal, in essence, with the *variables* of some formula, rather than the *values* of these variables. (This also appears in ACT’s use of a functional template [And83, p204], and in the Programmer Apprentice work, e.g., [Ric79].) That is, we all feel it is the formal interconnections between the sets of analogous terms which are important, not the details of what the terms happen to be. This

explicates what remains invariant between the analogues: *viz.*, those connections.

Lit8. Derivation Analogy

The *NLAG* system deals only with one type of learning, where the goal is to acquire new, independent facts. Another purpose of an analogy may be to focus a search: for example, to provide hints which guide a deduction.¹⁷ This is the purpose of Kling's Zorba system [Kli71]. There, a detailed proof of a theorem in the source domain is used to guide the proof of a related theorem in the target domain. In essence, Zorba uses the analogy to specify which clauses should be considered first during the deduction.

Carbonell's recent derivation analogy work (an extension to his ARIES system) has a similar theme [Car83a]. His system begins by storing the decisions made during a problem solving session: in particular, which branches proved successful and which did not. It then tries to re-use this meta-level information when solving a similar problem: this information is used to suggest which branches to take and which to avoid, to guide the search for a solution to this target problem.

Of course, these systems can only offer advice if they find the current situation sufficiently similar to the original situation. This analysis is based on a base-level similarity between the two problems/ Hence the kernel of analogy is not strictly at the meta-level, even though the information transferred is.

Lit9. Similarity Analogy

This includes the work of Evans [Eva68], Thibadeau [Thi77] and Hofstadter [Hof84], as well as the earlier ideas of Aristotle [Ari52].¹⁸ My major comment about these cases is that my *NLAG* system is unable to handle them. This is because *NLAG* relies on abstractions, and abstractions only make sense when there are *a priori* clusters of facts. Unfortunately, the space of similarity analogies is too homogeneous.

That is, given some $A:B :: C:?$ problem, the *wisdom of the ages* extends only

¹⁷Note:2-2 elaborates this distinction.

¹⁸In fact, even the word, "analogy", derives from the Greek "αναλογια," which means proportional.

as far back as that first $A \mapsto B$ trial. That is, there is no reason beyond the one earlier case to make any predictions.

Compare this to, e.g., the **Group** axioms: when some operation is **Associative**, it does seem a good bet that it is **Closed** over some domain; etc. That is, these naturally occurring facts do seem to cluster.

NLAG also cannot handle the related case of *similarity metaphors*. Why in the world should **East**, **Sun** and **Stars** fit together, in describing **Juliet**? ... and in particular, her balcony, Juliet, and her eyes? ([Sha72]) Ask yourself what possible set of problems you could solve with this information.

Lit10. Types of analogy

The recent [FG83] paper discussed three types of analogy-like situations, *viz.*, abstractions, analogy and similarity. My work is squarely in understanding the first process, giving some teeth to [Pol54]'s claim that analogy can be viewed as a common abstraction. (I see another contribution of my work is in further distinguishing these categories.)

This seemed related to the philosophical and linguistic discussions on live versus dead metaphor. (*Cf.*, [Mar84], [Sea79] and [Pyl79].) A dead metaphor¹⁹ is a phrase which can now be semantically understood without appeal to the original metaphor. (The canonical example is the "*leg* of a table".) A live metaphor, on the other hand, is semantically false, and can only be understood if it is regarded *qua* metaphor. (An example is referring to the person, Bill, as a chair.) There seems a full spectrum between these cases — e.g., consider phrases like "social butterflies".

By analogy, a "live analogy" is one where the connection between the two analogues is totally novel and does not correspond to any known generality. Let me define a "dead analogy" as one where the connection is deducible. Within my abstraction framework, this would mean that not only is the common abstraction known, but so is the full instantiation of both the source and target analogues — *i.e.*, each is deducible from the initial theory.

¹⁹[CM83] refers to these as "frozen metaphors."

Recall that \Vdash_{PT} insists that the target instantiation is not deducible. This places *NLAG* between these two end-points: while both the abstraction and the source instantiation must be known, the target instantiation is not.

Chapter 10

Conclusion

“Perhaps every science must start with metaphor and end with algebra...”

Models and Metaphors, Black (1962)

This final chapter fits my *NLAG* model of analogical inference into the overall picture of both analogy and learning systems. Section 10.1 begins by placing this work. Section 10.2 describes some future research directions, continuing both the main line of this research as well as pursuing themes tangentially related. Section 10.3 concludes by listing the major contributions of this research.

10.1 Placement

This section states the objectives of the *NLAG* system, in order to place it within the space of Artificial Intelligence systems. Many of these general comments reflect and generalize some of the specific differences discussed in Section 9.4.

First and foremost, *NLAG* is primarily a learning system, in that the overall system *exhibits superior performance on subsequent occurrences of the same and similar problems* (paraphrased from [Sim83, p28]). This is because *NLAG* adds additional facts to the underlying knowledge base during each run, facts which allow the standard inference engine to solve the original target problem, as well as many similar problems. (As this inference engine had been unable to solve the

problem using the initial knowledge base, this is a definite improvement.) Using the definitions presented in Note:2-2, this means that *NLAG* is a *Learn₁* process.

Another obvious characteristic of this *NLAG* system is that it deals with analogies, here, for the purpose of conjecturing underivable facts. This follows from the **Common** condition appearing throughout, *cf.*, Figures 2-1, 3-1 and 4-5. This tells the conjecturing process — here, analogical inference — to select only particular underivable facts, namely those which correspond to facts known to hold for the source analogue.

The third relevant characteristic is that *NLAG* is concerned with solving a particular problem. In particular, *NLAG* attempts to find just *useful* conjectures, *i.e.*, just conjectures which suggest ways of solving that specific problem. This means it is seeking useful analogies, as opposed to establishing arbitrary commonalities between the analogues. *NLAG* is, basically, following the *case method approach*: using particular problems to motivate and guide the search for new information.

Another distinguishing characteristic of this process is its top-down nature. Each abstraction is a specific model, which serves to focus *NLAG*'s search. Many other AI systems (especially analogy-seeking programs) work in a bottom-up fashion, often attempting to form the “generality” during the process. (See Note:4-3.)

A final difference between this research and many others is methodological. While many reports describe only an effective algorithm or relate their results to psychological observations, my primary goal is to provide a clean and formal conceptual framework for this analogy process. This is why this dissertation has provided a semantic account of analogy, various syntactic definitions and a collection of formally stated heuristics, as well as a detailed description of a running implementation.

10.2 Future Work

While this research has provided several insights in the area of abstraction-based useful analogical inference, it is not *the solution* to either the “learning problem” or the “analogy problem”. This section discusses some future issues and describes

how they extend the *NLAG* model.

FW1. Abstractions:

The \Vdash_{PT} model of analogy deals exclusively with utilizing known abstractions. A more comprehensive analogy system may be able to acquire new abstractions “on the fly”, based on the current situation (which may include the particular task at hand, any hints given, the learner’s initial state and internal biases, *etc.*).

Other than the quick discussion in Note:4-1, this dissertation has not addressed the question of how to generate these re-usable formulae, which we labeled as abstractions. An important prerequisite is a better understanding of what these abstractions are. For example, should every relation qualify? (Subsection 7.6.3’s comments imply the answer may be “yes”.) Or perhaps just every n -ary one, where $n > 1$?¹ A related question is “How else might they be used?”; this is partially addressed in Note:4-2.

The rest of this point addresses the question of finding abstractions in general, extending the comments made in Note:4-1. This task appears to have two components: first find a useful perspective (see Definition 1 in Section 4.1) and then re-express this collection of facts in a more general, re-usable form.

Section 4.1 described one possible solution to the first part. (Rosenbloom’s recent work discusses many of the subtle complexities in this straightforward-seeming task [Ros83].) The second part, determining which constants should become variables and which remain fixed, seems more problematic. For example, why is *OhmsLaw* considered a fact about *Current* and *VoltageDrop* (which are allowed to vary to for example, *FlowRate* and *PressureDrop*), but not about multiplication, junctions or equality, which remain as fixed constants?

A semantic understanding of learning may provide useful insights on this issue. This is discussed in the next point.

FW2. Semantic Analysis of Learning:

Learning is often described in purely syntactic terms, in terms of incorporating

¹This relates to Note:10-1. Notice also that a great many abstractions deal with second-order facts that is, many take functions and relations as their arguments. This may be significant...

new facts [DLCD82]. It is, as such, subject to the nuances of the particular representation involved. This research on analogical inference has produced, as a side-product, a preliminary description of a semantic approach to learning. This is discussed extensively in Chapter 8 and is used in defining Triviality and Aboutness (in Section 2.4), determining how constraining a fact is (in Section 5.3), and defining Commonality (in Section 9.2). (In a separate but related work, [GG83] used this semantic approach to define an extentional definition of novelty. See Note:8-1.)

I feel this semantics-based approach will prove useful to learning in general. For example, it may help us understand how abstractions are generated. The prior point mentioned how bundles of sentences might be produced, but did not discuss how these facts can be “variabilized”. Note:10-2 suggests how this semantic approach may provide an answer.

This ties in directly to the issue of new term creation in three ways. First, we suggested above how this process could be used to produce those terms we label as abstractions. This, in turn, leads to the second use: of reifying some term used to instantiate an abstraction. (This was mentioned in Section 4.1.) The third use is even more grandiose: perhaps this semantic account leads to an effective way of defining the ontology itself, based on an understanding of efficient methods of grouping the known objects. Such a semantic analysis could help define an ontology which effectively “cuts the world at its joints” [Boy79].

Given the relatively quick progress made in the areas of defining triviality, aboutness, commonality and novelty, and in quantifying how constraining a conjecturing is, the area of learning semantics seems ripe for future research.

FW3. Other senses and uses of analogy:

This dissertation has explored only one type of analogy task: using analogy as a mechanism for acquiring unprovable base facts. Furthermore, it has exploited only one way of focusing the search: via the goal of solving a particular problem. It also assumes the inference system used by the learning system’s performance element ([BMSJ78]) was deductively complete. The remaining points consider

some variants of these conditions.

The intuitive notion of learning by analogy covers many other cases. For example, an analogical hint could suggest that the learner focus on some derivable but “hidden” fact, especially in situations when the learner’s deductive capabilities are incomplete (e.g., resource limited). Here, the analogy may serve to “re-arrange” the learner’s theory rather than add to it.² We could extend the \Vdash_{PT} model to capture this case by changing the logical inference operator, \models , which appears in Figure 4-5, to a different deductive operator, one which may be neither sound nor complete. It is not clear what new problems this change will cause.

A different but closely related extension involves weakening \Vdash_{PT} ’s requirements for the source analogue. For example, we might require that the source abstraction instance be only satisfied by the initial theory, rather than deduced from it. This means that the source and target analogues would be treated symmetrically. (I suspect that the I_{Close} and I_{Least} intuitions would still apply; it would be interesting to see if this is true.)

We have only considered the task of learning by analogy. This seems related to the task of using analogies as a tool for explanation: there the teacher is attempting to use the learner’s \sim process to laconically communicate a large corpus of information. Do special problems arise in this situation?

Section 9.4’s Point Lit9 already discussed why this abstraction-based model of analogy is inappropriate for proportional analogies, as that task involves educating the particular common formula. Is this task, of generating a locally re-usable formula, different from the process of generating the globally re-usable formulae we call abstractions? Do some or all of the processes suggested in the points above apply? That is, what are the special issues associated with producing general utility formulae (aka abstractions) versus the special re-usability needed for proportional similarity. Perhaps other behavior would be required; and almost

²Using the terms defined in Note:2-2, we have only considered $Learn_1$ purposes; an analogy might be used for $Learn_2$ purposes as well.

certainly other heuristics would be preferable.³

Going one step farther, how does this sense of analogy compare with analogy *qua* classification scheme — *à la* Wittgenstein’s “game” (see [Wit53])?

FW4. Space of heuristics:

This dissertation presents a variety of heuristics which guide *NLAG* through the space of possible analogies. (The important ones are summarized in Section 5.4.) They do not exhaust the full space of possible heuristics. There are certainly other rules which either enhance those Section 5.4 rules, or offer advice in situations where they do not apply.⁴ (As the rules presented in Section 5.4 are well justified, I am not considering additional (or alternate) rules which violate them.⁵)

All of these rules are domain independent; there are domain-dependent ways of ordering the search as well. For example, Subsection 5.2.1 noted how weak the H_{JK} rule was. There must be other, more specific ways of using the problem statement to find the relevant abstraction. (Each would propose a more specialized way of using the target problem to determine the appropriate abstractions and abstraction instances.)

Another set of rules may operationalize some standard observations. For example, for many source analogues, one abstraction tends to be *a priori* more likely than the others. (Consider the phrase “John is a pig”, and notice that only one of pig’s set of possible abstraction seems obvious.) One type of rule might use such domain-specific information to dictate that this particular abstraction be considered first. This is related to the use of a “salience” measure and to

³Shepard provides a wealth of evidence that the process of finding a common formula is essential to general human problem solving [She84]. That article describes the incredible contortions people go through to see two (consecutive) objects as instances of the same one object, but somehow distorted. This suggests that people automatically try to find the commonality joining a pair of possibly related objects, however muddled and well camouflaged it may be. See also Footnote 23 in Note:4-5.

⁴There is tremendous latitude here, as many of those heuristics are just approximations to the underlying ideas; *e.g.*, many are syntactic approximation to semantic constraints. This alone gives us license to modify them, providing we stay within their frameworks.

⁵The only rule I considered changing was H_{MGA} . The empirical evidence presented in Section 7.5 argues against this change, demonstrating H_{MGA} ’s superiority over its direct competitor, H_{MSA} .

frozen metaphors (see [Sea79] and [CM83]). Each of these could lead to a class of relevant domain-dependent heuristics.

FW5. Other forms of Analogical Hints:

So far, we have considered only one type of analogical hint, always of the form $A \sim B$.⁶ In later implementations, this initial *AH* parameter may encode other information as well: e.g., “A is like B and C is like D”, “A is like B, in some manner C”, or “A is like B, except in that C”, etc.

We have also assumed that there is a single target problem to solve. Could this requirement be weakened or eliminated? Perhaps a more elaborate system could, instead, utilize a set of positive and negative instances of some phenomenon.

(To tie this in with Point FW4, such extended systems would need other rules to use this other type of information.)

10.3 Contributions

This section lists the major contributions of this research effort. Its primary goal is the construction of an effective procedure for using a given analogical hint to propose conjectures which are useful for a particular problem. En route, I addressed the issues: [1] what is an analogical inference in general, [2] what is a useful analogical inference within the context of solving a particular problem, and [3] how can such useful analogies be found effectively. My dissertation also describes a running program to demonstrate that this model of analogy can be implemented and that it works effectively. To demonstrate its generality, this research also provides a semantic account of this analogical inference process, based on a variant of Tarskian semantics.

In my mind, the most significant long-term contribution of this work is a clarification and formalization of many of the issues related to analogy. In particular, this dissertation has defined the general concept of *analogical inference*, and discussed

⁶In [CM83]’s notation, this means we are describing “metaphorical reasoning”. By contrast, “analogical reasoning” occurs when the learner must first find the analogues.

could allow us to specify that all members of one class must belong in another class. (One interesting objective would be to produce any arbitrary *Allowed* set by using various learning steps, starting from the full set of all eligible interpretations. This starting set is $Allowed(\mathcal{A}, \{\})$, where this \mathcal{A} knows nothing about any symbol; i.e., $\underline{x}^{\mathcal{A}} = \langle \{\} \{\} \rangle$ for all $x \in \mathcal{L}$.)

A parallel approach involves augmenting the underlying theory Th (e.g., by adding a new sentence), and coördinating this with related modifications to \mathcal{RW} . (Notice that if every object can be (syntactically) named, then each semantic modification (read “change to \mathcal{RW} ”) can be stated syntactically.) However, this dissertation does not address either of these issues. (See Point FW2 in Section 10.2.)

Finally, this semantic system is related to several others. The semantic and syntactic arguments to *Allowed* relation resembles the $\langle L S \rangle$ pair of language and connectives mentioned in Weyrauch’s FOL system [Wey78]. Also, we observed above that the base partial interpretation \mathcal{RW} is the point-wise intersection of the set of all allowed interpretations. From this perspective, we see that \mathcal{RW} corresponds to what [Kri80] calls “implicit knowledge”. We can also think of each member of $I \in Allowed(\mathcal{RW}, Th)$ as a world *reachable* from \mathcal{RW} , using the notation from [Moo80], *et al.* (This connection is even tighter if we used the *Allowed'* relationship defined in Point PI2.)

One advantage of this *Partial Interpretation Semantics* notation over the others is that my system explicates exactly what is known, and not known. Rather than assign a multi-valued truth value to full propositions, this system operates at a smaller grain size, assigning partial extentions to the symbols. In particular, we can see just what is known about each lexical symbol: these are precisely the set of members of its known (positive and negative) extentions.

8.3 Semantic Definition of Analogy

This section uses the partial interpretation notion of semantics to define the analogy relation. The next section uses this definition of analogy to describe the analogical inference process.

Intuitively, an analogy conveys some similarity between the objects \boxed{A} and \boxed{B} . In particular, it means that both objects are members of the same class of objects. We need to refine this by considering only non-trivial sets.⁷ That is, we eliminate many degeneracies by imposing the non-triviality constraint, $\neg Trivial_{sem}(C, \mathcal{RW})$, to mean that the set $\underline{C}^{\mathcal{RW}}$ is not all encompassing:

Definition 26 $Trivial_{sem}(\varphi, \mathcal{RW}) \iff \varphi^{\mathcal{RW}} = \mathcal{U}$.

(It is straightforward to verify that *Trivial* corresponds to *Trivial_{sem}*.)

This non-triviality constraint makes sense: claiming that both \boxed{A} and \boxed{B} are members of the same class is meaningless if that class is universal.

The actual definition of analogy uses one further assumption, viz., that each constant symbol c is totally specified: i.e., ${}_+c^{\mathcal{RW}} = \{\langle \boxed{c} \rangle\}$ and ${}_-c^{\mathcal{RW}} = \{\}$. (See page 168.) This means the simple underlined form, \underline{c} , uniquely designates c 's "standard" denotation, \boxed{c} . (Equation 8.9's monotonicity conditions guarantee that this term is unambiguous; i.e., it always refers to this "standard" interpretation.)

This leads to a formal semantic definition of the analogy relation, *Analogy_{sem}*:

Definition 27 $Analogy_{sem}(\boxed{A}, \boxed{B}, \mathcal{RW}, \langle C \langle \underline{a_1} \dots \boxed{A} \dots \underline{a_n} \rangle \langle \underline{b_1} \dots \boxed{B} \dots \underline{b_n} \rangle \rangle) \iff \langle \underline{a_1} \dots \boxed{A} \dots \underline{a_n} \rangle \in {}_+C^{\mathcal{RW}} \ \& \ \langle \underline{b_1} \dots \boxed{B} \dots \underline{b_n} \rangle \in {}_+C^{\mathcal{RW}} \ \& \ \neg Trivial_{sem}(C|_1, \mathcal{RW})$

Hence, one way that $\boxed{+}$ is like $\boxed{*}$ is that each is a known member of a tuple included in Group; that is,

$$\begin{aligned} \langle \boxed{\mathcal{R}} \quad \boxed{+} \quad \boxed{0} \rangle &\in {}_+Group^{\mathcal{RW}} \\ \langle \boxed{\mathcal{R}_0} \quad \boxed{*} \quad \boxed{1} \rangle &\in {}_+Group^{\mathcal{RW}} \end{aligned} \tag{8.17}$$

Likewise, $\boxed{Current}$ is like $\boxed{FlowRate}$ in that each is a member of $\underline{RKK}|_1$, i.e., each object can appear as the first "argument" of RKK.

⁷In all that follows, we consider just *constructible* sets, i.e., we only consider $C \in CL_{\mathcal{L}}$. This additional constraint will be left implicit in the subsequent definitions. In general, this corresponds to the "constraint" that syntactic definitions (e.g., for *Analogy_F* or \vdash) deal only with formulae.

a series of refinements — [a] General Analogical Inference, [b] Useful Analogical Inference, [c] Abstraction-Based (Useful) Analogical Inference, [d] additional ordering and pruning heuristics and [e] various operational details. These culminate in an actual implementation, the *NLAG* program. Furthermore, the various models of analogy have been described semantically, syntactically and operationally.

Empirical results from this implementation confirm many of our intuitions about analogies, as well as provide other insights, especially on the nature of abstractions. In particular, the ensuing analysis explains the source of power underlying this abstraction-based approach (in terms of the synergy found by using the abstraction's coherent clauses) and suggests that the abstraction *label* is not that important (arguing that it has been linguistically trivialized). This leads to a characterization of the tasks and domains where this abstraction-based approach can be particularly effective.

The semantic account of these processes should allow future researchers to repeat these results in other systems, built using other representations. The formality of these descriptions, especially when augmented by the comparison to other systems, should facilitate extending this work in new directions.

This work has made a variety of secondary contributions as well. The formal definitions explicate many relevant distinctions, ones often blurred in other descriptions. Two examples are the differences between the analogy relation and the analogical inference process, and between the processes of general analogical inference and useful analogical inference.

Understanding the useful analogical inference process led to a series of discussions about the I_{Most} and I_{Least} intuitions, and to their resolution, in terms of the I_{Close} maxim. Many parts of this dissertation elaborate this I_{Close} insight, and demonstrate how our notion of *abstractions* realizes this idea. This research has described what these abstractions are (both intuitively and formally) and discussed how they can be used during the analogy process, why they can be used, and why they should be used.

I feel that this notion of clustering interrelated facts will continue to play an

important rôle in both learning and Expert Systems. (This is especially true in artificial domains, where each such cluster corresponds to a design or plan.) Note:4-2 elaborates this position.

Another contribution is the analysis of how this model differs from (and expands on) the standard models of analogy found in the literature. To summarize some of the arguments presented in Chapter 9: This work demonstrates certain shortcomings of the prevalent view, which holds that the best analogy is the one which express the “maximal commonality” between the analogues. In particular, we found that the resulting analogy is often inappropriate, can be extremely inefficient to find, and still might not lead to useful conjectures.

Chapter 9 also argued that an effective model of analogy should explicate the similarities and differences between the analogues, have an explicit model of the learner (in terms of his initial collection of facts) and, since an analogy is a semantic phenomenon, permit this collection of facts to be reformulated.

Other contributions of this project include the preliminary discussion of a semantic basis of learning, a better understanding of abstractions and the variety of useful additions added to the MRS system.

To conclude, the phenomenon of analogy appears ubiquitous in essentially all modes of reasoning, and especially in that elusive reasoning step called “learning”.⁷ A comprehensive analogy system must incorporate the abilities both to use a relevant abstraction when it exists, and to generate such an abstraction “on the fly”, based on the current situation. This research addresses one important part of that process: how to effectively find and use appropriate abstractions. The resulting abstraction-based routine, *NLAG*, represents an effective way of implementing this task. I anticipate a refined version will be an essential part in future, more elaborate learning systems.

⁷In fact, many psychologists describe learning in terms of assimilation, accommodation and adaptation ([McD79], [Pyl79]). Analogical reasoning plays an important rôle in all three.

Bibliography

- [AB73] John R. Anderson and Gordon H. Bower. *Human Associative Memory*. Winston and Sons, Washington, D C., 1973.
- [AFB78] R. Albrecht, L. Finkel, and J. R. Brown. *BASIC for Home Computers*. John Wiley & Sons, Inc., New York, 1978.
- [AFS83] John R. Anderson, Robert Farrell, and Ron Sauers. *Learning to Program in LISP*. Technical Report NR 157-465, Carnegie-Mellon University, September 1983.
- [Ama68] Saul Amarel. On representations of problems of reasoning about actions. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, pages 131–171, American Elsevier Publishing Company, New York, 1968.
- [And83] John R. Anderson. *Knowledge Compilation: The General Learning Mechanism*, pages 203–212. University of Illinois at Urbana-Champaign, Monticello, Illinois, June 1983.
- [Ari52] Aristotle. *Rhetorics, de rhetorica ad alexandrum, poetica*. In W. D. Ross, editor, *The Works of Aristotle*, Clarendon Press, Oxford, 1952. Translated by W. R. Roberts.
- [Aus62] John Austin. *How to do Things with Words*. Harvard University Press, Cambridge, 1962.

- [Bar79] David Barstow. *Knowledge-Based Program Construction*. Elsevier, North Holland, 1979.
- [BCEM78] James E. Bennett, Lewis Creary, Robert Englemore, and Robert Melosh. *SACON: A Knowledge-Based Consultant for Structural Analysis*. HPP Working Paper STAN-78-699, Computer Science Department, Stanford University, September 1978.
- [Bea67] Monroe C. Beardsley. Metaphor. In Paul Edwards, editor, *Encyclopædia of Philosophy*, pages 284–89, The Macmillan Company and The Free Press, New York, 1967.
- [Bla62] Max Black. *Models and Metaphors*. Cornell University Press, Ithica, 1962.
- [BMSJ78] Bruce G. Buchanan, Thomas M. Mitchell, Reid G. Smith, and C. R. Johnson, Jr. Models of learning systems. In *Encyclopedia of Computer Science and Technology*, Dekker, 1978.
- [Boy79] Richard Boyd. Metaphor and theory change: what is “metaphor” a metaphor for? In Andrew Ortony, editor, *Metaphor and Thought*, pages 356–408, Cambridge University Press, Cambridge, 1979.
- [BP83] Jon Barwise and John Perry. *Situations and Attitudes*. The MIT Press, Cambridge, MA, 1983.
- [Bra79] Ron Brachman. On the epistemological status of semantic networks. In Nicholas V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–49, Academic Press, 1979.
- [Bro79] Richard Brown. *Use of Analogies to Achieve New Expertise*. Technical Report AI-TR-403, Massachusetts Institute of Technology, April 1979.
- [Bro81] D. C. Brotsky. *Program Understanding through Cliché Recognition*. Master’s thesis, MIT, August 1981.

- [Bro84] Rodney A. Brooks. *Model-Based Computer Vision*. UMI Research Press, Ann Arbor, 1984.
- [Bro85] William J. Broad. Subtle analogies found at the core of Edison's genius. *The New York Times*, 15–16, March 1985.
- [BT78] Harry G. Barrow and J. Martin Tennenbaum. Recovering intrinsic scene characteristics from images. In A. R. Hanson and E. M. Riseman, editors, *Computer Vision Systems*, pages 3–26, Academic Press, New York, 1978.
- [Bur83a] Mark H. Burstein. Concept formation by incremental analogical reasoning and debugging. In Ryszard S. Michalski, editor, *Proceeding of the International Machine Learning Workshop*, pages 19–25, University of Illinois at Urbana-Champaign, Monticello, Illinois, June 1983.
- [Bur83b] Mark H. Burstein. A model of learning by incremental analogical reasoning and debugging. In *AAAI-83*, pages 45–48, AAAI, Washington, DC, August 1983.
- [Bur84] Mark H. Burstein. *Learning by Reasoning from Multiple Analogies*. PhD thesis, Yale University, New Haven, CT, 1984.
- [BvL81] John Seely Brown and Kurt van Lehn. Repair theory: a generative theory of bugs in procedural skills. *Journal of Cognitive Science*, March 1981.
- [BW77] D.G. Bobrow and T. Winograd. An overview of KRL, a knowledge representation language. In *IJCAI-77*, Massachusetts Institute of Technology, August 1977.
- [Car81a] Jaime G. Carbonell. A computational model of analogical problem solving. In *IJCAI-81*, pages 147–152, University of British Columbia, August 1981.

- [Car81b] Jaime G. Carbonell. Invariance hierarchy in metaphor interpretation. In *Proceedings of the Third Annual Conference of the Cognitive Science Society*, pages 292–295, Cognitive Science Society, University of California, Berkeley, August 1981.
- [Car83a] Jaime G. Carbonell. Derivational analogy in problem solving and knowledge acquisition. In Ryszard S. Michalski, editor, *Proceeding of the First International Machine Learning Workshop*, University of Illinois at Urbana-Champaign, Monticello, Illinois, June 1983.
- [Car83b] Jaime G. Carbonell. Learning by analogy: formulating and generalizing plans from past experience. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, 1983.
- [Cha84] Roz Chast. *Possible Universes: An Assortment of Cartoons*. Harper and Row Publishers, Inc., New York, 1984.
- [CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc., New York, 1973.
- [Cle81] John Clement. Analogy generation in scientific problem solving. In *Proceedings of the Third Annual Conference of the Cognitive Science Society*, pages 137–140, Cognitive Science Society, University of California, Berkeley, August 1981.
- [CM81] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog*. Springer-Verlag, New York, 1981.
- [CM83] Jaime G. Carbonell and Steven Minton. *Metaphor And Common-Sense Reasoning*. Technical Report CMU-CS-83-110, Carnegie-Mellon University, March 1983.
- [Coc80] Ira Cochin. *Analysis and Design of Dynamic Systems*. Harper and Row Publishers, Inc., New York, 1980.

- [Coh80] Paul Cohen. 1980. Indirect Personal communication. (*E.g.*, unofficial subtitle of Professor Silver's Logic Class was "Why Anyone Could have Discovered Forcing Theory"; UC Berkeley, 1975).
- [Cyp82] D. Scott Cyphers. *Programming Cliches and Cliche Extraction*. Working Paper 223, Massachusetts Institute of Technology, February 1982.
- [Dar58] Francis Darwin. *The Autobiography of Charles Darwin and Selected Letters*. Dover, New York, 1958.
- [Dar78] Lindley Darden. Discovery and the emergence of new fields in science. *Philosophy of Science*, 1:149–160, 1978.
- [Dar83] Lindley Darden. Reasoning by analogy is scientific theory construction. In Ryszard S. Michalski, editor, *Proceeding of the First International Machine Learning Workshop*, pages 32–40, University of Illinois at Urbana-Champaign, Monticello, Illinois, June 1983.
- [Dav85] Todd Richard Davies. Analogy. June 1985. Bachelor of Science with Honors in the Humanities.
- [DDH72] O. J. Dahl, E. Dijkstra, and C. A. R. Hoare. *Structured Programming*. Academic Press, 1972.
- [DeJ85] Gerald DeJong. A brief overview of explanatory schema acquisition. In Thomas M. Mitchell, editor, *Proceeding of the Third International Machine Learning Workshop*, Rutgers University, Skytop, Pennsylvania, June 1985.
- [Der83] Nachum Dershowitz. *Programming by Analogy*, pages 26–31. University of Illinois at Urbana-Champaign, Monticello, Illinois, June 1983.
- [Die85] Thomas Glen Dietterich. Learning at the knowledge level. 1985. forthcoming. (Based on his presentation given during the Third International Workshop on Machine Learning.).

- [dK] Johan de Kleer. An assumption-based tms. A unpublished XEROX PARC manuscript (1985).
- [dKB84] Johan de Kleer and John Seely Brown. *A Qualitative Physics Based on Confluences*, pages 7-84. Elsevier Science Publishers B.V., 1984. Reprinted from *Artificial Intelligence: An International Journal*, Volume 24.
- [DLCD82] Thomas G. Dietterich, Robert London, Ken Clarkson, and G. Dromey. Learning and inductive inference. In Paul Cohen and Edward A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, William Kaufman, Inc., Los Altos, CA, 1982.
- [DM76] Nachum Dershowitz and Zohar Manna. *The Evolution of Programs: A System for Automatic Program Modification*. Technical Report AIM-294, Computer Science Department, Stanford University, December 1976.
- [DM77] Lindley Darden and Nancy Maull. Interfield theories. *Philosophy of Science*, 44:43-64, 1977.
- [DM83] Thomas G. Dietterich and Ryszard S. Michalski. A comparative review of selected methods for learning from examples. In Ryszard S. Michalski, Jaime G. Carbonell, and Thomas M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983.
- [Dou83] Sarah Ann Douglas. *Learning to Text Edit: Semantics in Procedural Skill Acquisition*. PhD thesis, Stanford University, June 1983. Department of Cognitive Ergonomics.
- [Doy79] Jon Doyle. A truth maintenance system. *Artificial Intelligence: An International Journal*, 12(3), 1979.

- [EA81] R. Elio and John R. Anderson. The effects of category generalization and instance similarity on schema abstraction. *Journal of Experimental Psychology: Human Learning and Memory*, 7:397–417, 1981.
- [Emm85] Dorothy Mary Emmet. Analogy. In Philip W. Goetz, editor, *Encyclopædia Britannica*, page 337, Encyclopædia Britannica, Inc., Chicago, 1985.
- [End72] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, Inc., New York, 1972.
- [Eva68] Thomas G. Evans. *A Program for Solution of Geometric-Analogy Intelligence Test Questions*, chapter 5. The MIT Press, Cambridge, 1968.
- [Fei63] Edward A. Feigenbaum. Part 1: Artificial Intelligence: Introduction. In Edward A. Feigenbaum and Julian Feldman, editors, *Computers and Thought*, pages 1–10, McGraw-Hill Book Company, San Francisco, 1963.
- [Fel78] Jerome Feldman. 1978. Personal Communication: AI Colloquium at Stanford.
- [Fer67] Frederick Ferré. Analogy in theology. In Paul Edwards, editor, *Encyclopædia of Philosophy*, pages 94–97, The Macmillan Company and The Free Press, New York, 1967.
- [FG83] Kenneth D. Forbus and Dedre R. Gentner. Learning physical domains: towards a theoretical framework. In Ryszard S. Michalski, editor, *Proceeding of the International Machine Learning Workshop*, pages 198–202, University of Illinois at Urbana-Champaign, Monticello, Illinois, June 1983.
- [FH77] Richard F. Fikes and Gary G. Hendrix. A network-based knowledge representation and its natural deductive system. In *IJCAI-77*, MIT, August 1977.

- [FHN72] Richard Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robots plans. *Artificial Intelligence: An International Journal*, 3:251–288, July 1972.
- [Fil68] Charles Fillmore. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*, pages 1–88, Holt, Rinehart and Winston, New York, 1968.
- [Fin79] Nicholas V. Findler, editor. *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, 1979.
- [Fin85] Joseph Jeffrey Finger. *Residue: A Deductive Approach to Design Synthesis*. Technical Report Stan-CS-85-1035, Heuristic Programming Project, Stanford University, 1985. Also #HPP-85-1. (Submitted to *IJCAI-85*).
- [Flo78] Robert W. Floyd. 1978 ACM Turing award lecture: the paradigms of programming. *Journal of the ACM*, 22(8), August 1978.
- [FM83] Edward A. Feigenbaum and Pamela McCorduck. *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. Addison-Wesley Publishing Co., Menlo Park, CA, 1983.
- [Fra22] Sir James G. Frazer. *The Golden Bough: A study in Magic and Religion*. Macmillan, New York, 1922. Abridged version.
- [Fra79] Bruce Fraser. The interpretation of novel metaphors. In Andrew Ortony, editor, *Metaphor and Thought*, pages 172–185, Cambridge University Press, Cambridge, 1979.
- [Fre32] Sigmund Freud. *New Introductory Lectures on Psychoanalysis*. W. W. Norton, New York, 1932.
- [Fri79] Peter Friedland. *Knowledge-Based Experiment Design in Molecular Genetics*. PhD thesis, Stanford University, October 1979.

- [Fri84] Peter Friedland. 1984. Personal Communication, related to MOLGEN project.
- [FW80] Funk and Wagnalls. *Funk and Wagnalls Standard Dictionary*. Lippincott and Crowell, 1980.
- [Gal78] Robert G. Gallager. *Information Theory and Reliable Communication*. John Wiley and Sons, Inc., New York, 1978.
- [Gen80a] Micheal R. Genesereth. Metaphors and models. In *AAAI-80*, pages 208–211, Stanford University, August 1980.
- [Gen80b] Dedre Gentner. *The Structure of Analogical Models in Science*. Technical Report 4451, Bolt, Beranek and Newman Inc., July 1980.
- [Gen80c] Dedre Gentner. Studies of metaphor and complex analogies. In *Symposium on Metaphor as Process*, A.P.A, Montreal, September 1980.
- [Gen83] Dedre Gentner. Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, 7(2), 1983.
- [Gen85] Micheal R. Genesereth. April 1985. Personal Communication.
- [GG83] Russell Greiner and Michael R. Genesereth. What's new? a semantic definition of novelty. In *IJCAI-83*, pages 450–54, Karlsruhe, Germany, August 1983.
- [GGGS80] Micheal Genesereth, Russell Greiner, Milton R. Grinberg, and David E. Smith. *MRS Manual*. December 1980. HPP Working Paper HPP-80-24 (Updated Dec 1984).
- [GL80] Russell Greiner and Douglas B. Lenat. A representation language language. In *AAAI-80*, pages 165–169, Stanford University, August 1980.
- [Gly80] Clark Glymour. *Theory and Evidence*. Princeton University Press, Princeton, 1980.

- [Gly83] Clark Glymour. Two programs for testing hypothesis of any logical form. In *Proceeding of the Second International Machine Learning Workshop*, pages 96–98, University of Illinois, Urgana-Champaign, 1983.
- [Goe85] Carl Gustav Hempel. In Philip W. Goetz, editor, *Encyclopædia Britannica*, page 828, Encyclopædia Britannica, Inc., Chicago, 1985.
- [Gri75] H. P. Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics*, pages 44–45, Academic Press, New York, 1975.
- [Har72] R. Harré, editor. *The Philosophies of Science: An Introductory Survey*. Oxford University Press, New York, 1972.
- [Hay85] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence: An International Journal*, 26(3):251–321, July 1985.
- [Hem65] Carl Gustav Hempel. *Aspects of Scientific Explanation and other Essays in the Philosophy of Science*. The Free Press, New York, 1965.
- [Hem66] Carl Gustav Hempel. *Philosophy of Natural Science*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1966.
- [Her64] I. N. Herstein. *Topics in Algebra*. Xerox College Publishing, Waltham, MA, 1964.
- [Hes66] Mary Hesse. *Models and Analogy in Science*. University of Notre Dame Press, Notre Dame, 1966.
- [Hes67] Mary Hesse. Models and analogies in science. In Paul Edwards, editor, *Encyclopædia of Philosophy*, pages 354–59, The Macmillan Company and The Free Press, New York, 1967.
- [Hin62] K. Jaako J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell Press, Ithica, 1962.

- [HM78] Frederick Hayes-Roth and John McDermott. An inference matching technique for inducing abstractions. *Communications of the ACM*, 21(5):401–411, May 1978.
- [HM82] Frank Halasz and Thomas P. Moran. Analogy considered harmful. In *Human Factors in Computer Systems*, National Bureau of Standards, Gaithersburg, Maryland, March 1982.
- [Hob83a] Jerry R. Hobbs. Metaphor interpretation as selective inferencing: cognitive process in understanding metaphor (part 1). *Empirical Studies of the Arts*, 1(1):17–33, 1983.
- [Hob83b] Jerry R. Hobbs. Metaphor interpretation as selective inferencing: cognitive process in understanding metaphor (part 2). *Empirical Studies of the Arts*, 1(2):125–142, 1983.
- [Hof84] Douglas Hofstadter. *The COPYCAT Project: An Experiment in Non-determinism and Creative Analogies*. Artificial Intelligence A. I. Memo 755, Massachusetts Institute of Technology, January 1984.
- [HR70] David Halliday and Robert Resnick. *Fundamentals of Physics*. John Wiley and Sons, Inc, New York, 1970.
- [Hru83] Poetics Today. 1983. Special Issue on “Metaphor”, Benjamin Hrushovski, editor, 4(2), Israel Science Publisher Ltd., Jerusalem.
- [Hum02] David. Hume. *An Enquiry Concerning Human Understanding*. Oxford Univeristy Press, Oxford, second edition, 1902. (Rewrite of Book I of “A Treatise of Human Nature”, 1758).
- [Kan82] Takeo Kanade. Vision. In Paul R. Cohen and Edward A. Feigenbaum, editors, *The Handbook of Aritificial Intelligence*, chapter XIII, pages 125–322, William Kaufman, Inc., Los Altos, CA, 1982.

- [Ked84] Smadar Kedar-Cabelli. *Analogy with Purpose in Legal Reasoning from Precedents*. Technical Report LRP-TR-17, Rutgers, Laboratory for Computer Science Research, July 1984.
- [Ked85] Smadar Kedar-Cabelli. Purpose-directed analogy: a summary of current research. In Thomas M. Mitchell, editor, *Proceeding of the Third International Machine Learning Workshop*, pages 80–83, Rutgers University, Skytop, Pennsylvania, June 1985.
- [Kim81] Scott Kim. *Inversions: A Catalog of Calligraphic Cartwheels*. BYTE Books, A division of McGraw-Hill, Peterborough, N.H., 1981.
- [Kli71] Robert E. Kling. A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178, 1971. Also IJCAI-71, British Computer Society, London, 1971.
- [Kow79] Robert Kowalski. *Logic for Problem Solving*. North Holland, Elsevier Science Publishing Company, Inc., New York, 1979.
- [Kri80] Saul Kripke. *Naming and Necessity*. Harvard Press, Cambridge, MA, 1980.
- [Kun82] Jeffrey R. M. Kunz, editor. *The American Medical Association Family Medical Guide*. Random House, Inc., New York, 1982.
- [Lam84] Leslie Lamport. *The L^AT_EX Document Preparation System*. February 1984. Second Preliminary Edition.
- [LE77] Victor R. Lesser and Lee D. Erman. A retrospective view of the hearsay-ii architecture. In *IJCAI-77*, pages 790–800, Massachusetts Institute of Technology, 1977.
- [Leb85] Michael Lebowitz. Complex learning environments: hierarchies and the use of explanation. In Thomas M. Mitchell, editor, *Proceeding of the Third International Machine Learning Workshop*, pages 110–112, Rutgers University, Skytop, Pennsylvania, June 1985.

- [Len82a] Douglas B. Lenat. AM: discovery in mathematics as heuristic search. In Randall Davis and Douglas B. Lenat, editors, *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill International Book Company, San Francisco, 1982. (Also PhD thesis, Stanford Report STAN-CS-76-570, July 1976).
- [Len82b] Douglas Bruce Lenat. The nature of heuristics. *Artificial Intelligence: An International Journal*, 19(2):189–249, 1982. (Also Stanford Technical Report HPP-80-26, Dec 1980).
- [Len83a] Douglas Bruce Lenat. EURISKO: A program that learns new heuristics and domain concepts. The nature of heuristics III: program design and results. *Artificial Intelligence: An International Journal*, 21(1-2):61–98, 1983.
- [Len83b] Douglas Bruce Lenat. The role of heuristics in learning by discovery: three case studies. In *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983.
- [Len84] Douglas B. Lenat. 1984. Personal Communication, related to Eurisko.
- [Lev77] S. Levin. *The Semantics of Metaphor*. The Johns Hopkins University Press, 1977.
- [Lit73] C. Scott Littleton. *The New Comparative Mythology: An Anthropological Assessment of the Theories of Georges Dumézil*. University of California Press, Los Angeles, 1973.
- [LJ80] George Lakoff and Mark Johnson. *Metaphors We Live By*. The University of Chicago Press, Chicago, 1980.
- [Lor74] K. Z. Lorenz. Analogy as a source of knowledge. In *Lez Prix Nobel en 1973*, pages 185–195, Elsevier, New York, 1974.
- [Low84] Micheal Lowry. The use of reformulation in computational geometry. 1984. Thesis Proposal.

- [MAE*62] J. McCarthy, P.W. Abrahams, D.J. Edwards, T. P. Hart, and M.I. Levin. *LISP 1.5 Programmer Manual*. MIT Press, Cambridge, MA, 1962.
- [Man49] D. G. Mandelbaum, editor. *Selected Writings of Edward Sapir*. University of California Press, 1949.
- [Mar67] Normal M. Martin. Rudolf Carnap. In Paul Edwards, editor, *Encyclopædia of Philosophy*, pages 25–33, The Macmillan Company and The Free Press, New York, 1967.
- [Mar84] A. P. Martinich. A theory for metaphor. *Journal of Literary Semantics*, 13:35–56, 1984.
- [McC79] John McCarthy. *Ascribing Mental Qualities to Machines*. AIM STAN-CS-79-725, Computer Science Department, Stanford University, March 1979.
- [McD79] John McDermott. Learning to use analogies. In *IJCAI-79*, pages 568–576, Tokyo, August 1979. (Also as CMU Tech Report).
- [McD80] John McDermott. R1: an expert in the computer systems domain. In *AAAI-80*, pages 269–271, Stanford University, Stanford, CA, August 1980.
- [Mel51] Herman Melville. *Moby Dick or A Whale*. Harper & Brothers Publishers, New York, 1851.
- [MG84] Jock Mackinlay and Micheal R. Genesereth. Implicit languages. In *AAAI-84*, pages 226–232, University of Texas at Austin, August 1984.
- [MH69] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie and B. Meltzer, editors, *Machine Intelligence 4*, Edinburgh University Press, Edinburgh, Scotland, 1969.

- [Mic83] Ryszard S. Michalski. A theory and methodology of inductive learning. In Ryszard S. Michalski, Jaime G. Carbonell, and Thomas M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983.
- [Mil00] John Stuart Mill. *A System of Logic*. Harper & Brothers Publishers, New York, 1900.
- [Min75] Marvin Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
- [Min85] Steven Minton. Overview of the prodigy learning apprentice. In Thomas M. Mitchell, editor, *Proceeding of the Third International Machine Learning Workshop*, pages 120–122, Rutgers University, Skytop, Pennsylvania, June 1985.
- [Mit79] Thomas M. Mitchell. Version spaces: a candidate elimination approach to concept learning. In *IJCAI-79*, pages 305–310, Massachusetts Institute of Technology, 1979.
- [Mit83] Thomas M. Mitchell. Learning and problem solving. In *IJCAI-83*, Karlsruhe, Germany, August 1983. Computers and Thought Award Lecture.
- [Mit85] Thomas M. Mitchell, editor. *Proceeding of the Third International Machine Learning Workshop*. Rutgers University, Skytop, Pennsylvania, June 1985.
- [MMS85a] Thomas M. Mitchell, Sridhar Mahadevan, and Louis I. Steinberg. Leap: a learning apprentice for VLSI design. In *IJCAI-85*, Los Angeles, 1985.
- [MMS85b] Thomas M. Mitchell, Sridhar Mahadevan, and Louis I. Steinberg. A learning apprentice for vlsi design. In Thomas M. Mitchell, editor, *Proceeding of the Third International Machine Learning Workshop*, pages 123–125, Rutgers University, Skytop, Pennsylvania, June 1985.

- [Moo80] Robert C. Moore. *Reasoning about Knowledge and Action*. Technical Note 191, SRI Interational, October 1980.
- [MP77] David Marr and T. Poggio. *A Theory of Human Stereo Vision*. AI Memo 451, Massachusetts Institute of Technology, 1977. Cambridge, Massachusetts.
- [MU77] Robert Moll and John Wade Ulrich. Program synthesis by analogy. *SIGART Newsletter*, 12(8):22-28, August 1977.
- [NA79] H. Penny Nii and N. Aiello. AGE (Attempt to GEneralize): a knowledge-based program for building knowledge-based program. In *IJCAI-6*, Tokyo, August 1979.
- [Nag61] Ernest Nagel. *The Structure of Science*. Harcourt, Brace & World, New York, 1961.
- [NB81] R. Nevatia and K. R. Babu. Linear feature extraction and description. In *IJCAI-81*, pages 639-641, British Columbia, 1981.
- [Nil80] Nils J. Nilsson. *Principles of Artificial Intelligence*. Tioga Press, Palo Alto, 1980.
- [NM73] Allan Newell and James Moore. How can Merlin understand. In L. W. Gregg, editor, *Knowledge and Cognition*, Erlbaum Associates, Potomac, Md., 1973.
- [Nov83] Gordon S. Novak Jr. Knowledge-based programing using abstract data types. In *AAAI-83*, pages 288-291, Washington, DC, August 1983.
- [NS72] Allan Newell and Herbert A. Simon. *Human Problem Solving*. Prentise-Hall, Englewood Cliffs, 1972.
- [Ort79] Andrew Ortony, editor. *Metaphor and Thought*. Cambridge University Press, Cambridge, 1979.

- [Pai79] Allan Paivio. Psychological processes in the comprehension of metaphor. In Andrew Ortony, editor, *Metaphor and Thought*, pages 150–171, Cambridge University Press, Cambridge, 1979.
- [Pla61] Plato. The republic. In Edith Hamilton and Huntington Cairnos, editors, *The Collected Dialogues of Plato*, pages 574–844, Bollingen Series, Pantheon Books, 1961. Translated by Paul Shorey.
- [Pol54] George Polya. Induction and analogy in mathematics. In *Mathematics and Plausible Reasoning*, Princeton University Press, Princeton, 1954.
- [Pol57] George Polya. *How to Solve It: A New Aspect of Mathematical Method*. Princeton University Press, Princeton, second edition, 1957.
- [Py179] Zenon W. Pylyshyn. Metaphorical imprecision and the ‘top-down’ research strategy. In Andrew Ortony, editor, *Metaphor and Thought*, pages 420–436, Cambridge University Press, Cambridge, 1979.
- [Qui60] Wilhelm Quine. *Word and Object*. MIT Press, Cambridge, Massachusetts, 1960.
- [Red79] Micheal J. Reddy. The conduit metaphor — a case of frame conflict in our language about language. In Andrew Ortony, editor, *Metaphor and Thought*, pages 284–324, Cambridge University Press, Cambridge, 1979.
- [Rei78] Raymond Reiter. Deductive question-answering on relational data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 149–177, Plenum Press, New York, 1978.
- [RG77] R. B. Roberts and Ira P. Goldstein. *FRL Users Manual*. Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts, 1977. A.I. Memo 408.
- [Ric36] I. A. Richards. Metaphor. In I. A. Richards, editor, *The Philosophy of Rhetoric*, Oxford University Press, London, 1936.

- [Ric79] Charles Rich. *A Library of Programming Plans with Applications to Automated Analysis, Synthesis and Verification of Programs*. Technical Report, MIT, 1979.
- [Ric81] Charles Rich. *Inspection Methods in Programming*. PhD thesis, MIT, June 1981.
- [Ric83] Alex Rich. Research news: Professor Alex Rich and Z DNA. *Science*, 222:496, November 1983.
- [RN82] Paul S. Rosenbloom and Allan Newell. Learning by chunking: summary of a task and a model. In *AAAI-82*, August 1982.
- [Rob85] Frank Neville H. Robinson. Electricity and magnetism. In Philip W. Goetz, editor, *Encyclopædia Britannica*, pages 201-292, Encyclopædia Britannica, Inc., Chicago, 1985.
- [Ros73] E. Rosch. On the internal structure of perceptual and semantic categories. In T. E. Moore, editor, *Cognitive Development and the Acquisition of Language*, Academic Press, New York, 1973.
- [Ros83] Paul S. Rosenbloom. *The Chunking of Goal Hierarchies: A Model of Practice and Stimulus-Response Compatibility*. PhD thesis, Carnegie-Mellon University, August 1983.
- [Rus76] S. W. Russell. Computer understanding of metaphorically used verbs. *American Journal of Computational Linguistics*, 1976. microfiche 44.
- [Rus85] Stuart Russell. *The Compleat Guide to MRS*. June 1985. Stanford KSL Report HPP-85-12.
- [RW80] Brian K. Reid and Janet H. Walker. *SCRIBE: Introductory User's Manual*. third edition, May 1980. Preliminary Draft.
- [RW81] Charles Rich and Richard C. Waters. *Abstraction, Inspection and Debugging in Programming*. AI Memo 634, Massachusetts Institute of Technology, June 1981.

- [SA77] Roger C. Shank and Robert P. Abelson. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.
- [Sch85] Roger Schank. Questions and explanations. In Thomas M. Mitchell, editor, *Proceeding of the Third International Machine Learning Workshop*, Rutgers University, Skytop, Pennsylvania, June 1985. (Based on invited presentation, not appearing in the proceedings.).
- [Sea79] John R. Searle. Metaphor. In Andrew Ortony, editor, *Metaphor and Thought*, pages 92–123, Cambridge University Press, Cambridge, 1979.
- [SF80] Reid G. Smith and Peter Friedland. *Unit Package User's Guide*. December 1980. (HPP-80-28, and Defense Research Establishment Atlantic # 80/L).
- [SG85] David E. Smith and Micheal R. Genesereth. Ordering conjunctive queries. *Artificial Intelligence: An International Journal*, 26(2):171–215, May 1985.
- [SGLS80] Gerald Jay Sussman and Jr. Guy Lewis Steele. Constraints — a language for expressing almost-hierarchical descriptions. *Artificial Intelligence: An International Journal*, 14:1–39, 1980.
- [Sha72] William Shakespeare. *The Complete Signet Classic Shakespeare*. Harcourt Brace Jovanovich, Inc., San Francisco, 1972.
- [She84] Roger Shepard. Ecological constraints on internal representation: resonant kinematics of perceiving, imaging, thinking, and dreaming. *Psychological Review*, 91(4), October 1984. Expanded version of the third James J. Gibson Memorial Lecture, presented at Cornell University on 21 October 1983.
- [Sho84] Edward H. Shortliffe. 1984. Personal Communication, related to Oncocin.

- [Shr79] Howard E. Shrobe. *Dependency Directed Reasoning for Complex Program Understanding*. PhD thesis, MIT, April 1979.
- [Sim83] Herbert A. Simon. Why should machines learn? In Ryszard S. Michalski, Jaime G. Carbonell, and Thomas M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983.
- [Smi68] Norman Kemp Smith. *A Commentary on Kant's 'Critique of Pure Reason'*. Macmillan & Company, Limited, Portway, Bath, 1968.
- [SS77] Richard M. Stallman and Gerald Jay Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence: An International Journal*, 9(2):135-196, 1977.
- [SSB*81] Edward H. Shortliffe, A. C. Scott, M. B. Bischoff, William van Melle, and C. D. Jacobs. Oncocin: an expert system for oncology protocol management. In *IJCAI-81*, pages 876-881, International Joint Conferences on Artificial Intelligence, August 1981. Vancouver, B.C.
- [Sus75] G. J. Sussman. *A Computer Model of Skill Acquisition*. American Elsevier, New York, 1975.
- [Swa81] William R. Swartout. *Producing Explanation and Justification of Expert Consulting Programs*. PhD thesis, Massachusetts Institute of Technology, January 1981.
- [Tap81] Steve Tappel. Examples of reformulation. 1981. Thesis Proposal.
- [Tar52] Alfred Tarski. The semantic conception of truth and the foundations of semantics. In L. Linsky, editor, *Semantics and the Philosophy of Language*, pages 13-47, University of Illinois Press, Urbana, 1952.
- [Thi68] Christian Thiel. *Sense and Reference in Frege's Logic*. D. Reidel Publishing Company, Dordrecht-Holland, 1968.

- [Thi77] Robert Thibadeau. *Reaching (For) an Understanding about Analogy*. Department of Computer Science DCS-TM-8, Rutgers, May 1977.
- [Utg84] Paul E. Utgoff. *Shift of Bias for Inductive Concept Learning*. PhD thesis, Rutgers, Laboratory for Computer Science Research, October 1984.
- [van85] Kurt vanLehn. 1985. Personal Communication, during Induction Session at IML-85.
- [vB79] Kurt van Lehn and John Seely Brown. Planning nets: a representation for formalizing analogies and semantic models of procedural skills. In R. E. Snow, P. A. Frederico, and W. E. Montague, editors, *Aptitude learning and instruction: Cognitive process and analyses*, Lawrence Erlbaum Associates, Hillsdale, 1979.
- [vG82] Anne vdl Gardner. Search. In Avron Barr and Edward A. Feigenbaum, editors, *The Handbook of Aritificial Intelligence*, chapter II, pages 19-140, William Kaufman, Inc., Los Altos, CA, 1982.
- [vGDB81] Anne v.d.L. Gardner, James E. Davidson, and Avron Barr. Understanding natual language. In Avron Barr and Edward A. Feigenbaum, editors, *The Handbook of Aritificial Intelligence*, chapter IV, pages 223-321, William Kaufman, Inc., Los Altos, CA, 1981.
- [vL85] Kurt van Lehn. Learning procedures one disjunct at a time. *Artificial Intelligence: An International Journal*, 1985. In press.
- [Wal75] David Waltz. Generating semantic descriptions from drawings of scenes with shadows. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*, chapter 2, McGraw-Hill Book Company, New York, 1975. (Also, Project MAC Technical Report AI-TR-271).
- [Wat71] Richard C. Waters. *Automatic Analysis of the Logical Structures of Programs*. PhD thesis, MIT, December 1971.

- [WBKL83] Patrick H. Winston, Thomas O. Binford, Boris Katz, and Micheal Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *AAAI-83*, pages 433-39, Washington, DC, August 1983.
- [Wey78] Richard W. Weyhrauch. *Prolegomena to a Theory of Formal Reasoning*. Technical Report AIM-315, Stanford University, December 1978.
- [Wil83] Robert Wilensky. *Planing and Understanding*. Addison Wesley, Reading, MA, 1983.
- [Win79] Patrick H. Winston. *Learning and Reasoning by Analogy: The Details*. Artificial Intelligence AIM 520, Massachusetts Institute of Technology, April 1979. Revised from June 1979.
- [Win80] Patrick H. Winston. Learning and reasoning by analogy. *Communications of the ACM*, 23(12):689-703, December 1980.
- [Win81] Patrick H. Winston. *Learning New Principles from Precedents and Exercises: The Details*. Artificial Intelligence AIM 632, Massachusetts Institute of Technology, November 1981. Revised from May 1981.
- [Win82] Patrick H. Winston. *Learning by Augmenting Rules and Accumulating Censors*. Artificial Intelligence AIM 678, Massachusetts Institute of Technology, September 1982. Revised from May 1982.
- [Wit53] Ludwig Wittgenstein. *Philosophical Investigations*. Macmillian, New York, 1953.
- [WK84] Joe Weening and Arthur Keller. Dover fonts as of October 15, 1984. October 1984. (File found on {SAIL} GRANTF.TEX[TEX,SYS]).
- [WR82] Rajendra S. Wall and Edwina L. Rissland. Scenarios as an aid to planning. In *AAAI-82*, pages 193-196, Washington, DC, 1982.

[Zie77] Olgierd Cecil Zienkiewicz. *The Finite Element Method, Third Edition*.
McGraw Hill Book Company, Limited, London, 1977.

Appendix A

Notes from the Text

“The first rule of style is to have something to say. The second rule of style is to control yourself when, by chance, you have two things to say; say first one, then the other, not both at the same time.”

How to Solve It, Polya (1957)

A.1 Notes from Chapter 1

Note:1-1. Other Examples of Analogies: (from page 3)

This note lists a variety of other examples of analogies, to embellish the entries shown on page 2.

- **Strategies:** Many game-playing plans and strategies carry over from checkers to chess, and even to less related games, like blackjack (see [Bro79] and [Car83a]).
- **Programming:** Recall how much easier it was to write the second recursion program than it was to write the first.
- **History of Science:** Professor Yanosky used his knowledge of the “trp” operon to explain one aspect of the observed behavior of the “his” operon ([Fri84]). The DNA translation process can be regarded as a railroad train on a track (see [Ric83]). Taking a different slant, Lenat discusses how one might view DNA as a program [Len83b].

- **Medical Findings:** It is often easier to understand a second medical treatment once one had grasped the nuances of a prior, related case. For example, several protocols require that a drug be administered periodically, but in attenuated dosages ([SSB*81,Sho84]).
- **Artifacts (e.g., names):** Another researcher correctly guessed that the name of the bold-face extended font associated with the computer modern family would be CMBX10, basing his conjecture on the observations that the computer modern roman font was CMR10, and that the name of the almost computer modern bold-face extended font, AMBX10, was derived from the related roman font, AMR10, by changing the embedded “R” (for Roman) to “BX” (for Bold face eXtended). ([WK84, p4] later confirmed this assumption.)

Note:1-2. “Correct” Analogies:

“It would be foolish to regard the plausibility of such [analogically inferred] conjectures as certainty, but it would be just as foolish, or even more foolish, to disregard such plausible conjectures.”

How to Solve It, Polya (1957)

(from pages 9, 11, 43, 65, 135, 166 and A-49)

An analogical inference suggests that certain new conjectures be added to a theory. Its goal, like that of any learning process, is to add *correct* conjectures. This note asks what it means to claim that some analogically-inferred conjecture is correct. This discussion has several parts. First, it presents two reasons why the standard goal of “semantically correct” information is inappropriate here. Second, it defines and discredits another possible meaning of correctness, one based on conversational goals. This leaves a purely syntactic measure of correctness, which is discussed next. Finally, this note points out that this “concession” is neither that major nor that limiting.

As the first proposed criterion of correctness, perhaps we should insist that these new conjectures be semantically correct in the standard Tarskian sense

[Tar52]: an analogical inference is considered correct only if its conjectures actually correspond to the world. Hence, we want *NLAG* to propose that “Snow is white” only when, in fact, snow is white.

There are two objections to this correctness criterion. First, it is not obtainable. *NLAG* has only representations of the objects in the world, not the objects themselves. Since it lacks direct access to the outside world itself, it is restricted to syntactic manipulations. This means that such a “real world semantics” is impossible.¹

Secondly, it is not always desirable to generate only semantically correct statements. For example, given the “Snow is like this Ice Cream” analogical hint, a sufficiently naïve learner might reasonably conjecture that snow is green, as the speaker was referring to mint ice cream. In fact, in a sufficiently convoluted situation, this “Snow is Green” conjecture might even be the intent of the speaker.

If this example seems implausible, consider closely the “FlowRate is like Current” analogy discussed throughout this dissertation. The desired conclusion includes the claim that FlowRate obeys Ohm’s Law. This, however, is not quite true. Once turbulent flow is considered, FlowRate is proportional to PressureDrop to an empirically derived k^{th} power, where $0.5 < k < 0.7$ [Coc80,Zie77].² The simple “hydraulic Ohm’s Law” — pressure equals flowrate times pipecharacter — is still considered a useful “fact” to learn and this analogical hint, an appropriate means for communicating it.

The gist of this second critique is that the purpose of an analogy is often conversational and, while the speaker’s intended message is often a semantically correct fact, there is no reason to force this. Hence, the desired message need not be semantically valid. (This is related to various linguistic views of pragmatics, including the Barwise/Perry theory of conversational semantics [BP83],

¹In particular, the *NLAG* system does not go out, buy the needed pipes, mill them to size, hook them together, and measure the flow-rate...

²See also Equation A.12 in Note:4-1.

Searle's theory of speech acts [Sea79] and Grice's notion of conversational postulates [Gri75].)

The points above argue against the use of a Tarskian form of semantics as a criterion for determining the correctness of analogy. (Subsection 8.2.1 presents yet another argument, explaining that this framework is inappropriate for describing a learning process.) The second argument above suggests a different form of semantics, one based on conversational standards. Here, an analogy is judged correct if it conveys the message the speaker intended, independent of whether or not this corresponds to the real world.

The argument against "conversational semantics" is similar to the first one used to defeat Tarskian semantics: the *NLAG* system cannot read the speaker's mind. Of course, it can still use its knowledge of the speaker's knowledge and its estimates of his accuracy, *when such information is included in its initial theory*. This then reduces to syntactic considerations. Beyond this observation, this dissertation does not explicitly address this collection of issues.³

This leads to our third proposal: a syntactic criterion for correctness. We insist that any conjecture generated be "reasonable"; that the answer be justifiable in terms of the hint, problem, and current knowledge base. The hardest pruning rule we can offer along these lines is the **Consistent** criterion shown in Section 2.3's Figure 3-1. Hence, the "correctness" objective reduces to the requirement that the analogy provide a *consistent extension* to the theory.

The upshot of this analysis is that we cannot expect *NLAG* to derive only facts which are semantically or conversationally correct. The final point of this note argues that this is not a major sacrifice.

First, view *NLAG* as a Knowledge Acquisition program. The objective of capturing only correct information is seldom met by such Expert System building

³I avoid this for several reasons. The most obvious is that it represents a major digression from my *analogy* theme. Also, it invariably leads to the issue of pragmatics: what did the speaker *really* mean when he gave the analogical hint. This requires understanding what the speaker knew of the learner's knowledge, which might, in turn, include the learner's knowledge of the speaker, and so on... Of course, we already saw that our model can still consider such issues; we need only encode them as additional information within the learner's theory and reason about them accordingly.

tools anyway, for both unavoidable reasons (e.g., because the correct answer is simply not known or the teacher is ignorant) as well as avoidable ones (e.g., based on efficiency considerations, both with respect to the Knowledge Acquisition process and to the eventual performance system).

The second realization is that people, too, often misunderstand analogies, even when the teacher was trying to communicate a semantically correct fact. Consider again the scenario given in Section 1.3. Given a naïve understanding of hydraulics, it seems quite reasonable to use each pipe's cross-section, rather than its pipe-characteristic, as the constant of proportionality. Here, once you "get" the analogy, you would believe that the pressure drop is the product of the flow rate through the pipe and its cross section.⁴

Notice, however, that few people are not bothered by this realization: most consider the analogical inference process inherently a method of *plausible reasoning*, for proposing *likely* sentences.

A.2 Notes from Chapter 2

Note:2-1. Why Consider Disjunctions and Negations?

(from page 17)

Equation 2.1's definition of analogical inference places no type restriction on what might qualify as an analogy formula. This means \vdash permits disjunctions and negations in addition to conjunctions of positive literals. This can lead to strained examples of analogy formulae, e.g.,

$$\varphi_N(x) \Leftrightarrow \neg(x = \text{Fred}) \quad (\text{A.1})$$

$$\varphi_O(x) \Leftrightarrow x = B \vee \text{Unlikely}(x). \quad (\text{A.2})$$

(In fact, we could use $\varphi_{or}(x) \Leftrightarrow R_1(x) \vee R_2(x)$ for almost any relations R_1 and R_2 , insisting only that $R_1(B)$ holds and that R_2 is not known to be false with

⁴This is, of course, wrong: wider pipes should exhibit less pressure-drop than thinner pipes, not more. See Run#2a-4 in Section 7.3.

respect to A . Such a φ_{or} formula would qualify even if $R_1(A)$ is provable false — $Th \models \neg R_1(A)$. See also Note:2-7.)

I did consider restricting the allowable analogy formulae, to permit only conjunctions of positive literals. This is not enough, as every other researcher in the field of learning has noticed. For example, we need the $\neg(x = \text{Fred})$ clause to explain that Alice is like Beth in that each is a “non-Fred” occupant of a room. An example of disjunction occurs when we want to say that Bird1 is like Bird2 in that each is either a Robin or a Cardinal. (This could arise in a lexically impoverished situation, in which we lack the terms needed to describe the class of Red-Birds.)

This dissertation discusses two ways of sidestepping this issue. First, the Abstraction restriction (discussed in Chapter 4) renders it academic. Secondly, Subsection 9.2.2 defines a quantitative measure (labeled *specificity*) which gives low ratings to the more counterintuitive (read “less likely”) of these cases.

Note:2-2. Types of Learning:

(from pages 165, 217 221 and 224)

The goal of the most analogy systems is to acquire new base-level facts. In particular, the resulting knowledge base contains more facts after the analogical inference, where these additional facts were not derivable from the initial knowledge base. This means there may be problems which the resulting system can solve, but the initial system could not. (This says nothing about how long it may take to arrive at this solution; indeed, the timing behavior of this system may degrade for other problems. Here, we are considering only functionality — whether or the knowledge base contains enough information to solve a particular problem.)

The rest of this note focuses on learning systems in general (as opposed to our general concentration on only the specialized form of *learning by analogy*). We define consider any system which adds such new facts to be a $Learn_1$ system.

It is easy to express this condition formally: Start with a complete deductive system which “knows” the facts in the theory Th , we say that this system has

$Learn_1$ ed a new fact, σ , if $Th \not\models \sigma$. Of course, we do not want our system to learn inconsistent facts, we only consider learning σ if $Th \not\models \neg\sigma$.

The astute reader will recognize these as the **Unknown** and **Consistent** criteria, where $\sigma = \varphi(A)$. (See Equation 2.1.) This means our model of general analogical inference, \vdash , is a $Learn_1$ process. (Its **Common** condition means it is an analogy system as well.)

There are other types of learning systems. Another possible objective is increased efficiency: to enable a fixed inference engine to solve a particular problem in fewer steps. Here, the overall system can still solve the same collection of problems, but after the learning step, it can reach the (same) answers faster. (I think of such systems as “re-arranging” the knowledge base, rather than adding to it.) We consider such systems to be $Learn_2$ systems.⁵

These $Learn_2$ systems are clearly important: Once we have coded the rules of chess, nothing more can be $Learn_1$ ed. We still consider a learning program a success if it can learn to play better. (The purpose of this note is not to downgrade such programs, but only to characterize them as $Learn_2$ systems.)⁶

The chunking work [RN82] falls under this $Learn_2$ rubric, as does the LEAP program [MMS85a] and many recent works in explanation based learning [Mit85]. In the area of analogical reasoning, Kling’s Zorba program, designed to use an analogical connection to shorten proofs, is a $Learn_2$ process [Kli71], as is Carbonell’s work on finding T-space operators [Car81a].

There are connections and trade-offs between these two types of learning. One connection which blurs the distinction between these two types is the realization that few systems use a deductively complete inference system. Hence, the learning system can propose an entailed proposition which this incomplete deductive process would never have derived. While this is clearly a $Learn_2$ step (as it added a derivable proposition), it may appear to be a $Learn_1$ step since this proposition

⁵Dietterich calls this ($Learn_2$) knowledge level learning, which he distinguishes from non-deductive learning ($Learn_1$) [Die85].

⁶Of course, many of these systems do acquire new information about *chess playing strategies*, rather than about chess itself. Hence, such systems are $Learn_1$ ing facts in the “meta-space above chess”.

is outside the space of derived propositions. (Perhaps we should define these as “*Learn*_{1.5}” systems?)

A related problem arises when a problem is “sufficiently intractable” that the complete deductive system would not find the answer in any reasonable amount of time. (Recall the chess example mentioned above.) Here, again, are situations where a learning system produces a derivable answer that, for all practical purposes, would never have been found.

As an example of a trade-off between these variants of learning, the performance of many (deductively-complete) systems degrades as new facts are entered, meaning that a *Learn*₁ step may have a negative effect with respect to *Learn*₂’s efficiency measure.

Note:2-3. Why not Existentially Bound Unary Formulae:

(from page 17 and 195)

The simple definition of analogical inference shown in Equation 2.1 does not include the **NonTrivial** constraint. This is because the **Unknown** condition ($Th \not\models \varphi(A)$) means that φ ’s domain is not universal ($\exists s Th \not\models \varphi(s)$) when the analogy formula φ is unary; this is sufficient to guarantee that the unary φ is **NonTrivial**. This claim is not true when considering non-unary formulae, explaining why Figure 2-1 needs an explicit **NonTrivial** constraint. This observation suggests that we might “existentially quantify away” $\varphi(x_1, \dots, x_n)$ ’s other $n - 1$ arguments, and simply use the unary $\varphi|_i(x)$. This note argues against doing so.

The problem is that a given formula may have more than one instantiation involving the target analogue, and the $\varphi|_i$ existential form merges all of these instances together. For example, imagine we want to infer $\text{Group}(\mathcal{R}, +, 0)$ from the hint “+ is like *”, via the fact $\text{Group}(\mathcal{R}_0, *, 1)$ (i.e., $Th_{+*} \models \text{Group}(\mathcal{R}_0, *, 1)$).

To use Equation 2.1, we must use the analogy formula $\text{Group}|_2(x)$: i.e., we would want to conjecture $\text{Group}|_2(+)$. This is problematic if the initial theory

included another instance of this $\text{Group}|_2$ relation which involved $+$: e.g., when

$$Th_{+*} \models \text{Group}(Z, +, 0). \quad (\text{A.3})$$

Since Equation A.3 means that $Th_{+*} \models \text{Group}|_2(+)$, we see that $\text{Group}|_2(+)$ does not pass Equation 2.1's **Unknown** test. This prevents us from analogically inferring $\text{Group}(\mathfrak{R}, +, 0)$.

One possible solution would be to use different relations: e.g., **Group1** and **Group2**. Here **Group2** could state the provable form, $\text{Group2}(Z, +, 0)$. This leaves $\text{Group1}|_2$ free, as $Th_{+*} \not\models \text{Group1}|_2(+)$ still holds (even though $Th_{+*} \models \text{Group2}|_2(+)$). This means that \vdash allows $\text{Group1}(\mathfrak{R}, +, 0)$ to be analogically inferred as desired, i.e.,

$$Th_{+*}, + \sim * \vdash \text{Group1}(\mathfrak{R}, +, 0).$$

(Of course, this assumes that $Th_{+*} \models \text{Group1}(\mathfrak{R}_0, *, 1)$; it makes no difference whether or not $\text{Group2}(\mathfrak{R}_0, *, 1)$ is in Th_{+*} .)

Unfortunately, the **Abstraction** constraint shown in Chapter 4 means that we do not get to choose the vocabulary: we have to abide with the relations explicitly given. (Besides, this trick would not work if our theory included the obvious $\text{Group1} \Leftrightarrow \text{Group2}$ connection.)

This forces \vdash to deal with n -ary formulae, which necessitates the explicit $\neg\text{Trivial}(\varphi|_i, Th)$ check included in Figure 2-1. The discussion leading to Definition 1, as well as all of Section 2.4, explain this constraint.

Note:2-4. Theory-Based Definition of Analogy, Analogy_T :

“Merohedral isomorphism may be considered as [a] very precise sort of analogy.”

How to Solve It, Polya (1957)

(from pages 22, 185, A-18 and A-39)

In Section 1.3's fluid dynamics example, we implicitly assumed that an analogy is a correspondence between the facts about the two analogues. This is slightly

different from the “common formula” conceptualization defined in Section 2.3. This note restates this “correspondence of facts” intuition more formally, in terms of the $Analogy_T$ relation. It also demonstrates that $Analogy_T$ is equivalent to the $Analogy_F$ relation defined in Definition 2.

As with $Analogy_F$, this $Analogy_T$ relation is syntactic and with respect to a particular theory, Th . (In our standard example, it uses the theory Th_{CF} which includes many sentences about Current and few sentences about FlowRate.) We claim that A and B are analogous with respect to Th if some of Th 's facts about A directly correspond to a related set of Th 's facts about B — i.e., if some mapping takes each sentence of P_A into a sentence of P_B , where P_A is a set of sentences which is about A in the same manner that P_B is about B.

Many (in fact, most) of the possible sentence-to-sentence mappings produce meaningless results. We are only interested in systematic mappings, which take one set of interrelated sentences (which all deal with one analogue) into a similarly interrelated set of sentences (dealing with the other). Hence, the P_B to P_A mapping can be viewed as a sentence-to-sentence mapping, induced by a symbol-to-symbol mapping. That is,

| | |
|---|-------|
| <p>Any analogy can be expressed as an isomorphism — as a symbol-to-symbol-based mapping between a pair of (partial) theories.</p> | (A.4) |
|---|-------|

This condition is both necessary and sufficient, once the analogues are represented appropriately. (Subsection 9.3.2 describes the problems in seeking an analogical connection between a pair of uncoöperative representations.)

As an example, the correspondence discussed in Section 1.3 mapped P_C (the EC facts known as Ohm's and Kirchoff's Laws) into the new FS facts P_F . It is derived from the symbol-symbol mapping S_{CF} shown in Figure A-1.

Definition 33 formally expresses the claim that *an analogy is a sentence to sentence mapping induced by a symbol to symbol mapping, which takes a particular*

$$\begin{array}{l}
 S_{CF} = \left\{ \begin{array}{l} \text{Current} \mapsto \text{FlowRate}, \text{ VoltageDrop} \mapsto \text{PressureDrop}, \\ \text{Resistance} \mapsto \text{PipeCharacter}, \text{ Resistor} \mapsto \text{Pipe} \end{array} \right\} \\
 P_C = \left\{ \begin{array}{l} \forall j \sum_{p:Conn(p,j)} \text{Current}(j,p) = 0 \\ \forall loop \sum_{\langle i,j \rangle \in loop} \text{VoltageDrop}(i,j,[x]) = 0 \\ \forall d \text{ Resistor}(d) \Rightarrow [\text{Current}(j_d^1, d) + \text{Current}(j_d^2, d) = 0] \\ \forall d \text{ Resistor}(d) \Rightarrow [\text{VoltageDrop}(j_d^1, j_d^2, [d]) = \text{Current}(j_d^1, d) * \text{Resistance}(d)] \end{array} \right\} \\
 P_F = \left\{ \begin{array}{l} \forall j \sum_{p:Conn(p,j)} \text{FlowRate}(j,p) = 0 \\ \forall loop \sum_{\langle i,j \rangle \in loop} \text{PressureDrop}(i,j,[r]) = 0 \\ \forall d \text{ Pipe}(d) \Rightarrow [\text{FlowRate}(j_d^1, d) + \text{FlowRate}(j_d^2, d) = 0] \\ \forall d \text{ Pipe}(d) \Rightarrow [\text{PressureDrop}(j_d^1, j_d^2, [d]) = \text{FlowRate}(j_d^1, d) * \text{PipeCharacter}(d)] \end{array} \right\}
 \end{array}$$

Figure A-1: Desired Analogy Mapping, $M = \langle S_{CF} P_C P_F \rangle$

set of sentences about B into a related set of sentences about A.⁷

Definition 33 $Analogy_T(A, B, Th, \langle S P_A P_B \rangle)$

$$\begin{aligned}
 &\iff TermMap(S) \ \& \ S(B) = A \\
 &\ \& \ Th \models P_A \ \& \ Th \models P_B^8 \\
 &\ \& \ \forall f_B \in P_B \ \exists f_A \in P_A \ f_A = S[f_B] \\
 &\ \& \ \neg Trivial_T(B, P_B, Dom[S]; Th)
 \end{aligned}$$

where $TermMap(m)$ means that m is a function which maps terms to terms and $Dom[f]$ refers to the domain of the function f . The $Trivial_T(x, P, \Lambda, Th)$ relation corresponds to the earlier $Trivial$ and means that the axioms P are not about the concept x . For now, it is sufficient to consider $Trivial_T(x, P, \Lambda, Th)$ to mean that x 's mention in P is tautological, with respect to the theory Th and to the terms allowed to vary from one analogue to the other, Λ . (See Definition 34.)

⁷Hesse calls such mappings "formal analogies"; see [Hes67].

⁸Point NotStd2 in Note:2-5 shows that $P_A \subset Th$ is not sufficient.

We refer to this overall $\langle S P_A P_B \rangle$ triple as the mapping M . In a slight abuse of notation, we allow S to refer to the obvious sentence-to-sentence mapping induced by the underlying symbol-symbol mapping, or even the extension which deals with sets of sentences — hence, $P_A = S[P_B]$.⁹ This means that the pair $\langle S P_B \rangle$ is sufficient to define a unique M . Section 9.3 demonstrates that each of these semi-independent parameters must be specified precisely.

We still have to define $Trivial_T$. Based on the arguments given in Section 2.4, a term x appears trivially in a set of facts P (with respect to the set of terms allowed to vary, Λ , and a background theory, Th), if we can prove (within Th) that *any* term could appear in x 's place, for some “setting” of those varying terms, Λ .

Definition 34 $Trivial_T(x, P, \Lambda, Th) \iff$

$$\forall s \exists \mathcal{R} \left[\begin{array}{l} TermMap(\mathcal{R}) \ \& \ \mathcal{R}(x) = s \ \& \ Dom[\mathcal{R}] = \Lambda \ \& \\ Th \models \mathcal{R}[P] \end{array} \right]$$

The balance of this note proves that $Analogy_T$ is equivalent to $Analogy_F$, and provides a quick comparison of these definitions.

Proof that $Analogy_T$ is Equivalent to $Analogy_F$:

This proof uses, as a lemma, the equivalency of their respective *Triviality* conditions. That second proof follows.

$$Analogous_T(A, B, Th) \iff Analogous_F(A, B, Th)^{10} \quad (A.5)$$

\implies : (Given $Analogy_T(A, B, Th, \langle S P_A P_B \rangle)$, find a corresponding $Analogy_F(A, B, Th, \langle \varphi [a_1, \dots, a_n] [b_1, \dots, b_n] \rangle)$)

⁹By convention, B (and hence P_B) refers to the source analogue, and A , to the target analogue. This explains why the S mapping may appear backwards.

¹⁰By convention, $Analogous_\chi$ is the “exists-form” of the corresponding $Analogy_\chi$ — i.e., $Analogous_\chi(A, B, Th) \iff \exists \Sigma Analogy_\chi(A, B, Th, \Sigma)$. (This repeats the comment in Point CA2 on page 58.)

Begin by writing P_B as a single conjunction, $\varphi_B = \bigwedge_{\rho \in P_B} \rho$. Then let the formula $\varphi(x_1, \dots, x_n)$ be the “variabilized form” of φ_B , formed by replacing each occurrence of the the j^{th} member in the S ’s domain, $b_j \in \text{Dom}[S]$, with the (assumed currently unused) variable $?_j$ in this φ_B : i.e., $\varphi = \varphi_B[b_1/?_1, \dots, b_n/?_n]$.¹¹ By construction, this φ_B (and hence P_B) is equivalent to $\varphi(b_1, \dots, b_n)$. Similarly, P_A is equivalent to $\varphi(a_1, \dots, a_n)$, where a_j is the j^{th} term in S ’s range. This means we can construct an *Analogy_F* triple $\langle \varphi [a_1, \dots, a_n] [b_1, \dots, b_n] \rangle$ from any *Analogy_T* triple $\langle S P_A P_B \rangle$.

\Leftarrow : Given *Analogy_F*($A, B, Th, \langle \varphi [a_1, \dots, a_n] [b_1, \dots, b_n] \rangle$), let $S = \{a_j \mapsto b_j\}_j$, $P_A = \{\varphi(a_1, \dots, a_n)\}$ and $P_B = \{\varphi(b_1, \dots, b_n)\}$.

Now to prove the required lemma: Using the $\langle \varphi \ i \rangle$ to $\langle B P_B \text{ Dom}[S] \rangle$ correspondence defined above, we show that

$$\text{Trivial}(\varphi|_i, Th) \iff \text{Trivial}_T(B, P_B, \text{Dom}[S], Th) \quad (\text{A.6})$$

\Rightarrow : $\neg \text{Trivial}_T(B, P_B, \text{Dom}[S], Th)$ guarantees that there is some constant s such that, $Th \not\models \mathcal{R}[P_B]$ for any term-term mapping \mathcal{R} where $\mathcal{R}(B) = s$. Letting $[c_1, \dots, s, \dots, c_n]$ be the range of any such \mathcal{R} , we see that $Th \not\models \varphi(c_1, \dots, s, \dots, c_n)$, using the φ defined in the proof above. Since this holds for any set $\{c_j\}_{j \neq i}$, we see that $\neg \text{Trivial}(\varphi|_i, Th)$, as desired.

\Leftarrow : Given $\neg \text{Trivial}(\varphi|_i, Th)$, there is an s such that $Th \not\models \varphi(z_1, \dots, s, \dots, z_n)$ for any set $\{z_j\}_{j \neq i}$. Using $\text{Dom}[S] = \{b_j\}_{j \neq i} \cup \{B\}$, define \mathcal{R} as $\{b_j \mapsto z_j\}_{j \neq i} \cup \{B \mapsto s\}$ for any such instantiation. Observe now that $\mathcal{R}[B] = s$ and $Th \not\models \mathcal{R}[P_B]$. This demonstrates the existence of the constant s needed to establish $\neg \text{Trivial}_T(B, P_B, \text{Dom}[S], Th)$.

□

As a quick summary: we have defined two equivalent syntactic definitions of analogy: *Analogy_F* is based on a common Formula, and *Analogy_T*, on an

¹¹The simple notation $\phi[x/y]$ means that the term x is lexically replaced by the term y throughout the formula ϕ . It extends to the n -ary form, $\phi[x_1/y_1, \dots, x_n/y_n]$, in the obvious manner.

interrelated pair of Theories (read “collection of sentences”). Both use the full deductive closure of the theory, rather than just the facts explicitly included. As both P_B and $\varphi(b_1, \dots, B, \dots, b_n)$ refer to the same collection of facts, they are used interchangeably as the *source of the analogy*. (Similarly P_A and $\varphi(a_1, \dots, A, \dots, a_n)$ each encode the same *target of the analogy*.) Section 8.3 provides a corresponding Semantic definition, $Analogy_{Sem}$, based on objects and extensions of relations.

Finally, while many other researchers use a notion of analogy similar to this $Analogy_T$ conceptualization, there are some important differences. Two are presented in Note:2-5, and others appear in Chapter 9.

Note:2-5. How $Analogy_T$ Extends the Traditional View of Analogy:

“And we must at all cost avoid over-simplification, which one might be tempted to call the occupational disease of philosophers if it were not their occupation.”

How to do Things with Words, Austin (1962)

(from pages 201, 202, A-11, and A-56)

The traditional view holds that an analogy is simply a mapping, one which maps a set of source terms onto a corresponding set of analogous target terms. Our $Analogy_T$ fits this description: in particular, its S is precisely this term-to-term mapping.¹²

However, many analogy systems explicate only the terms mapped across, and implicitly assume that all and only the *prima facie* source facts¹³ should be considered. $Analogy_T$ extends this pre-theoretical conception by applying the term-to-term mapping more selectively, applying it only to a specified subset of the derivable source facts. Hence, $Analogy_T$ explicates the set of relevant facts which should be mapped from source to target. This is why its definition includes a particular domain of relevant facts (denoted by the P_B) as well as this S mapping.

¹²Many systems map only atomic terms, i.e., symbols. The fact that S generalizes this by permitting non-atomic terms as well is irrelevant to this discussion.

¹³These are facts primitively stored: i.e., the ones MRS's *LOOKUP* would return. See also Footnote 2 on page 111.

There are many advantages in explicating this *range of applicability*. It means the analogy [1] is not misled by a spurious occurrence of one of these analogous symbols, especially when it does not appear in a sentence of the relevant perspective; [2] does not “over-extend” itself (this responds to the challenge posed in [HM82]); and [3] can be considered “partial”, to permit multiple analogies (see [DM77] and [Bur84]).

This note explains and further justifies this extension. First, Point NotStd1 shows why one cannot use the entire set of facts as the domain of the mapping. The second point shows why the set of *prima facie* facts is not sufficient. While many analogy system deal only with facts primitively stored in their knowledge base, Point NotStd2 demonstrates that the analogy system may first have to perform some deductions before it can find the pertinent source sentences to map over.

Two other preliminary comments: First, we re-express these points within the *Analogy_F* formulation at the end of this note. Second, there are several other distinctions between this *Analogy_T* version of analogy and the more traditional view. Section 9.3 presents some of these, demonstrating that the different formulations of the same facts can lead to different analogies and that a general analogy system may need to extend the language.

NotStd1. Just Subset: (*i.e.*, $P_B \neq Th$)

Once we have a symbol-to-symbol mapping, S , why not just apply it to every Th fact which is about B , or even to all of Th ?¹⁴ The intuitive answer is that the mapping may only apply in certain contexts. That is, the initial theories may include many irrelevant facts, and there is no reason to insist that we find analogues for this superfluous information before we can claim to have an analogy. Worse, it can lead to a contradiction:

¹⁴One possible motivation is that, while *Analogy_T*'s *Trivial_T* condition does bound P_B from below, it does nothing to limit the size of set: *i.e.* it is easy to show that

$$[\neg Trivial_T(x, P, \Lambda, Th) \ \& \ P \subset P'] \implies [\neg Trivial_T(x, P', \Lambda, Th)].$$

For example, imagine that

$$\text{“Discoverer(Current) = Discoverer(VoltageDrop)”} \in P_C$$

and

$$\text{“Discoverer(FlowRate) \neq Discoverer(PressureDrop)”} \in Th_{CF}.$$

Here, the S_{CF} mapping we found so useful above (which included $\{\text{Current} \mapsto \text{FlowRate}, \text{VoltageDrop} \mapsto \text{PressureDrop}\}$) would lead to the inconsistent theory:

$$S_{CF}[\text{“Discoverer(Current) = Discoverer(VoltageDrop)”}] + Th_{CF}.$$

Since we are dealing only with consistent extensions, \vdash would not allow this “Discoverer(FlowRate) = Discoverer(PressureDrop)” addition. This means that P_B cannot be the full theory: only some subsets are possible. (In Hesse’s notation, this “Discoverer(Current) = Discoverer(VoltageDrop)” clause is considered a “negative analogy” [Hes66].)

NotStd2. Deductive Closure: (i.e., $P_B \subset Th$ is *not* sufficient)

This point discusses some situations where we need to use the deductive closure of the starting theory to find an analogical connection. For example, imagine that Th ’s only mention of Ohm’s Law was embedded in a proposition which included other facts which are irrelevant to this analogy, e.g.,

$$\begin{aligned} & [\forall d \text{ Resistor}(d) \Rightarrow \text{VoltageDrop}(j_d^1, j_d^2, [d]) = \text{Current}(j_d^1, d) * \text{Resistance}(d)] \in Th_{CF}. \\ & \quad \& [\text{Discoverer(Current) = Discoverer(VoltageDrop)}] \end{aligned} \tag{A.7}$$

Point NotStd1 demonstrated that we cannot just include this full conjunct in P_C , meaning that we cannot use this useful Ohm’s Law unless we have access to such derivable facts.

As another example, imagine that the only mention of Ohm’s Law in the starting Th_{CF} is embedded in a clause which indicates that the relationship holds independent of temperature. (See [HR70].) Once again, this is a situation where an *entire* source fact cannot be mapped into the target domain, but a weaker, derivable phrase can (and should) be.

These examples demonstrate that two concepts should be deemed analogous even when needed facts are only implicit to the representation. Hence, we must allow $P_B \subset DC(Th)$, rather than only $P_B \subseteq Th$. This is why *Analogy_T* uses $Th \models P_B$ rather than $P_B \subseteq Th$, see Footnote 8 on page A-11.

Now to quickly repeat the above arguments using the *Analogy_F* formulation: First, to parallel Point NotStd1: Notice that the *Trivial* requirement only bounds the set of legal analogical formulae in one direction: i.e.,

$$[\neg Trivial(\varphi, Th) \ \& \ \forall x \varphi'(x) \Rightarrow \varphi(x)] \implies \neg Trivial(\varphi', Th)$$

This lets us consider conjoining together all source facts, to form a huge formula. This is refuted by considering the formula $\varphi(x, y) \Leftrightarrow Discoverer(x) = Discoverer(y)$ and imagining, for argument sake, that $\varphi(\text{Current}, \text{VoltageDrop})$ held but that $\varphi(\text{FlowRate}, \text{PressureDrop})$ did not. (Of course, all of the arguments presented in Subsection 9.2.3 apply here as well.)

For Point NotStd2's claim to be non-vacuous, we can consider the source facts to be the leaf decomposition of the instantiated formula: i.e., P_B must satisfy $LD(Th, \varphi(b_1, \dots, b_n), P_B)$, using the *LD* relation defined in Definition 20. We similarly require $LD(Th, \varphi(a_1, \dots, a_n), P_A)$. We can now apply the same arguments shown above to these P_B and P_A collections of clauses.

Note:2-6. Lexical Measures for Number of Analogies:

(from pages 31 and 32)

It is almost impossible to make general quantitative statements about the number of possible analogies based only on lexical features (i.e., the number of symbols and number of initial sentences) since this involves computing the number of consistent extensions to a given theory. We can, however, derive order-of-magnitude approximations by ignoring many of the syntactic and semantic requirements we have imposed. For a quick, back-of-the-envelope calculation, we do not insist that the analogy be unknown, consistent or non-trivial. (This means our estimate is too large.) We also disregard the comments made in Note:2-5 and Section 9.3

concerning exact formulation, deductive closure and new terms. (Any of these factors could lead to an infinite search space.)

This note approximates a finite lower bound by considering a pair of simplified versions of this analogical inference task. Both simplifications are described in terms of the *Analogy_T* relation, defined in Note:2-4 above.

Approach 1:

One simplification is to restrict the sentence-sentence mapping M to the theory Th : i.e., only consider the submap of M which maps Th into Th . The search space for this simplified problem is still enormous. The number of possible partial mappings taking $N = \|Th\|$ source facts into the N target facts is $(N + 1)^N$ ¹⁵ (We *do* need to consider all of Th ; Section 2.5 demonstrates that it is not enough to consider only those sentences which lexically include *Current*.)

Approach 2:

This second approach decomposes the task of finding an analogy into the two considerations posed in Figure 2-4. Page 32 shows we should consider the full set of $2^{\|Th\|}$ possible P_B s. Now consider S : how many constant-constant mappings are there? If the language \mathcal{L} has $\|\mathcal{L}\|$ constant symbols, there are about $\|\mathcal{L}\|^{\|\mathcal{L}\|}$ such partial mappings. (If we follow Section 9.3.2's advice to allow reformulation (read "new terms"), this number becomes unbounded.)

If we assume that P_B and S are independent, the number of possible analogies is approximately their product. Figure A-2 summarizes the numbers. (Once again, realize that this is an overestimate in one sense, since it includes inconsistent, known and trivial mappings; and an underestimate in another, since it considers only sentences explicitly in Th and only pre-defined constants.)

Note:2-7. Full Number of Possible Analogical Inferences:

(from pages 32, 35 and 193)

¹⁵This "!" indicates amazement, not factorial, fortunately... and that "15" refers to this footnote, not exponentiation.

- Any $P_B \subseteq Th$
 $2^{\|Th\|}$
- Any symbol-to-symbol map, S :
 $\|\mathcal{L}\| \|\mathcal{L}\|$
- If independent:
 $2^{\|Th\|} * \|\mathcal{L}\| \|\mathcal{L}\|$

Figure A-2: Number of Analogies

This note extends the composition scheme discussed on page 32, especially Equation 2.19. In particular, it considers other ways of extending an analogy.

First, to elaborate the comment in Footnote 13 on page 32: the conjunction of two analogies is not guaranteed to be an analogy. The problem is that the two analogy formulae may not be independent. In particular, $Th \not\models \neg\varphi_1(A)$ and $Th \not\models \neg\varphi_2(A)$ does not guarantee that $Th \not\models \neg(\varphi_1(A) \& \varphi_2(A))$.

As an example, consider the theory

$$Th = \{ f(B), g(B), \neg f(A) \vee \neg g(A), \neg f(C), \neg g(C), \dots \}.$$

It is easy to confirm that

$$\begin{aligned} Th, A \sim B &\vdash f(A) \\ Th, A \sim B &\vdash g(A). \end{aligned} \tag{A.8}$$

Now consider the formula

$$h(x) \Leftrightarrow f(x) \& g(x).$$

Does

$$Th, A \sim B \stackrel{?}{\vdash} h(A) \quad ?$$

Unfortunately not, as it violates the **Consistent** condition; i.e., $Th \models \neg h(A)$.

Fortunately, such cases are rare. This is why I chose to ignore this situation for this analysis. A similar situation occurs when we consider disjoining analogies; this note disregards such cases as well.

satisfies

$$Th \models \varphi(B)$$

$$Th \not\models \varphi(A)$$

$$Th \not\models \neg\varphi(A)$$

Towards building up the class of legal analogies, we now restrict these Ψ_χ s to consist of only *general base unary formulae*, where a *gbuf* is either a *buf* (as defined in Definition 10 on page 34) or the negation of a *buf*. (This means that $\phi_1(t) = \text{Kirchoff1}(t)$ and $\phi_2(o) = \neg\text{Group}(\mathcal{R}, o, 0)$ each qualify as a *gbuf*, but $\phi_3() = \text{Kirchoff2}(\text{FlowRate})$ and $\phi_4(d, o) = \text{Monoid}(d, o, 1)$ do not.)

We can generate a subset of the legal analogies by combining members of these sets. Let Π represent a collection of subsets, where Π_χ is an arbitrary subset of Ψ_χ for each Ψ_χ in Table A-1. We require that Π_{0+} be non-empty; any other subset can be empty. For any such collection of subsets, we can form

$$\varphi_\Pi(x) \iff \left[\begin{array}{c} \bigwedge_{\psi \in \Pi_{0+}} \psi \quad \& \quad \bigwedge_{\psi \in \Pi_{++}} \psi \\ \bigvee_{\psi \in \Pi_{-+}} \psi \quad \vee \quad \bigvee_{\psi \in \Pi_{--}} \psi \end{array} \right] \vee \left[\begin{array}{c} \bigvee_{\psi \in \Pi_{0-}} \psi \quad \vee \quad \bigvee_{\psi \in \Pi_{00}} \psi \\ \bigvee_{\psi \in \Pi_{-+}} \psi \quad \vee \quad \bigvee_{\psi \in \Pi_{--}} \psi \end{array} \right] \quad (\text{A.10})$$

Using Equations 2.19 and A.9, we see that this $\varphi_\Pi(x)$ is (usually) a legal analogy.¹⁷ How many such φ_Π s can be generated? Assume there are n_χ members in each Ψ_χ : i.e., $n_\chi = \|\Psi_\chi\|$. (E.g., there are n_{0+} “base” analogies.) As there is one formula φ_Π for every legal Π , this leads to

$$(2^{n_{0+}} - 1) * 2^{n_{++}} * 2^{n_{0-}} * 2^{n_{00}} * 2^{n_{-+}} * 2^{n_{--}} * 2^{n_{-0}} \quad (\text{A.11})$$

lexically distinct analogy formulae! To a first approximation, this corresponds to the power set of all general base unary formulae. (It omits only members of Ψ_{+-} and Ψ_{+0} , and has the additional requirement that at least one member of Ψ_{0+} be included.) Each of these Ψ_χ sets is huge, since it includes all possible distinct bindings for each relevant n -ary relation. (See page 35.) This shows, again,

¹⁷Recall from Footnote 16 that we are incorrectly assuming that this φ_Π formula is **Consistent**.

that the space of all possible analogies is gigantic, even with the “trivializing” *no reformulation* assumption.

Two final notes:

- This set of analogy formulae does not include all possible analogies. For example, imagine that Th contains only one B-sentence which is not a base unary formula:

$$Th = \{ \text{frob}(B) \vee \text{gorf}(B) \}$$

Since there are no unary base relations in Ψ_{0+} , the method described above would be unable to construct any analogies. However, of course,

$$Th, A \sim B \vdash \text{frob}(A) \vee \text{gorf}(A).$$

- While the φ_n s generated in this manner are guaranteed to be lexically distinct, some will certainly overlap semantically; *i.e.*, Th will be able to prove that some different φ s are equivalent.

A.3 Notes from Chapter 3

Note:3-1. Ease of Finding Analogies: Two Considerations:

(from pages 43, 78, 115, 209, A-26 and A-82)

One possible measure of how good an analogy is might be how easy it was to find that analogy. This note presents two comments on this measure. First, it does say something, albeit very little, about the utility of the analogy itself. Second, this easiness measure can be quantified in terms of the number of deductions and conjectures which were required to find the analogy. This means it depends on both the lexical *form* and semantic *content* of the starting theory, Th . This note discusses why the first (form) is of little worth, while the second (content) is quite important. (Section 9.3 echoes this view.)

First, why consider this easiness measure at all? While it certainly is relevant when analyzing the difficulty in *finding* the analogy (*e.g.*, when determining how long the learner may require to flesh out an analogy), there are no obvious ties

connecting it to the “absolute goodness” of an analogy, once it is found. Even though this measure does not, in and of itself, provide any direct insight into what constitutes a good analogy, there are secondary reasons why it should be considered.

Its significance follows from the indirect reason that people (and hence their programs) have cached away pertinent facts, in a form which leads to important analogies first. To restate this same observation from another angle: People use a certain vocabulary to encode their observations. This ontology reflects both our perceptual abilities and our notions of salience. Different cultures will have different set of concepts; consider examples like the inability of certain African tribesmen to express any number above four, or our inability to conceptualize a proposition which is both true and false. This set of terms restricts our ability to conceptualize, and constitutes what van Lehn considers an *absolute bias* [van85].

Even with this fixed vocabulary, there are still many ways of describing any phenomenon. Here, again, we tend to express the phenomena in certain ways and not others. For example, we may prefer to state “Fred is married to Jane” rather than “There was a marriage event which involved Fred and Jane”. Even though these two formulations are semantically equivalent, this preference also reflects a bias (here, a relative one). My claim is that these biases emphasize those aspects which we consider salient, often at the expense of down-grading other less-important features.

Of course, this bias dictates how we specify objects in general, and the analogues in particular. (I view this as a projection process: each object is projected onto a certain vocabulary and description conventions.)

Now consider the task of finding the analogy between two analogues. This process is trivial if the analogues are represented appropriately. Here the analogy isomorphism requires only matching explicitly known features of the analogues. Otherwise, the mapping may involve “implicit” facts, which must first be exposed. (See Subsection 9.3.2.) This requires a reformulation pre-processing, which means performing some number of additional deductions.

This suggests that the complexity of the analogical inference process depends on our ontological bias. Fortunately, this bias seems to work in our favor: we tend to represent things in a way which reflects their salient features. Given this assumption, we should prefer analogies which are closest to our representation; i.e., which require the fewest number of deductions (i.e., which exposes the fewest “implicit” facts).

This is the indirect justification suggested above. It is still only a weak heuristic, completely dependent on strong assumptions about not only our internal encoding of the world, but also on the particulars of the goal analogy, since an encoding which favors one analogy may disfavor another.

(Of course, this only begs the question one step farther, since we do not know why these particular vocabulary items and relations (but not others) were deemed sufficiently important to have been selectively cached. We simply accept this as given, and attempt to use this stored *a priori* information. Section 7.6 provides some empirical evidence that suggests that people’s linguistic conventions have done pretty well at this task. See also Note:4-4.)

Our implementation includes both of these criteria (favoring fewer deductions and fewer conjectures) as heuristics: H_{JK} is an example of the first and H_{MGA} , H_{FC} and H_{FT} exemplify the second. (All appear within Chapter 5.)

Onto the second part of this issue: the claim that these two measures have very different levels of importance. Despite the hand-wavy arguments above, the number of deductions measure is still totally syntactic, which is contrary to my view that an analogy is a semantic relationship. (See Section 8.1.) Hence, this measure is only considered “at the noise level”, as a final tie-breaker, in one part of one heuristic. This seems an appropriate rôle, given its weak and indirect justification.¹⁸

On the other hand, the minimum number of conjectures principle does have

¹⁸This measure appears much more prominently in many other analogy systems: [Win82]’s reliance on *prima facie* facts is a prime example. In fact, that system uses this criterion to prune the space, not only to order it. Subsection 9.3.2 argues that this criterion should not play so prominent a rôle.

a solid semantic relevance and should play a major rôle in evaluating an analogy. Section 5.3 argues this position.

A.4 Notes from Chapter 4

Note:4-1. How Can Abstractions be Derived?

(from pages 48 and 222)

As a preliminary disclaimer, realize that this research is specifically **not** concerned with the problem of generating abstractions. The focus, instead, is on providing a mechanism capable of using such already-known abstractions effectively. However, these abstractions would seem rather contrived and uninteresting if their derivation required analogies as well. This note demonstrates that this is not the case: that these abstractions can be derived in a way that is independent of any analogy process. (See also Note:4-2 for a discussion of the general utility of abstractions, which illustrates other uses beyond this specific analogy application.)

Section 4.1 mentioned that each abstraction can be viewed as a system of co-occurring relationships, and then described each abstraction as a generalization from prior experiences. How does one generalize from an experience? The chunking [Ros83] and Leap [MMS85a] research each point to the need for a thorough analysis of that situation, to determine, in particular, which factors were important and which tangential. From this perspective, generalization is a process which preserves only relevant features, in a form which allows them to be used in a new situation.

There may be many different ways of finding situations to serve as base cases. Section 4.1 suggested that traces of a general problem solver at work can be used. Expanding that argument, consider the general task of solving a “Find the Current” problem, independent of any analogy. As mentioned earlier, we need deal only with relations like Current, VoltageDrop and Resistance, and with objects like wires and junctions. Imagine that these associated facts — those used to solve this problem — were gathered together. After solving several of these problems, patterns would emerge: certain sets of facts would be

used together consistently, while others would never be considered. Each such (near-)clique would become a perspective: the **RKK-EC** collection of facts is one example.

When enough examples of these clusters are found, they could be variabilized (*à la* [FHN72]) to form the abstractions — such as the **RKK** one used so effectively above. Of course, this parameterizing task depends critically on selecting which scenarios should go together, as it is the commonalities and differences between the members of such collections that determines which of the constants should become formal variables.¹⁹ Although life is easier if a teacher supplies the abstraction-finding program with such “Scenario A is like Scenario B” information, this is not necessary. A sophisticated “variabilizer” might use a set of general heuristics to determine which scenarios seem to fit together, based, for example, on second-order information about syntactic properties of the relations.²⁰ (See Points FW1 and FW2 in Section 10.2.)

This task of generating abstractions remains a very difficult learning problem, one which involves many hard processes, including cluster analysis and induction. However, we see that it can be done independently of any *a priori* analogical information.

An important subcase of this cluster-as-needed idea is to generate new abstractions from old. Three different methods are discussed below.

Method#1 involves combining sets of abstractions together, e.g., two merging group structures into a field.

¹⁹It is often sufficient to simply find cases where different constants are used in the same position in two different situations, and then change this pair of constants into a single formal parameter (possibly one scoped by moving up some hierarchy, *à la* the Candidate Elimination Algorithm [Mit79]). This tact only works when given the proper ontology: i.e., only when the world is “cut at its joints” [Boy79]. Of course, we know that this approach works quite often. I view this as a further reinforcement of my claim that problem solving activities should induce the choice of vocabulary: see Note:3-1 and Subsection 7.6.3.

²⁰As evidence that this may be possible, notice that people seem able to generalize quite well from a single example. The relates to Quine’s idea of ostension [Qui60] as well as various recent AI research on learning-from-one-example (e.g., [Cyp82]) as well as the body of work on explanation-based learning [Mit85].

Method#2 is based on weakening a single abstraction. This has two forms. One involves removing only some component sentences, but leaving all of the formal variables. For example, this would let us derive the ring abstraction (or the integral domain abstraction) by weakening the notion of field.

The other variant involves is to “ignore” some of the parameters. We can think of something as general as a script as an abstraction, one with dozens of arguments [SA77]. For some tasks, it is sufficient to specify only certain relevant ones parameters, and allow the other ones to be defaulted (or simply disregarded, if they are irrelevant to the current task). That is, we may only need a partial instantiation of some abstraction. This suggests two related ways of generating abstractions: [1] filling in default values for some variables (to derive the **VelvetTurtle-Script** from the more general **Restaurant-Script**) and [2] removing some of the arguments, together with the component clauses that deal with these alone — this would let us derive **Group** from **Field**.

Method#3 involves turning some (possible implicit) constant in an abstraction into a new variable. This suggests two ways of generalizing the **RKK** abstraction. One involves replacing the constant junction with a variable. This leads to the definition of **RKK-Junc** which appears in Figure A-3. Here, we use $j(d, i)$ to denote the i^{th} connection of type j associated with the device d : (This obsoletes the more restrictive j_d^i notation which dealt exclusively with junctions; i.e., $j_d^i = \text{junction}(d, i)$.) As we are not restricted to devices meeting at junctions, we can now talk about freeways meeting at clover-leaves, etc.

The other generalization is more subtle. In Note:1-2, we saw that a more nearly correct resistance analogue should deal with a power of the through term. This suggests expanding the **RKK**(t, c, r, l) relation into the more comprehensive version, **RKK-Exp**(t, c, r, l, e), which explicates this exponent. This differs from **RKK** only in using the Ohm-Exp rule rather than the Ohm rule, where

$$\begin{aligned} \text{Ohms-Exp}(t, c, r, l, e) &\iff \\ \forall d \ 1(d) \Rightarrow [c(j_d^1, j_d^2, [d]) = t(j_d^1, d)^e * r(d)] &\end{aligned} \tag{A.12}$$

$$\mathbf{RKK}\text{-Junc}(t, c, r, l, j) \iff$$

$$\left\{ \begin{array}{l} \mathbf{Ohms}\text{-Junc}(t, c, r, l, j) \\ \quad \forall d \, l(d) \Rightarrow [c(j(d, 1), j(d, 2), [d]) = t(j(d, 1), d) * r(d)] \\ \mathbf{ConservedThru}\text{-Junc}(t, l, j) \\ \quad \forall d \, l(d) \Rightarrow [t(j(d, 1), d) + t(j(d, 2), d) = 0] \\ \mathbf{Kirchoff1}\text{-Junc}(t, j) \\ \quad \forall k \, k \in j|_1 \Rightarrow \sum_{p: \text{Conn}(p, i)=k} t(k, p) = 0 \\ \mathbf{Kirchoff2}\text{-Junc}(c, j) \\ \quad \forall l \in \text{Loop}(j) \Rightarrow \sum_{\langle i, j \rangle \in l} c(i, j, [x]) = 0 \end{array} \right.$$

Figure A-3: Definition of the **RKK-Junc** Abstraction

This allows us to express **RKK-Exp**(Current, VoltageDrop, Resistance, Resistors, 1) and **RKK-Exp**(FlowRate, PressureDrop, PipeCharacter, Pipe 0.6).

Another reason to add an addition term is to expose a definable term. For example, suppose we wanted to explicitly (and externally) reason about the inverse of a group's operation. We would need to use the new **Group'** abstraction, which explicitly included the extra argument. By example, we would have **Group'**($\mathcal{R}, +, 0, -$). (This relates to the discussion on page 135.)

Note:4-2. Other Future Uses of Abstraction:

(from page 228 and A-25)

This note discusses other possible applications of *coherent clusters of facts*, i.e., abstractions. In particular, it considers their eventual applicability and utility to both machine learning and expert system.

This research shows the advantages of acquiring a connected set of facts at a shot, rather than the more piecemeal approach of ingesting a single fact at a time. While I only tested this claim when considering *learning by analogy*, it is hard

to imagine that this principle would not apply to other forms of learning as well; hence, this clustering idea will have an important impact on the general area of machine learning. (Of course, this idea is not new: this approach is implicit in any system which, for example, instantiates a complete frame whenever it proposes a new concept; *cf* [Len82a,Len83a].)

This suggests a rôle for abstractions during the education of future Expert Systems, *i.e.*, during their Knowledge Acquisition phase. I claim that abstractions can be used upgrade their performance as well. To be able to address a variety of tasks over a multitude of domains, such systems will undoubtedly use massive Knowledge Bases. This means that only a very small percentage of the facts will be applicable to any problem. Without the effective use of some focusing tool, their problem solving efficiency, and hence their effectiveness, can easily degrade as these systems expand. These abstractions suggest a mechanism of focusing attention on certain clusters of facts, thereby effectively permitting the problem-solver to ignore the remaining, probably irrelevant facts. This facility becomes essential when the inference engine is expected to *deduce* (as opposed to simply retrieve) necessary facts.

Note:4-3. Importance and Generality of Model Based Approach:

(from pages 64 and 221)

Section 4.3 demonstrated that the learning-by-analogy task can often be reduced to selecting and instantiating the appropriate existing abstraction. This note discusses this model-based approach in general, expressing both its prominence and relevance.

The *NLAG* project represents just one approach to the problem of learning by analogy, one based on pre-existing models. A variety of other tasks are also based on this underlying top-down process, including many sophisticated Artificial Intelligence programs: *e.g.*, the ACRONYM's model-based vision system [Bro84], Molgen's skeletal plan instantiation process [Fri79], the Programmer's Apprentice cliché and plan instantiation [Ric79,Ric81,Bro81],²¹ Barstow's PECOS code

²¹The transitive closure of relevant references leads to [Flo78] and [DDH72]. These, too, discuss the

generation program [Bar79], Wilensky's plan instantiation schemes [Wil83], and the XCON's (*née* R1) configuration program [McD80]. Other "model-based systems" include the various "how-to" books which describe complex tasks such as preparing a résumé, giving a presentation, or wood-working. Each case (book, algorithm or system) describes one or more abstractions by providing, for each abstraction, the relevant parameters (e.g., the introduction, statement and conclusion of a talk, or the various parts of a wooden ship, ...) and certain properties and constraints of those components, both individually (e.g., specifying what goes into an introduction, or how to build certain subparts, ...) and collectively, giving relevant interconnections among these parts (e.g., the whole talk should be on the same theme, or the wood's colors should match, ...).

There is, of course, another way of addressing many of these tasks: One could, instead, work bottom-up, constructing successively higher-level interpretations based on lower-level primitives. Other systems follow this approach.

These two approaches have different merits and limitations. Section 4.3 mentions some of the benefits and limitations inherent in the top-down approach, demonstrating that such systems can be quite focused (and hence efficient) when dealing with situations they can handle, *i.e.*, when some known model applies. A model-based system, however, is inherently dependent on its models: its scope is limited to the space defined by these structures. Systems which use a bottom-up approach, on the other hand, can cover a relatively unbounded space. The cost is efficiency: bottom-up systems do not have the focus associated with top-down searches.

This dichotomy appears prominently in computer image understanding systems. Some, including the ACRONYM system mentioned above, work top-down, attempting to locate specific models. Others work bottom-up; *cf.*, [Wal75], [NB81], [BT78] and [MP77]; see also [Kan82] for a comprehensive survey. Section 9.2 illustrates that this holds for research in analogy as well

As suggested above (and discussed on page 53), *NLAG* only finds those analogies which correspond to a known, pre-defined abstraction. This reliance compares with *ACRONYM*'s dependency on its specific three-dimensional models. Each system "interprets" its data (be it pixels or propositions), and each finds instances of a general object only when that general object is included in its initial data base. (*E.g.*, *ACRONYM* could not, for example, find a person in a picture unless it had an explicit model of a person.) Neither they nor I would claim that our respective systems can fully solve the general vision (respectively, analogy) problem; their objective, like mine, is to establish the utility and effectiveness of using such models. Each system is providing one step towards the general solution (see Section 10.2).

While this instantiation process has not received the glamour accorded to other learning tasks (notably induction and learning generalizations [DLCD82]) it is nevertheless an important and complex undertaking — well worthy of the Nobel laureates and university professorships it has earned its practitioners: consider the descriptions of such notables as Darwin and Edison shown on page 2, as well as Lorenz ([Lor74]), Cohen ([Coh80]), ... Although their tasks involve "merely" instantiating one of the abstractions explicitly present, few would claim these accomplishments were trivial.²²

Note:4-4. Justification for Abstractions:

(from page 66, 107 and 183)

This note presents two comments, each related to the question of how one can justify using abstractions. The first is the quick observation that there seems to be no semantic justification for abstractions. The second elaborates this lack of justification position, arguing that there is no intrinsic reason why abstractions should work at all.

While we can state the abstraction-based analogical inference process semantically (this is done in Section 4.2's Point Sem4), it is surprising that there seems no

²²If only it were so trivial to instantiate the "how to write a good dissertation" skeleton with respect to, say, the topic of analogy...

semantic justification for the use of abstractions, neither for analogies in particular nor for learning in general. (By contrast, we can provide semantic reasons for many of the other heuristics, including Subsection 5.3's least constraint rule and Subsection 9.2.2's (unused) maximal specificity rule.) Instead, the abstraction's *raison d'être* seems inherently computational, that is, syntactic. This non-result holds for all of the abstraction related rules, including this H_{Abst} rule as well as the other two refining rules, H_{JK} and H_{CC} (described in Section 5.2).

We can, however, provide some *syntactic* justifications for this concept of abstractions, in terms of traces of past performances. (See Section 4.1 and Note:4-1.) Unfortunately, on closer examination, even this is suspect. One can view the process of generating abstractions as an induction process: Knowing that some collection of facts holds in one domain, we inductively infer that it will hold together in others as well. How can we justify this claim; that is, why does this domain-induction work? Well, it worked from hydraulics to electricity, and from natural numbers to matrices, and from people to computers, *etc.*, and so, by induction, it will work in general. But why should this induction, from certain pairs of domains to the general collection of pairs of domain, work? We face the same problem in justifying this claim that confronted Hume, and lead him to his famous dilemma [Hum02]. Of course, the assumed continuity here is over domains, rather than (just) over time. This means we are exploiting the "continuity of the world" [Len84] — in a conceptual, rather than just temporal, sense.

Note:4-5. Is the World *Really* Simple, or is it only Our Perception:

"The feeling that harmonious simple order cannot be deceitful guides the discoverer [...] and is expressed by the Latix saying: simplex sigillum veri (simplicity is the seal of truth)."

How to Solve It, Polya (1957)

(from page 66)

I claimed that one can effectively use abstractions in natural domains. There are two possible justifications for this position. View#1 holds that the world

really is simple, that is, the world really does exhibit this continuity property. Alternatively, View#2 claims that this simplicity is all in our perception; that the world is really quite complex, but we have simplified it to permit us to deal with it.²³ For the purposes of this research, it is irrelevant which of these views *really* holds.²⁴ It is enough that we can effectively model the world as continuous, whether or not the world *really* is.

As a final aside, realize that analogies (read “common abstractions”) were thought to have a firm ontological basis until relatively recently. For example, scholars in the Middle Ages believed that the universe was ordered such that the “macrocosmic pattern of the whole is reproduced in the microcosmic pattern of the parts” [Emm85]. This perceived association allowed them to draw inferences from one to the other. (Of course, many of these beliefs still persist, consider Astrology. See also the Law of (Magical) Sympathy, discussed in [Lit73] and [Fra22].)

A.5 Notes from Chapter 5

Note:5-1. Facts Lexically Including the Source Analogue are Sufficient:
(from pages 78 and 114)

Given a theory, Th , and a symbol, A , $NLAG$ searches for all derivable abstraction instances which contain A : i.e., all S such that

$$\begin{aligned} & \text{Abstraction}(S) \\ & \exists a_{j \neq i} Th \models S(a_1, \dots, A, \dots, a_n) \end{aligned} \tag{A.13}$$

²³Shepard has amassed a great deal of evidence to support the claim that people will do almost anything to establish a “continuity” [She84]. For example, people prefer to see an object simply moving or being somehow distorted, rather than accept the possibility that one object disappeared and was replaced by another.

²⁴Unfortunately, we derive no insights from the attempt to “apply” each view to itself: If the world really is simple, then this simple claim that “Abstractions apply to many domains” should hold. On the other hand, suppose we only think that the inherently complex world is really simple. Once again, we would deceive ourselves into believing this simplistic “Abstractions apply to many domains” idea. Hence each view is consistent, even when applied to itself.

This note proves that the derivation of any such abstraction instance must involve a member of Th which lexically includes this symbol A . Stating this formally requires two definitions:

- The support for a conclusion σ with respect to a theory, Th , $ST_{Th}(\sigma)$.
(See Definition 8 on page 31.)
- The “ x ” lexical projection of a theory, Th , which is the subset of Th ’s sentences which lexically include the symbol x :

Definition 35 $\mathcal{P}_x(Th) \stackrel{\text{def}}{=} \{\sigma \in Th \mid \text{LexInclusion}(\sigma, “x”)\}$

(See definition on page 24.)

Given these two definitions, we claim that

$$\begin{aligned} Th \models \mathbf{S}(a_1, \dots, A, \dots, a_n) \quad \text{and} \\ \text{Abstraction}(\mathbf{S}) \end{aligned} \tag{A.14}$$

guarantee that

$$ST_{Th}(\mathbf{S}(a_1, \dots, A, \dots, a_n)) \cap \mathcal{P}_A(Th) \neq \{\}. \tag{A.15}$$

This note actually proves that Equation A.15 holds for any formulae (not just abstraction formulae), providing the term A appears non-trivially. (Definition 5 defines triviality.) Since abstraction formulae must be non-trivial in all of their arguments (see Section 4.2’s Point CA1), they automatically qualify: i.e., this proof is sufficient.

Proof: Towards a contradiction, assume that

$$ST_{Th}(\mathbf{S}(a_1, \dots, A, \dots, a_n)) \cap \mathcal{P}_A(Th) = \{\};$$

i.e., that no $\sigma \in ST_{Th}(\mathbf{S}(a_1, \dots, A, \dots, a_n))$ lexically includes the symbol A . This can only happen if there is some σ_j of the form $\forall x \varphi(x)$ which is instantiated to A . (If there are more than one such universal formulae, choose one arbitrarily, as if it is the first one to be instantiated.) This means that A ’s occurrence in that instantiated form is trivial: by the nature of instantiation, any other term

could have been used as well. Since nothing else distinguishes A in this derivation, the full proof could have used any other symbol in A 's place. That is, this same $\mathcal{ST}_{Th}(\mathbf{S}(a_1, \dots, A, \dots, a_n))$ could prove $\mathbf{S}(x_1, \dots, X, \dots, x_n)$ for any X we like (with a possibly different set $\{x_i\}$, i.e., $\{x_i\} \neq \{a_i\}$). So A 's appearance in the original $\mathbf{S}(a_1, \dots, A, \dots, a_n)$ is trivial, violating our initial non-triviality assumption. \square

As a final comment, it is easy to see that this lexical inclusion criterion is not sufficient if we allow trivial formulae. For example, observe that $f(A)$ can be derived from $Th = \{\forall x f(x)\}$. However, its support contains no member which lexically contains the symbol A : i.e., $\mathcal{P}_A(Th) \cap \mathcal{ST}_{Th}(f(A)) = \{\}$. (In fact, $\mathcal{P}_A(Th) = \{\}$.)

Note:5-2. No New Terms:

(from page 102)

The *NLAG* process does not generate new terms. This was a deliberate decision, based on the realization that such new terms seldom help to solve the given problem since they lack the (usually necessary) ties to the current situation.

For example, imagine we had generated a new Resistance-like hydraulics term for the "Find the flowrate" problem; call it G0091. Notice we still can not solve the problem, since that solution would require computing the value of that function applied to the various pipes, and we have no way of ascertaining that value. (That is, we have no way of determining Pipe3's G0091 value.)

One the other hand; this claim is not universally true. There are situations where generating a new term would help to solve a problem. For example, notice that the PressureDrop constant played only a place holder rôle in our canonical hydraulics problem. This means that an arbitrary G0017 would have done as well.

A.6 Notes from Chapter 6

Note:6-1. Ordered based on Number of Deductions:

(from pages 115 and 137)

This note discusses how to order the space of deductions which follow from a kernel, Φ , with respect to a theory Th . By assumption, the kernel is a subset of the theory ($\Phi \subseteq Th$) and the theory is consistent ($Consistent(Th)$).

This ordering is defined in terms of the distance between a proposition and the kernel, with respect to an underlying theory. We use the $K_i(Th, \Phi)$ family of relations to define this distance, where

$$\begin{aligned} K_0(Th, \Phi) &= \Phi \\ K_{n+1}(Th, \Phi) &= K_n(Th, \Phi) \cup \\ &\quad \{ \kappa \mid \exists \rho_0 \in K_n(Th, \Phi), \{ \rho_i \}_{i=1}^m \subset DC(Th), \{ \rho_i \}_{i=0}^m \vdash_{IM} \kappa \} \end{aligned} \quad (A.16)$$

where $DC(Th)$ is the deductive closure of the theory Th and the \vdash_{IM} operator encodes our particular inference mechanism. As our current implementation is based on production rules, some ρ_j is of the form " $\sigma_1 \& \dots \& \sigma_m \Rightarrow \kappa'$ ", where this κ' can be instantiated to κ and each σ_k , to some ρ_i .

We can define the *distance* of a proposition from a kernel in terms these K_i relations: *viz.*,

$$dist(\varphi, \Phi, Th) = \min_i (\varphi \in K_i(Th, \Phi)). \quad (A.17)$$

This *dist* metric defines "closeness" in the obvious way: φ is closer than ρ (with respect to the kernel Φ and theory Th) whenever $dist(\varphi, \Phi, Th) < dist(\rho, \Phi, Th)$.

(In case of ties, the order depends on various idiosyncrasies of the underlying MRS implementation. In MRS's current implementation, the order depends on when the various base facts and defining rules were asserted. We could redefine this *dist* measure to break ties accordingly.)

Note:6-2. Why *Inst-Source* Considers More General Abstractions?

(from pages 105, 115 and A-88)

Page A-88 shows that *NLAG* considers **KK** to be a possible common abstraction even after the more specific **RKK** has failed. This point discusses why this makes sense.

After **RKK** has failed to qualify as a useful analogy, one might wonder whether we should bother to try **KK** at all? After all, how could a subset of **RKK**'s facts lead to an answer to this query when **RKK**, *en masse*, could not? To state this intuition formally: from $Th + \varphi(A) \not\models PT$, we know that $Th + \rho(A) \not\models PT$ for any more general ρ (i.e., whenever $\varphi(x) \Rightarrow \rho(x)$);

$$\frac{Th + \varphi(A) \not\models PT \quad Th \models \forall x \varphi(x) \Rightarrow \rho(x)}{Th + \rho(A) \not\models PT} \quad (A.18)$$

While Equation A.18 is correct, there are two reasons why it does not apply here. First, it pertains only to usefulness, but not to the other part of being a useful analogy, namely, that the new conjecture is a legal analogy. All *NLAG* knows is that **RKK**(FlowRate, ...) is not a legal \Vdash_{PT} analogy. It does not know whether this conjecture failed the analogy test or the usefulness test. Since

$$Th_{CF, \text{FlowRate} \sim \text{Current}} \Vdash \text{KK}(\text{FlowRate}, \dots) \quad (A.19)$$

might hold (and thence lead to a useful analogical inference \Vdash_{PT}), even though

$$Th_{CF, \text{FlowRate} \sim \text{Current}} \Vdash \text{RKK}(\text{FlowRate}, \dots) \quad (A.20)$$

does not, it behooves *NLAG* to consider this abstraction even after the more specific one has failed.

This first reason, by itself, is not sufficient. A more thorough analysis might still have caught this case: that is, *NLAG* could have observed that Equation A.20 did hold. This means that **KK**(FlowRate, ...) must have failed the usefulness test, since this is now the only way that **RKK**(FlowRate, ...) could fail to be a legitimate \Vdash_{PT} analogy.

Equation A.18 seems to imply that this information is now enough to eliminate the **KK** abstraction. However, that equation deals only with unary formulae, and

so does not consider the other “Skolem variables”; the more general abstraction might permit other bindings for these arguments, ones not permitted for the more specific one. (That is, knowing that $\forall x [\exists y \varphi(x, y)] \Rightarrow [\exists z \rho(x, z)]$ does not guarantee that $\forall x \forall y \varphi(x, y) \Rightarrow \rho(x, y)$.) These alternate other terms might make this new information useful to this problem.

A further analysis of these **KK** and **RKK** abstractions could solve this problem too; that is, we could observe that **RKK** is more specific *in every argument*. If *NLAG* used a truly complete deductive mechanism, this would be conclusive: that is, we could now positively eliminate **KK** once **RKK** had failed. However, *NLAG* uses various heuristics which leave it incomplete. At this point, it is worth the (minor) additional expense of actually considering these more general abstractions, rather than perform the increasingly vast amount of computation to determine whether or not we would need to consider this abstraction. (Thanks in part to the *SuperCache* facility, described in Subsection 6.2.4’s Point MRS1, this computation is usually relatively inexpensive, anyway.)

A.7 Notes from Chapter 7

Note:7-1. How \Vdash_{PT} Compares with Our Intuitions:

(from pages 138 and 162)

While it has never been an explicit goal of the research, the *NLAG* system does seem to correspond to the way people (or at least the author) use analogy. For example, the idea of abstractions is clearly consistent with many psychological theories: *i.e.*, people do tend to clump facts into “cohesive” groups, rather than view them in isolation (see [Ros73, Min75]). This is true not only in general problem solving activities, but for learning as well: people do tend to learn coherent clusters of facts at once.

NLAG’s underlying mechanism also seems “humanly reasonable”: going first bottom up to develop an hypothesis (here, a possible common abstraction) and then top down, trying to instantiate it, first in the source and then in the target domain. (This resembles visual perception or speech understanding. See [Fel78])

and [LE77,NA79,Hay85].)

Many of the particulars of the *NLAG* model of analogy resemble pre-theoretical intuitions of analogy as well. (Table 7-7 provides some empirical evidence supporting many of these claims.)

- **Depends on Learner:** Given A and B, the analogical connections which can be found depends on the learner's initial knowledge, *Th*, as well as the problem-solving context. (See [Fra79] for psychological evidence of this obvious idea; as well as Chapter 2 and Section 9.3.)
- **Not Unique:** Even given A, B and *Th*, there is often more than one possible analogy. (The goal of solving a problem does reduce the number of candidates, but usually not to a single member.) (See Sections 2.2 and 3.3.)
- **Obvious, in Hindsight:** Analogies tend to be obvious in hindsight, although not *a priori*. That is, an analogy becomes almost self-evident once it is "deciphered". This is because an analogy is a simple isomorphism when viewed in the correct framework (*i.e.*, using the appropriate abstraction). (See Note:2-4 and Section 9.3.2.)
- **Symmetric Relation, Non-symmetric Inference:** The analogy relation is symmetric between source and target analogues: *i.e.*, *Analogy_F* is symmetric in its first and second arguments. Of course, the analogical inference process is not, since it relies on facts known about one analogue to postulate unknown facts about the other. This also makes sense: the A~B analogical hint may be used to state that some B-fact pertains to A, even though we would not think of using B~A to go the other way: *i.e.*, to transfer the related A-fact to B. (Searle discusses this one-way transference [Sea79].)
- **Knowledge (usually) Constrains Search:** Knowing more relevant information about the target analogue helps: it means less searching is required, as there are fewer wrong candidates.²⁵ On the other hand, knowing more

²⁵In the current *NLAG* implementation, it is possible that a new target domain fact might increase the number of analogies. For example, adding "(mem F00 functions)" means that F00 is now known to be function, and so is eligible for any variable whose generator is, *e.g.*, (mem ?2 functions). This is due to the nuances of my operationalization of the *H_{FT}* rule.

about the source analogue can be problematic, if these facts suggest multiple abstractions. Here, additional search may be required to determine which abstraction should be considered.

Finally, knowing more abstractions is a mixed blessing. It does provide greater coverage, but may mean that understanding any given analogy may be slower, since there may be more possible commonalities to consider. (This can vary, depending on the sophistication of the *Find-Kernel*-like process — i.e., on how effectively the target problem can be used to focus the search.)

In each of the latter cases, the degradation is significantly lessened if the new information is somehow disjoint from the target facts sought (or desired abstraction). In such cases, this new information does not interfere.

Note:7-2. Another Example of Analogy: + is like *:

(from pages 2 and 138)

Consider the following scenario: The algebraic expert system, AES, knows a lot about the addition operation (+), but very little about multiplication (*). It is given the target problem:

$$\text{Find the } \underline{x} \text{ which solves } (= (* 3 4 \underline{x} 10) 600) \quad (A.21)$$

Knowing almost nothing about *, AES is unable to answer this query. It is then told the analogical hint,

$$* \text{ is like } +.$$

Its learning component (read “*NLAG*”) now determines that the **Group** abstraction is appropriate, because solving the analogous equation

$$\text{Find the } \underline{x} \text{ which solves } (= (+ 3 4 \underline{x} 10) 600) \quad (A.22)$$

requires the observations that

- + must be closed under the integers
[to guarantee that the result is defined],
- + must be associative
[to regroup the addends], and

- + must be invertible

[to cancel out the left and right terms in the \underline{x} -containing left side of the equation].

From *Inst-Source's* finding that $(\text{Group reals} + 0)$ holds, *Inst-Target* tries to satisfy the template $(\text{Group ?dom} * ?id)$, where $?dom$ and $?id$ are existential variables. Assuming its initial lexicon is sufficient, it (eventually) finds the bindings $\{?dom \mapsto \text{reals-0}, ?id \mapsto 1\}$, where reals-0 is the set of all non-zero numbers. The solution to this target problem requires this **Group** conjecture, together with the fact $(\text{InvertOp} * / \text{reciprocal})$, and the rule

```
(if (and (group $op $dom $id)
         (InvertOp $op $inv) )
    (if (and (Express $data as ($list-1 . $x . $list-2))
            (= ($op ($inv ($op $list-1)) $ans ($inv ($op . $list-2)) )
                $x) )
        (= ($op . $data) $ans) ) )
```

Figure A-4: Rule Used to Solve A Group Problem

(The $(\text{Express } \$data \text{ as } \dots)$ clause abbreviates the code which takes the $\$data$ list $(3\ 4\ \$x\ 10)$ and returns the binding $\{\$list-1 \mapsto (3\ 4), \$list-2 \mapsto (10)\}$, leaving the variable $\$x$ as $\$x$.)

Notice that the target problem (Equation A.21) is now solvable: i.e., an answer to $(= (*\ 3\ 4\ \underline{x}\ 10)\ 600)$ can be derived from the initial theory plus this analogically derived $(\text{Group reals-0} * 1)$ conjecture.

Note:7-3. Another Example of Analogy: Programming:

(from pages 2, 138 and A-54)

This description owes much to the Programmer's Apprentice work (PA) done at MIT by Drs Rich and Waters. In particular, their notion of a *program plan* is essentially the same as my abstraction: Each plan deals with a set of "rôles", which are formal parameters. The plan constrains these rôles, both individually

(by type information) and collectively. Furthermore, the idea of an overlay (see [Ric81, p66]) is almost identical to my notion of analogy: it is an explicit mapping between parts of the analogues.

Their articles discuss several possible types of problems that the PA system currently addresses, including analysis, verification and construction of programs. However, they have not (yet) focussed on how to use analogies for these tasks. The scenario below is one possible approach.

Imagine the initial PA' system is unable to solve a target problem, *PT*, which deals with program PgmA, because it knows too little about this program. It is then told the analogical hint, PgmA~PgmB, where PgmB is a better known program. This suggests that PA' transfer some of the PgmB facts over to PgmA, *mutatis mutandis*. But which PgmA facts, and how they should map over? Here we use the problem *PT*: We are seeking new PgmB facts which, once incorporated, provide a solution to the motivating problem *PT*.

Consider first the task of constructing a program. Here, the target problem is of the form "Construct a program which satisfies a particular behavioral specification". Within this framework, this task reduces to selecting the relevant plan(s) and then finding the appropriate instantiations for its various rôles.

As an example, imagine that all we knew about the target TimesAll subroutine is that it takes one argument, which is a list, and returns the product of the elements of that list. The *PT* problem, now, is to produce code which implements this behavior.

Of course, this specification is not sufficient — we need to know more about TimesAll before we can generate the appropriate code. Now comes the analogical hint:

TimesAll is like PlusAll. (A.23)

Of course, we know essentially everything about this source analogue, PlusAll. This includes not only its input/output specification (that it returns the sum of the elements of its single argument), but also its derivation as an instance of the Iterative-Aggregation plan, instantiated using SUM for the ".add" part,

temporally composed with an instance of “CAR+CDR+NULL”, for the iteration part. (The interested reader is referred to [Ric81, p236, p211] for a detailed description of these terms.)

This plan can be used as a commonable abstraction, suggesting that *NLAG* produce the *TimesAll* program by instantiating this *Iterative-Aggregation* plan using *PRODUCT* as the “*.add*” term. This leads to executable code which is essentially the same as *PlusAll*’s, changing only the + to * and the 0 to 1. That is, *TimesAll* should be written in the same language as *PlusAll* (i.e., *LISP*), keep its underlying code structure, expect the same type of “data structure” (i.e., list of numbers), *etc.*

We present some comments on this scenario before considering other ways of using analogies in this programming domain:

[1] This “re-instantiate the same plan” approach (*viz.*, go up to a common plan, then come down by instantiating it) solves the “Skolem variable” issue: that is, it tells us what else should change, once *SUM* is replaced with *PRODUCT*. Here, it tells us to replace that 0 with a 1. (Notice that this works even if the code happened to contain other spurious “0”s which should remain “0”; see Note:9-6.)

[2] In general, the hint may only suggest a partially instantiated program. “Partial” means that additional user-interaction may be necessary before *PA*’ can produce the desired code. That is, this hint may suggest a plan which is more general than the executable code.

“Suggest” means that these *facts* are really only *conjectures*, which may have to be retracted later. For example, the desired *TimesAll* subroutine may use a different data structure or perform a different (non-aggregation) type of accumulation. The retraction process can utilize the explicit derivation of the not-quite-correct *TimesAll* program. Hence the redesign may require viewing the desired *TimesAll* program as an instantiation of some other plan, one which is more abstract than *Iterative-Aggregation*. (This approach, of ascending the abstraction hierarchy, seems to work in a great many cases.)

[3] There are, of course, a number of other target programs which can be constructed using `PlusAll` — an iterative LISP procedure which operates on a list structure — as a source. Some might exhibit the same I/O behavior as `PlusAll`, but differ by

- being written in Pascal, not LISP;
- using an array, rather than a list structure;
- being written using tail recursion, rather than iteration.

All of these would use the same basic **Iterative-Aggregation** plan structure. Others might use related but different plans and perform different tasks. For example, the common abstraction might be a more abstract generalization of **Iterative-Aggregation**. Examples include programs which

- compute the *average* of the terms,
- add up only the *positive* terms of a list,
- recur through the full s-expression, adding up all nonNIL atoms.

While the focus, so far, has been on learning for the purpose of plan construction (read “plan generation”), there are other applications for such “analogically-motivated” new facts. Many follow directly from the realization that `PgmA` is an instance of a `PlanY`, where this plan is suggested by the analogical hint. For example, reiterating some of the themes which appeared in [Wat71,Ric81]:

Debugging One might search for the particular types of bugs known to be commonly found with instances of `PlanY`.

(Examples include the typical fence post problems, divide-by-0 situations, *etc.*)

Verification Specific plans may suggest that certain (loop) invariants be considered.

Optimization (More) efficient data structures may be associated with a given plan.

Explanation Describing PgmA as being like PgmB (since each is known to be an instance of some plan PlanY) may be a nice, succinct way to communicate its operations, structure, *etc.*

Note:7-4. Could NLAG Have Considered Group?

(from page 152)

It would have been cute if NLAG had conjectured something like

(group junctions flow-rate j-wc-a)

There is certainly some support for this candidate:

- $j\text{-wc-a} \in \text{junctions}$
- $\text{flow-rate} \in \text{Functions}$
- $\text{Domain}(\text{flow-rate}, 1) = \text{junctions}$

However, there is pretty strong evidence to the contrary as well:

- $\text{Domain}(\text{flow-rate}, 2) = \text{Devices} \neq \text{junctions}$
- $\text{Arity}(\text{flow-rate}) = 3$

Even though it looked feasible at first, the type information alone would eliminate it from contention. In fact, the H_{JK} heuristic prevented this Group abstraction from even being considered.

A.8 Notes from Chapter 8

Note:8-1. Semantic Definition of Novelty:

(See pages 27, 176 and 223)

This note states-expresses the novelty criterion of [GG83] within the current partial interpretation framework. That paper uses the $\text{New}_{PI}(Th, x, \sigma)$ relation to mean that the sentence σ is a *new* fact about the concept x with respect to the theory Th . Basically, this relation holds when adding σ to Th restricts the extensions allowed to the symbol x .

Now to express this within the partial interpretation notation: Let *Before* be the initial set of allowed interpretations, i.e., $Before = Allowed(\mathcal{RW}, Th)$. Similarly, let *After* be the set of interpretations allowed after adding the sentence σ to the theory Th : i.e., $After = Allowed(\mathcal{RW}, Th + \sigma)$. Standard semantic theory tells us that $After \subseteq Before$; as σ is assumed independent of Th , this inclusion is proper; i.e., $After \subset Before$.

[GG83] claims that σ is a new fact *about the symbol A* if A's set of possible extensions is reduced, under some assignment to the other symbols in \mathcal{L} . To capture this notion of *assignment to other symbols*, we need to partition the allowable interpretations, subtracting out the extension of a particular symbol. This requires a new operator for comparing interpretations. The expression $I =_A J$ means the interpretations I and J are identical, except possibly for their respective assignments to the symbol A:

Definition 36 $I =_A J \iff \forall R \in \mathcal{L}. R \neq A \Rightarrow \underline{R}^I = \underline{R}^J$

Since this $=_A$ operator is an equivalence operator, it induces a partitioning over the set of allowed interpretations. For any set of interpretations S , we define I^S/A to be the subset of S 's interpretations which disagree only on their assignments to A:

Definition 37 $I^S/A \doteq \{J \in S \mid J =_A I\}$

(Although it may not be obvious, I^S/A need not equal J^S/A for $I, J \in S$. This depends on I 's (respectively J 's) assignments to the other members of \mathcal{L} .)

In what follows, we consider just the set of initial interpretations, *Before*. We now use the $/A$ partitioning to define the set of partitions, $\{I^{Before}/A \mid I \in Before\}$.

For each such I^{Before}/A cluster, consider how it changes after adding σ to Th . That is, how is I^{Before}/A related to I^{After}/A ? First, this I^{After}/A is a smaller set

of interpretations. In fact, it is the intersection of I/A and $After$, i.e.,

$$I^{After}/A = (I^{Before}/A) \cap After. \quad (A.24)$$

Three cases might occur:

- $I^{After}/A = I^{Before}/A$

This means the set of allowed interpretations consistent with this assignment to $\mathcal{L} - \{A\}$ does not change, and hence, A 's extension is unchanged (for this particular assignment of the other (non- A) symbols).

- $I^{After}/A = \{\}$

This means that adding σ eliminated all interpretations based on this set of other assignments. That is, there is no longer any assignment to A which is consistent with the other assignments.

- $\{\} \subset I^{After}/A \subset I^{Before}/A$

This means there was an $J \in I^{Before}/A$ which is not in $J \in I^{After}/A$. This, in turn, means there is some assignment to A which was possible based on Th , but is no longer possible, using $Th + \sigma$. This is, again, with respect to I 's particular assignment to the other symbols of $\mathcal{L} - \{A\}$. Furthermore, this assignment is not inconsistent, since this I^{After}/A set is non-empty.

The final case above is exactly the condition specified (and justified) in [GG83].²⁶ This means we can express this New_{PI} relation as

Definition 38 $New_{PI}(Th, x, \sigma) \iff$

$$\exists I, J. I \in Allowed(\mathcal{RW}, Th) \ \& \ J \in Allowed(\mathcal{RW}, Th + \sigma) \Rightarrow \\ I =_x J \ \& \ I \notin Allowed(\mathcal{RW}, Th + \sigma).$$

The implied dependency on the \mathcal{RW} real world interpretation is not a major limitation. In fact, this assignment can be arbitrary.

²⁶A thorough justification of this criterion is beyond the scope of this dissertation; the intrigued reader should look in that article for such details.

How does this tie in with analogical inference? Notice the analogically inferred conjecture, $\varphi(A)$, is a new fact about the (formula) symbol φ since it definitely restricts the set of possible interpretation of this symbol. We now know that \boxed{A} must be a positive instance of φ ; i.e., $\boxed{A} \in \varphi^{\mathcal{R}\mathcal{W}'}$, using $\mathcal{R}\mathcal{W}'$ as the new base interpretation induced by $Th + \sigma$. This means $New_{PI}(Th, \varphi, \varphi(A))$ holds.

However, $\varphi(A)$ is not necessarily about the concept A . In fact, if A is a constant symbol with a fixed extension, we can guarantee that $\varphi(A)$ is not new for A , since there can be no New_{PI} sentences about A , period.

A.9 Notes from Chapter 9

Note:9-1. How can other analogy systems work at all?

(from page 192)

In light of all of these criticisms leveled at many existing systems, one might wonder why they have worked at all. There are several reasons. The most prominent is that most deals only with a small set of hand-picked facts, which have been (consciously or not) honed to work in this situation. Hence, these systems are spared the worry of seeking deducible but implicit facts, and of expanding the vocabulary.²⁷

A second feature is that abstractions naturally tend to fit into a hierarchical organization. Thus, even after finding a natural-seeming cut off point (i.e., after conjecturing the facts needed to establish one abstraction), it may be worth extending it later. There may be additional conjectures which follow from this same initial kernel and belong to that more specific abstraction. That is, given that the analogical inference $S(a_1, \dots, A, \dots, a_n)$ holds, it may be worth seeing if $S'(a_1, \dots, A, \dots, a_m)$ holds as well, where $S' \Rightarrow S$ (i.e., S' is a specialization of S in the sense that **Field** and **Abelian-Group** are each specializations of **Group**).

This explains why it can be worthwhile seeking additional relations even after

²⁷Of course, this same "you built it in" objection could be raised with respect to my *NLAG* system. The critical feature is that *NLAG* could accommodate such implicit information — this follows from its top-down mechanism.

one abstraction has been found. This is especially true if relatively little is actually stored about analogues, and if everything known corresponds to one particular perspective.

Note:9-2. Why are Analogies Considered Inexact?

(from pages 71, 189 and 216)

Subsection 9.2.1 provided one set of reasons why many people view analogy as an inherently slippery process. This note describes three other possible factors which may indirectly contribute to this view.²⁸

People may think of an analogy as being inexact because it is semantically “incorrect”. For example, the metaphoric statement, “John is a wolf”, is not correct, in the strict Tarskian sense. (*Cf.*, [Bea67], [Mar84] and various entries in [Ort79].) This ties in with the comment that analogy inference is a process of plausible reasoning, one which not guaranteed to be correct; see Note:1-2. This means that conjectures may have to be retracted or modified in light of additional information.

Pylyshyn also considers certain scientific uses of metaphor to be instances of “referential imprecision”, as opposed to “logical imprecision”. [Pyl79] describes how, for example, the current sense of the term “mass” emerged from this metaphoric process.

Another possible reason why analogies seem inexact is many analogies deal with second-order facts. For example, Hobbs discusses how the ⟨Pitcher, Batter, Ball, Hit⟩ quadruple can resemble the ⟨Congress, President Bill, Sign⟩ collection [Hob83b]. Intuitively, we want to express this in terms of a pair of related relationships, e.g., Hitting-Event(Pitcher, Batter, Ball) and

²⁸Yet a fourth complication stems from the distinction between the analogy relation and the analogical inference process. (Recall now the footnote adjoining Section 2.3.) The fact that the analogical inference is inexact does not mean that the resultant analogy (relationship) is inherently “fuzzy”.

Signing-Event(Congress, President, Bill), and then focus on the second-order correspondences between these relations, Hitting-Event and Signing-Event. Many confusions may arise from attempts to express this within a first-order framework.

Similarly, Burstein wants to associate the physical Put-In-Box containment relation with the computer-based Put-In-Var relation [Bur83b]. Using the second-order **StorageReIn** abstraction, this analogical connection is straightforward: the instantiations are **StorageReIn**(Put-In-Box, Physical-Object, Physical-Box) and **StorageReIn**(Put-In-Var, NumericValue, ComputerVariable). Burstein instead, considers this only a partial analogy, arguing that this connection requires a “more general version” of the relations Put-In-Box and Put-In-Var.

Note:9-3. Lexical Measures of Commonality are Inadequate:

(from page 190)

This note proposes a pair of lexical measures for commonality; then shows each to be deficient.

To motivate the first proposal, consider how the sentence $\varphi(a_1, \dots a_n)$ differs from $\varphi(b_1, \dots b_n)$. This can be expressed in terms of the lexical difference between the lists of terms $[a_1, \dots a_n]$ and $[b_1, \dots b_n]$. Perhaps commonality can also be expressed in terms of lexical constraints — in terms of the number of symbols which are common, rather than distinct? This suggests that commonality can be defined in terms of the *number of symbols* shared by the facts describing the two analogues. (Or, equivalently, in terms of the number of non-logical symbols which remain fixed across the symbol-to-symbol mapping.)

This seems to work when considering how much each of the pairs of sentences below says about Fido and Duke:

| Sent ₁ | ↦ | Sent ₂ | # Common Symbols |
|-------------------|---|---------------------|------------------|
| Mem(Fido, Dogs) | ↦ | Mem(Duke, Dogs) | [2] |
| Mem(Fido, Dogs) | ↦ | Mem(Felix, Animals) | [1] |
| Mem(Fido, Dogs) | ↦ | OwnedBy(Duke, Fred) | [0] |

Here, the number of symbols in common correctly mirrors our intuitions; e.g., “Mem(Fido, Dogs)” to “Mem(Duke, Dogs)” seems a better analogy than either of the other two. This suggests that we express commonality in terms of $\#Symbols_{PF}$, where

Definition 39 $\#Symbols_{PF}[A_0(A_1, \dots, a_n), B_0(B_1, \dots, b_n)] = \|\{i : A_i = B_i\}\|$

There are some problems with this measure. First, it would claim that $\text{Monoid}(\mathfrak{R}, +, 0) \mapsto \text{Monoid}(\mathfrak{S}, +, 0)$ is as good as $\text{Group}(\mathfrak{R}, +, 0) \mapsto \text{Group}(\mathfrak{S}, +, 0)$, as each has three symbols in common. Second (and worse), it would favor $\text{Monoid}(\mathfrak{R}, +, 0) \mapsto \text{Monoid}(\mathfrak{S}, +, 0)$ over $\text{Group}(\mathfrak{R}, +, 0) \mapsto \text{Group}(\mathbb{Z}_7, +_7, 0_7)$, since the first has three symbols in common, and the latter, only one.

So much for that proposal. Clearly a meaningful similarity measure should extend beyond the superficial syntax and deal with the underlying “meaning” of the statement. For example, even though Group is the only symbol superficially shared by the sentences $\text{Group}(\mathfrak{R}, +, 0)$ and $\text{Group}(\mathfrak{R}_0, *, 1)$, these facts clearly express quite a bit in common. Group ’s definition exposes several other invariants, *viz.*, the fact that both $\langle \mathfrak{R} + 0 \rangle$ and $\langle \mathfrak{R}_0 * 1 \rangle$ triples satisfy Group means that

- $\langle + \rangle$ and $\langle * \rangle$ are each instances of associative operators,
- $\langle + \mathfrak{R} \rangle$ and $\langle * \mathfrak{R}_0 \rangle$ are each instances of closed relations,
- $\langle + 0 \rangle$ and $\langle * 1 \rangle$ are each instances of operator identities, and
- $\langle + \mathfrak{R} \rangle$ and $\langle * \mathfrak{R}_0 \rangle$ are each instances of invertible operators.

Expressing this within a lexical framework: any two instances of Group share several “syntactic fix-points”, here in the form of the relationships Associative , Closed , Identity and Invertible . This suggests that the list of commoned symbols should include the symbols included in facts implied by the formula’s definition.

We can express this formally using

Definition 40 $\#Symbols_D[\phi_1, \phi_2] = \max_{\{\langle \chi_1 \ \chi_2 \rangle : \chi_1 \ \phi_1 \ \& \ \chi_2 \ \phi_2 \}} [\#Symbols_{PF}(\chi_1, \chi_2)]$

This certainly seems about right: As

$$\text{Group}(d, op, id) \iff \text{Monoid}(d, op, id) \ \& \ \text{Invertible}(op, d),$$

this measure seems to rank Group over Monoid.

Unfortunately, this “include other equivalent statements” criterion turns out to be meaningless. Every formula ϕ is equivalent to a related formula ϕ' with an arbitrarily large number of symbols, equal to the number of symbols defined in the language. [Proof: Define $\phi'(x) \stackrel{\text{def}}{\iff} \phi(x) \ \& \ \bigwedge_{c_i \in \mathcal{L}} (c_i = c_i)$.] This means that every formula is equivalent to one with $\|\mathcal{L}\|$ constants, rendering this measure worthless. So much for that second hypothesis.

Now consider why this “ $(c_i = c_i)$ ” trick feels so unintuitive. Clearly, the resultant ϕ' sentence is not *about* c_i . This suggests that we only consider symbols which the formula is *about*. This ties in with the *Triviality* criterion discussed in Section 2.4, and suggests that we are dealing with a semantic relationship.²⁹ Subsection 9.2.2 confirms this intuition, by presenting a meaningful measure of commonality using such a semantic basis.

Note:9-4. When does Maximal Commonality Force Co-referentiality?

(from page 193)

As an extreme case of analogy, it may be possible that everything known about the source analogue holds for the target analogue, *mutatis mutandis*. Does this necessarily mean that the analogues are co-referential: i.e., that they refer to the same object? How does this situation relate to the claim that some formula is maximally specific? In particular, does the existence of such a maximally specific formula guarantee both concepts to refer to the same object? This note answers these questions.

These issues reduce to the question of what the initial theory *Th* can distinguish. The maximally specific formula can be a formula whose domain is that

²⁹The inadequacies of these syntactic approaches is consistent with the Logical Positivist position that logical and mathematical statements reveal only the basic structure of the language, but need not be descriptive of the physical world [Goe85]. Of course, these logical positivists would not think a semantic approach would work either...

single element. While Th may be unable to distinguish among the terms which satisfy this formula, it does not mean the two concepts are co-referential: For example, imagine that $Th = \{ \text{associative}(+) \}$, and we are given “* is like +” as an analogical hint. Here the only (and hence the best) thing we can say about * is that it is associative, $\text{associative}(*)$. This, of course, does not mean + and * are co-referential, but only that they are indistinguishable.

Of course, simply finding the maximally specific formula does not mean that the two analogues are even indistinguishable: The theory Th might begin by knowing that some B-fact does not hold for A, meaning B and A cannot refer to the same object. For example, consider

$$Th = \{ \text{UniqueFactorization}(*) \neg \text{UniqueFactorization}(+) \},$$

or

$$Th = \{ \text{DistributesOver}(+ *), \neg \text{DistributesOver}(+ +) \};$$

or even

$$Th = \{ (+ \neq *), \neg(+ \neq +) \}.$$

Note:9-5. Other times when H_{MSA} is Inappropriate:

(from pages 194 and 198)

The maximal specificity rule, H_{MSA} (see Heuristic 7), prefers the formula φ_1 over φ_2 whenever φ_1 is more specific. There are, however, times when φ_2 might seem better than φ_1 . (E.g., where we would want to consider **Monoid** before **Group**.) For example, due to φ_1 's additional information, it might not have a derivable instantiation in the source domain, or may have no consistent instantiation in the target domain. (Since this means that φ_1 is not a legal candidate, this situation does not violate H_{MSA} .)

The other (pseudo)counter-example occurs when the user exercises his right to specifically reject a qualifying analogy, $\varphi_1(a_1, \dots, A, \dots, a_n)$. (For example, he, but not Th , may know enough to suspect that integers are not invertible with respect to multiplication, and so might accept the conjecture **Monoid**($Z, *, 1$) but reject **Group**($Z, *, 1$.)

```
(defun PlusAll (a)
  (do ((index 1 (+ index 1))
      (total 0 (+ total (arrayfetch a index))) )
    ((greaterp index (arraylength a))
     total))
  ))
```

Figure A-5: Definition of PlusAll

Note:9-6. Exact Formulation: TimesAll~PlusAll:

(from pages 2, 203 and A-43)³⁰

Subsection 9.3.1 describes one situation where the exact form of the formula matters. This note presents another example of this situation. This presentation describes the source information as a collection of sentences (*i.e.*, use the *Analogy_T*'s formulation). This formulation partially explains why it is so tempting to blur the distinction between what the analogy keeps invariant and what changes from one analogue to the other. (Since the program code is already expressed in a formal language, this note does not rewrite it into the obvious predicate calculus notation.)

Suppose you want to write a program which computes the product of the elements of an array, and you are told that this TimesAll subroutine should be like the existent PlusAll program shown in Figure A-5. The obvious first step is to substitute "*" for "+" throughout the program code. Of course, this alone is not sufficient: these two different operators require different identities (here, +'s "0" should be mapped to *'s "1") and the old program name should be changed to a new one. This leads to the { + \mapsto *, 0 \mapsto 1, PlusAll \mapsto TimesAll } substitution. If we apply this to entire body of PlusAll's code, we arrive at the faulty program shown in Figure A-6.

The problem is the spurious "+" which appears in the code used to increment the index counter; since this is transformed to "*", the index counter is

³⁰This work was motivated by the Programmers Apprentice work, [Ric81,Wat71,Shr79,Bro81,Cyp82]. See also Note:7-3.

```
(defun TimesAll (a)
  (do ((index 1 (* index 1))
      (total 1 (* total (arrayfetch a index))) )
    ((greaterp index (arraylength a))
     total))
  ))
```

Figure A-6: Faulty Definition of TimesAll

never incremented. This substitution would work correctly if we rewrote PlusAll to use “(add1 index)” rather than the current “(+ index 1)”. This leads to PlusAll-add1 of Figure A-7.

```
(defun PlusAll-add1 (a)
  (do ((index 1 (add1 index))
      (total 0 (+ total (arrayfetch a index))) )
    ((greaterp index (arraylength a))
     total))
  ))
```

Figure A-7: Definition of PlusAll-add1

Realize these two subroutines are (behaviorally) equivalent, since

Definition 41 $\forall x$ (add1 x) $\stackrel{\text{def}}{=} (+ x 1)$.

This example shows a situation where “(add1 index)” works but “(+ x 1)” fails. The reverse situation can happen as well. Suppose we want to generate a subroutine which adds up the values of every second element of the array; the obvious goal is the Plus-2-All subroutine of Figure A-8. Applying the simple $\{1 \mapsto 2, \dots\}$ mapping to Figure A-5’s PlusAll code works perfectly, changing both “1”s into “2”s. Notice that PlusAll-add1’s code is not desirable; here the “1” in “(+ x 1)” must be explicit. (By contrast, it is only implicit in PlusAll-add1’s “add1” component.)

```
(defun Plus-2-All (a)
  (do ((index 2 (+ index 2))
      (total 0 (+ total (arrayfetch a index))) )
    ((greaterp index (arraylength a))
     total))
  ))
```

Figure A-8: Definition of Plus-2-All

As a final variant on this theme, imagine all we know about some unknown program was that it resembles PlusAll behaviorally (here, assume all we know about PlusAll is that it adds up the elements of a list), but it differs (structurally) by a $\{1 \mapsto 2\}$ mapping. What is the behavior of that resultant program?

If PlusAll is the structure to be altered, the result would be Plus-2-All, which adds every second element. What if PlusAll-add1 was the altered structure? Here the result would be a subroutine which adds up all but the first element of the array. (This is assuming we did not map PlusAll-add1's "add1" into an "add2" subroutine.)

It is easy to argue for either of these answers; in different situations either of these would be appropriate. This final example shows there may be many different meanings of a given "A is like B" hint, and which interpretation is selected depends not only on the information included in the mapping's domain, but on the formulation of that information as well. This also means that the same information (here, "... (+ index 1) ..." versus "... (add1 index) ...") may lead to different meanings, under different analogy contexts.

Of course, this versatility requires that we consider the deductive closure of the starting theory. That is, we should be able to produce either interpretation for the analogy given only the above "+" fact and Definition 41's definition of add1, even if this required deriving the needed (add1 index). (Point NotStd2 in Note:2-5 reiterates this point.)

Note:9-7. No need to generate new symbol to Expose Hidden Value:

(from page 208)

We never need to generate a new symbol to expose a hidden value to some symbol-symbol transformation.

Proof: The only way we can know that some relation is hiding a value is if there is some defining sentence, as Definition 31 was for the `ControlledByCurrent` relation. In this case, it would have been sufficient to use that defining form, `CValve`, rather than `ControlledByCurrent`. Of course, this alternate form may contain spurious symbols which must be “reformulated” away. The point here is that the cause of this reformulation was *not* in the first step (of exposing some constant), but rather in eliminating some constant (here, the one inadvertently exposed in the first step).

A.10 Notes from Chapter 10

Note:10-1. Diffuse “Abstractionness” Measure:

(from pages 55, 162 and 222)

Most recognized abstractions tend to be fairly specific relations, often involving many arguments. This note begins with an intuitive argument for why this may be true and concludes with a proposal for a non-boolean measure for re-usability.

At one extreme, consider a formula with a single argument, consisting of a single clause: e.g., $\varphi(x) = \text{Kirchoff1}(x)$. While such a formula might not be useful in-and-of-itself (that is, it may not be sufficient to solve some standard problem), it might prove a useful building block of other more useful formulae. This general claim holds for formulae consisting of relatively few clauses:³¹ that is, there may be some utility to supplying names to “small” general relations which are not, by themselves, abstractions. However, the number of formulae with n clauses grows combinatorially with n , and it becomes increasingly less cost-effective to continue naming all such specific formulae. This suggests that people begin focusing their names to relatively few of these n -clause formulae, selecting only the ones which have proven to be directly useful. This argues that,

³¹Here “clause” refers to any member of the leaf decomposition of the formula; see Subsection 6.2.3.

for large n , the only named formulae of n clauses are the relevant ones, i.e., are the ones which constitute coherent and complete clusters of clauses. This implies that any relation (read “reified formula”) which corresponds to a non-trivial cluster of clauses will be useful: i.e., an abstraction.

Now consider the further heuristic that the number of arguments of a relation tends to be positively correlated with the number of its clauses. This suggests that m -ary relations tend to be relevant, for large m . This provides an intuitive justification for the claim that m -ary relations (for $m > 1$) are more significant than unary features. This principle is discussed in [Gen80b] and [Car81b].

Despite these arguments, the current *NLAG* system applies a boolean condition: certain formulae are deemed re-usable (these are the atomic formulae which are based on the relations tagged as “abstractions”), and every other formula is considered totally un-re-usable. The “Find Maximally General Common Relation” analogy system proposed in Section 7.6.4 also uses a boolean measure of re-usability, *viz.*, one which ranks all and only relations as re-usable.

Perhaps we should rank some relations as more re-usable than others, via a multi-valued (or even continuous) “re-usability” measure? This measure would probably rank certain relations very highly (like our canonical abstractions, **RKK** and **Group**), and others less highly, like the simple unary relations (e.g., Kirchoff1). Most likely, it would leave arbitrary, un-reified formulae at 0.

The future analogy system would take this measure into account when finding and ranking useful analogies. This proposal, however, is but fodder for future work.

Note:10-2. How to Variabilize:

(from page 223)

Consider again OhmsLaw, now written out in prefix notation:

$$\begin{aligned} \forall d \text{ (Resistors } d) &\Rightarrow \\ (= \text{ (VoltageDrop (junction } d \ 1), (\text{junction } d \ 2), [d]) & \quad (A.25) \\ (* \text{ (Current (junction } d \ 1) } d) \text{ (Resistance } d)) &) \end{aligned}$$

Which terms should be variabilized? We can think of *every* term as a variable (see Equation A.26), and consider what constraints it must satisfy. (The result is something like a Ramsey sentence, *cf.* [Gly80].) Given our partial interpretation semantics, some symbols may be forced to be a single value — e.g., perhaps the only interpretation for ?7 is the multiplication symbol, *. Others, like the ?8 (*née* Current), should clearly be variables, which can have multiple instantiations. But what about ?4 — must it be junctions? It is in the middle; as we discussed in Note:4-1, we may want to talk about freeway-intersections.

$$\begin{aligned} \forall d \quad (?1 \ d) \Rightarrow \\ (?2 \ (?3 \ (?4 \ d \ ?5), \ (?4 \ d \ ?6), \ [d]) \quad (A.26) \\ (?7 \ (?8 \ (?4 \ d \ ?5) \ d) \ (?9 \ d) \) \) \end{aligned}$$

Appendix B

Data for the Experiments

This appendix includes details of the *NLAG* system and of the experiments run. SubAppendix B.1 presents *NLAG*'s actual code, embellishing Chapter 6's description. The other subappendices provide data associated with the experiments described in Chapter 7. SubAppendix B.2 explicates the contents of the initial theory, *Th_{CF}*. SubAppendix B.3 provides an annotated run of the *NLAG* system, working through the demonstration sketched in Section 7.2. SubAppendix B.4 describes the additional facts added to the initial theory for the other studies, discussed in Subsection 7.2.2 and Sections 7.3 through 7.5, and the results obtained.

B.1 Actual *NLAG* Code

This section presents *NLAG*'s actual code, listed in a breadth-first order. This system of rules is run by the MRS backward-chaining routine, *BC*, as called by *TRUEP*. (This actually uses a slight variant of the standard version of *BC*, modified only by the additions discussed in Chapter 6.)

The purpose of this code is to convey the basic organization of the modules and to illustrate the flow of information. It is not to present a specific detailed account of what each module does. This is why many nuances of the code are not explained in detail. For example, each of the relation symbols which is underlined below is "performed" by an attached procedure. A coarse description of the associated LISP

code appears at the end, in alphabetical order.

```
(if (and (ComAbs $im $Purpose $Abst $fs $ft $th)
         (Verify $Purpose $th ($Abst . $ft) ))
     (NLAG $IM $Purpose ($Abst . $fs) ($Abst . $ft) $th))

(if (and (Find-Kernel $Map $Purpose $facts $t1)
         (Inst-Source $facts $t1 ($Abst . $fs))
         (Inst-Target $Map $Purpose $Abst $fs $ft $t2))
     (ComAbs $Map $Purpose $Abst $fs $ft $t2))

(if (xor (H-jk1 $M $Q $facts $Th)
        (AllLexIn $M $Q $facts $Th))
     (Find-Kernel $M (Query $Q) $facts $Th))

(if (and (rsublst $M $Q $Q2)
         (unknown (= $Q $Q2))
         (Map-Domain $M $d)
         (Find-Theory $d $Th)
         (Activate-Theory $Th)
         (merge-of $f (consider-facts $v $Q2 $f) $facts)
         (DeActivate-Theory $Th) )
     (H-jk $M $Q $facts $Th))

(if (and (current-theory $th)
         (merge-of $f (and (Map-Domain $M $d) (member $e $d)
                          (pr-facts $e $f))
                   $facts) )
     (AllLexIn $M $Q $facts $th))

(if (and (useful-rules $q $r)
         (just $r $x . $justs))
     (consider-facts (rule $r) $q $justs))

(if (and (useful-constraints $q $r)
```

¹This is the H_{JK} heuristic.

```

      (just $r $x . $justs))
    (consider-facts (constraint $r) $q $justs))

  (if (FC-FindN $facts $th ($Abst . $fs))
      (Inst-Source $facts $th ($Abst . $fs)))

  (Generator-Lisp FC-FindN FC-Find-1 FC-Find-Next $i $th $o)

  (if (and (make-tgt $Map ($Abst . $fs) $ftQ $nm)
          (Conj-TruepN $ftQ $a1 $t1)
          (Resolve-Exists $t1 $a1 $Purpose $t2 $a2)
          (plug $ftQ $a2 ($Abst . $ft)))
      (Inst-Target $Map $Purpose $Abst $fs $ft $t2))

  (Generator-Lisp Conj-TruepN Conj-Truep-1 Conj-Truep-Next $x $a1 $th)

  (if (and (Get-Exists $a1 $vs)
          (Get-Relevant-Facts $vs $t1 t $facts)
          (Satisfy-Facts $vs $facts $Purpose $a2)
          (Compose-Map $a1 $a2 $a3))
      (Resolve-Exists $t1 $a1 $Purpose ($t1 . $a2) $a3))

  (if (and (Useful $Purpose $th ($Abst . $ft) )
          (Check-Consistent $th)
          (Check-Dep-Th $th) )
      (Verify $Purpose $th ($Abst . $ft) ))

  (if (truep-given $Q $th $a1)
      (Useful (Query $Q) $th $ca))

  (if (and (Split-Facts $vs $facts ($mult . $ps))
          (Single-Props $ps $pc1)
          (Order-Cands $pc1 $Purpose $pc2)
          (Find-Best-Cand $pc2 $a2)
          (Consistent-List $mult $a2) )
      (Satisfy-Facts $vs $facts $Purpose $a2))

  (if (and (Weight-By-Lookup $pc1 1 $pc2)

```

```

(Weight-By-Assoc $pc2 $Query 5 $pc3)
(Weight-By-CommonTheory $pc3 $Query 10 $pc4)
(Order-All $pc4 $pc5))
(Order-Cands $pc1 (Query $Query) $pc5))2

```

Many of the above relations — including `unknown`, `if`, `and`, `member`, `Generator-Lisp` — are defined in MRS. The interested reader is referred to [Rus85] for a description of their operations. (That manual also describes the notational conventions used, including prefacing the name of each variable with a dollar-sign and using the dot syntax (`(a . $b)`) to refer to a sublist.)

Other relations perform exactly what their name implies. For example, `“xor”` is exclusive or, of course; and `plug` corresponds to the MRS subroutine of the same name. Similarly, `merge-of` is like MRS’s `bagof`, but it assumes that each binding is a ordered list and merges them together.

Now for the other relations:

`Activate-Theory`, `DeActivate-Theory`: These relations work by side effect, causing an MRS theory (its sole argument) to be activated or deactivated, respectively.

`(Check-Consistent $th)` holds when the theory bound to `$th` is found to be consistent. This check involves individually removing each of the conjectures associated with this theory, and attempting to prove its negation in that diminished context. (This process assumes that the base theory (*sans* conjectures) is consistent, and that the (`TRUEP` '(not σ)) procedure call is effective.)

`(Check-Dep-Th $th)` holds when the user approves of all of the conjectures associated with the theory bound to `$th`.

`(Compose-Map $a1 $a2 $a3)` means that `$a3` is bound to the mapping derived by composing the mappings associated with `$a1` and `$a2`. Hence, `(Compose-Map`

²This rule implements the ordering described on page 127 in Subsection 6.2.3.

((current . flow-rate)) ((?2 . pressure-drop)) ((current . flow-rate) (?2 . pressure-drop)) holds.

Conj-Truep-1, Conj-Truep-Next: These relations implement the *CTruep* facility; see Subsection 6.2.3. Conj-Truep-1 sets up the context and returns the first answer; Conj-Truep-Next returns the subsequent values.

(Consistent-List \$mult \$a2) first plugs each of the propositions bound to \$mult with the substitution bound to \$a2. This relationship holds when each of these resultant propositions is found to be consistent with the current theory.

(Current-Theory \$th) means that \$th is bound to the currently active theory.

FC-Find-1, FC-Find-Next: These relations implement the *FC-Find* facility; see Subsection 6.2.2. FC-Find-1 sets up the context and returns the first answer; FC-Find-Next returns the subsequent values.

(Find-Best-Cand \$pc \$a1) holds when \$a1 is bound to the best remaining candidate for a binding of the existential variables. This requires examining the cross product of the possible values for each variable; this is encoded by \$pc. This relation is implemented as a generator, returning a single value each time.

(Find-Theory \$d \$Th) means that \$Th is bound to the most general theory associated with the elements of the list \$d. Here, (Find-Theory (current) electric) holds.

(Get-Exists \$a1 \$vs) means that \$vs is bound to the list of existential variables which appear in the value associated with \$a1. Hence, (Get-Exists (rkk flow-rate ?2 ?3 ?4) (?2 ?3 ?4)) holds.

(Get-Relevant-Facts \$vs \$t1 t \$facts) means that \$facts is bound to the list of derivable facts which each lexically contain one of the existential variables which appear in the value associated with \$vs. \$t1 is bound to the most complete (temporary) theory which contains those propositions.

(make-tgt \$map \$fsQ \$ftQ \$nm) means that \$ftQ is bound to target abstraction instance. Hence, (make-tgt ((current . flow-rate)) (rkk current voltage-drop resistance resistors) (rkk flow-rate ?2 ?3 ?4) \$n) holds.

(Map-Domain \$m \$d) means that \$d is bound to the domain of the mapping \$m. Hence, (Map-Domain ((current . flow-rate)) (current)) holds.

(Order-All \$pc4 \$pc5) holds when the sorted form of the propositions bound to \$pc4 is bound to \$pc. Each entry associated with \$pc4 consists of both a proposition and a numeric value; the sorting is based on that numeric value.

(Pr-Facts \$e \$f) means that \$f is bound to the set of facts currently known which contain the symbol to which \$e is bound.

(rsublst \$m \$q \$q2) means that \$q2 is bound to the result of applying the inverse of the mapping \$m to the expression \$q. Hence, (rsublst ((current . flow-rate)) (flow-rate j-wc-a pipe1 s0 \$fr) (current j-wc-a pipe1 s0 \$fr)) holds.

Single-Props: finds all bindings which are consistent with the single-variable expressions. This corresponds to the first part of *Resolve-Exists*, and thence to the "1-Consist" tests reported later.

Split-Facts: is used to divide up the facts which deal with the existential variables.

(truep-given \$q \$th \$al) holds when the query bound to \$q returns an answer (bound to \$al) when run in the context of the theory bound to \$th.

Useful-Rules, Useful-Constraints: These relations are used to find the rules (respectively, constraints) which might be used to determine the value of its first argument.

Overall Storage Requirements:

A total of 12 theories (each in its own file) are used in the experiments; these

require 116,741 bytes. *NLAG* also had access to an additional 12 theories, using an additional 39,672 bytes. These other theories provided additional test cases to the *NLAG* system. Some encode other lumped element linear systems. Others hold other types of information relating to current, like wave-guides, as well as less related things, e.g., matrices and programs (the latter in a program-apprentice style).

An additional 154,985 bytes of source code, divided into 18 files, went into overhead facilities (including some of the features described in Section 6.2).

The entire system lives on Diablo, a VAX 11/780 running the Unix^(tm) operating system, Version 4.2.

B.2 Initial Theory

We divide the initial knowledge base into four categories. The first, shown in SubAppendix B.2.1, describes general facts; this information is applicable to almost all domains (and in particular, to the domains of electricity and hydraulics). It includes the definitions of all the abstractions present. The second, shown in SubAppendix B.2.2, lists the facts initially known about the source domain of electricity, EC. Subappendix B.2.3 shows the few facts initially known about the target domain of hydraulics, FS. (The analogically derived conjectures extend this domain.) Finally, SubAppendix B.2.4 lists the facts which define the target problem, "Find the flowrate".

In general, only the "primitive" facts are shown. Once these propositions are asserted, other derivable facts may be stored as well. (As the one exception, some of the facts shown at the end of SubAppendix B.2.2 are derived.)

B.2.1 General Information

This subappendix summarizes the general information included in the initial theory. Using MRS's theory context mechanism, this information is partitioned into several logical "theories", each in its own physical file. After sketching these theories, we

focus on the least general of these general theories, LELS.

Hierarchy of Theories: From the most general down, these theories are

global holds the information needed for MRS to work. (Much of it comes with the core MRS system.) It includes the basic hierarchy of terms (e.g., (subclass function relations)), descriptions of the core MRS relations (e.g., "if", "=", "member" and "function"), and the meta-level facts which describes much of MRS's processing (e.g., "tostash" facts). It also houses the essential facts about things like arithmetic operators (e.g., "+"), files and theories. This requires about 450 propositions. An additional 145 propositions are used to store the context associated with each term. (This is used especially by Generator#2; see Subsection 7.2.2.)

domain holds both the domain specifications for the various relations and the information required to use these facts. An example fact is (domain-spec domain relations integers classes), which means that domain is a relation which takes three arguments; the first is a relation, the second is an integer and the third, a class. (This theory includes about 30 propositions.)

functional holds various relations used to manipulate relations. For example, it defines the starring relation, which could be used to generate the ancestor relation from parent. (This theory includes about 40 propositions.)

class holds a more elaborate hierarchy of terms and additional facts about the connectives, e.g., subclass*. (This particular function is defined using the assertion (starring subclass subclass*.) (This theory includes over 100 propositions.)

com-abs holds the two facts which define the abstraction relation.

genl-info holds general information about the world. (By contrast, most of the facts described above were about MRS operators and relations.) This includes

general facts about physical objects, people, time, composition of objects, *etc.* It also describes some mathematical entities, like `sin` and `cos`. (This theory includes about 75 propositions.)

`lels` holds general facts about lumped element linear systems. Most of its 127 facts are shown below.

(We later consider two additional theories, *viz.*, algebra and number. These are discussed in Section B.4.)

LELS Theory:

Facts about port, junctions, *etc.*

(mem port functions)

(domain-spec port devices integers junctions)

(mem junctions physical-objects)

(mem s0 states)

General typing information about the abstractions and related relations.

(domain-spec kirchoff1 functions)

(domain-spec kirchoff2 functions)

(domain-spec conserved-thru functions classes)

(domain-spec ohmslaw functions functions functions classes)

(domain-spec capac-Law functions functions functions classes)

(domain-spec induct-Law functions functions functions classes)

(mem Drop-Of functions)

(domain-spec Drop-Of functions functions)

(domain-spec Wire-Like classes classes)

(mem kk abstract-relns)³

³This forward chains to assert (abstraction kk).

```

(domain-spec kk functions functions)

(mem rkk abstract-relns)
(domain-spec rkk functions functions functions classes)

(mem ckk abstract-relns)
(domain-spec ckk functions functions functions classes)

(mem lkk abstract-relns)
(domain-spec lkk functions functions functions classes)

```

Rules which interrelate these concepts:

(Page A-74 shows some conclusions derived from these rules.)

```

(if (and (domain-spec $x . $l)
         (member states $l))
    (state-dependent $x))

(if (and (port $x 1 $j1) (port $y 1 $j1)
         (port $x 2 $j2) (port $y 2 $j2))
    (connected-in $x $y parallel))

(if (and (port $x 2 $j1) (port $y 1 $j1))
    (connected-in $x $y series))

(if (drop-of $ct-1 $ct)
    (constraint (+ $v2 $vd $v1)4
               (and (port $x 1 $j1)
                    (port $x 2 $j2))
               ($ct $j1 $j2 $s $vd)
               ($ct-1 $j1 $s $v1)
               ($ct-1 $j2 $s $v2)) )

(if (kirchoff1 $tt)
    (constraint (num-= (+ . $vs) 0.0)

```

```

      (and (bagof $d (port $d $i $j) $ds)
           (form-list $d $ds ($tt $j $d $s +NewVar+) $curs)
           (finals $curs $vs))5
      ($tt $j $x $s $e) . $curs))

(if (conserved-thru $tt $load)
    (constraint (+ $c1 $c2 0.0)
                (and (mem $x $load)
                     (port $x $i $j1)
                     (port $x $k $j2)
                     (< $i $k))
                  ($tt $j1 $x $s $c1)
                  ($tt $j2 $x $s $c2)))

(if (ohmslaw $tt $ct $rt $load)
    (constraint (* $c $r $vd)
                (and (port $x 1 $j1)
                     (mem $x $load)
                     (port $x 2 $j2))
                  ($rt $x $r)
                  ($ct $j1 $j2 $s $vd)
                  ($tt $j1 $x $s $c)) )

(if (induct-Law $tt $ct $indt $ind-load)
    (constraint (* $l $vd $di)
                (and (port $x 1 $j1)
                     (mem $x $ind-load)
                     (port $x 2 $j2))
                  ($indt $x $l)
                  ($ct $j1 $j2 $s $vd)

```

⁴See Point MRS2 in Subsection 6.2.4 for a description of this constraint syntax.

⁵These clauses are used to bind the variable \$curs to a set of propositions of the form, e.g., ((flowrate j-a d1 s3 \$1) (flowrate j-a d2 s3 \$2) ... (flowrate j-a dn s3 \$n)), where j-a is the junction shared by the devices d1 through dn, during situation s3. The variable \$vs is bound to the list of final variables, here, to (\$1 \$2 ... \$n).

```

($tt $j1 $x $s $i)
(d-dt $tt $j1 $x $s $di)) )6

(if (capac-Law $tt $ct $capt $scap-load)
  (constraint (* $c $dv $i)
    (and (port $x 1 $j1)
      (mem $x $scap-load)
      (port $x 2 $j2))
    ($capt $x $c1)
    (minus $c1 $c)
    ($ct $j1 $j2 $s $v)
    ($tt $j1 $x $s $i)
    (d-dt $ct $j1 $j2 $s $di)) )

```

The rules below convey type information.

```

(if (KK $tt $ct)
  (and (mem $ct functions)
    (domain-spec $ct junctions junctions states numbers)
    (mem $tt functions)
    (domain-spec $tt junctions devices states numbers)))

(if (ohmslaw $tt $ct $rt $load)
  (and (mem $rt functions)
    (domain-spec $rt $load numbers)
    (subclass* $load devices))

(if (capac-Law $tt $ct $capt $scap-load)
  (and (mem $capt functions)
    (domain-spec $capt $scap-load states numbers)
    (subclass* $scap-load devices)) )

```

⁶This d-dt relation is used to express derivative with respect to time. It was never implemented in its full generality; instead, it utilized only a few specific patterns.

```
(if (induct-Law $tt $ct $indt $ind-load)
    (and (mem $indt functions)
         (domain-spec $indt $ind-load states numbers)
         (subclass* $ind-load devices)) )
```

Finally, the definitions of the abstractions:

```
(if (and (kirchoff1 $tt)
         (kirchoff2 $ct) )
    (kk $tt $ct))

(if (and (kk $tt $ct)
         (conserved-thru $tt $load)
         (ohmslaw $tt $ct $rt $load) )
    (rkk $tt $ct $rt $load))

(if (and (kk $tt $ct)
         (capac-Law $tt $ct $capt $cap-load) )
    (ckk $tt $ct $capt $cap-load))

(if (and (kk $tt $ct)
         (induct-Law $tt $ct $indt $ind-load) )
    (lkk $tt $ct $indt $ind-load))
```

B.2.2 Initial Electric Facts, EC

```
(kirchoff1 current)
(kirchoff2 voltage-drop)
(conserved-thru current Elect-Devices)
(ohmslaw current voltage-drop resistance Elect-Devices)
(capac-Law current voltage-drop capacitance capacitors)
(induct-Law current voltage-drop inductance inductors)
```

(mem battery classes)

(mem resistor classes)

(mem wire classes)

(mem capacitor classes)

(mem inductor classes)

(mem ampere units)

(mem volt units)

(mem coulomb units)

(mem ohm units)

(mem voltage functions)

(domain-spec voltage junction states numbers)

(units-of voltage 3 volts)

(mem voltage-drop functions)

(domain-spec voltage-drop junction junction states numbers)

(units-of voltage-drop 4 volts)

(mem current functions)

(domain-spec current junction device states numbers)

(units-of current 4 amperes)

(mem resistance functions)

(domain-spec resistance resistor numbers)

(units-of resistance 2 ohms)

(mem capacitance functions)

(domain-spec capacitance capacitor numbers)

(units-of capacitance 2 farads)

(mem inductance functions)

(domain-spec inductance inductor numbers)

(units-of inductance 2 henrys)

(mem volt-batt functions)

(domain-spec volt-batt battery numbers)

(units-of volt-batt 2 volts)

(dimensions amperes (divides coulombs seconds))⁷

(dimensions volts (divides (times newtons meters) coulombs))

(dimensions joules (times newtons meters))

(dimensions ohms (divides volts amperes))

(UnitOf volts voltage)

(UnitOf ohms resistance)

(UnitOf amperes current)

(UnitOf coulombs quantity)

(wire-like wires Elect-Devices)

(subclass wires Elect-Devices)

(subclass resistors Elect-Devices)

(drop-of voltage voltage-drop)

Rules:

```
(if (and (mem $b battery)
         (port $b 1 $j1)
         (port $b 2 $j2)
         (volt-batt $b $v))
    (voltage-drop $j1 $j2 $s $v))
```

The facts below are all derived by forward chaining from the facts and rules above.

```
(if (and (mem $w wires)
         (port $w 1 $j1)
         (port $w 2 $j2))
    (voltage-drop $j1 $j2 $s 0.0))
```

⁷This symbolically describes the dimensions of these units.

```
(constraint (+ $c1 $c2 0.0)
  (and (mem $x elect-devices)
    (port $x $i $j1)
    (port $x $k $j2)
    (< $i $k))
  (current $j1 $x $s $c1)
  (current $j2 $x $s $c2))
```

```
(constraint (* $c $r $vd)
  (and (port $x 1 $j1)
    (mem $x elect-devices)
    (port $x 2 $j2))
  (resistance $x $r)
  (voltage-drop $j1 $j2 $s $vd)
  (current $j1 $x $s $c))
```

```
(constraint (num-= (+ . $vs) 0.0)
  (and (bagof $d (port $d $i $j) $ds)
    (form-list $d $ds (current $j $d $s +NewVar+) $curs)
    (finals $curs $vs))
  (current $j $x $s $e) . $curs) ))
```

B.2.3 Initial Hydraulics Facts, FS

```
(subclass water-devices devices)
(subclass pumps water-devices)
(subclass tanks water-devices)
(subclass tubes water-devices)
(subclass pipes tubes)
(subclass thin-pipes tubes)
(subclass orifices tubes)

(mem pressure functions)
```

(domain-spec pressure junctions states numbers)

(units-of pressure 3 pressure-units)

(mem pressure-drop functions)

(domain-spec pressure-drop junctions junctions states numbers)

(units-of pressure-drop 4 pressure-units)

(mem flow-rate functions)

(domain-spec flow-rate junctions devices states numbers)

(units-of flow-rate 4 meters³-by-sec)

(mem cross-section functions)

(domain-spec cross-section tubes numbers)

(units-of cross-section 2 square-meters)

(mem pipe-charact functions)

(domain-spec pipe-charact pipes numbers)

(mem Cd-of functions)

(domain-spec Cd-of tubes numbers)

(units-of Cd-of 2 dimension-less)

(mem pipe-shape functions)

(domain-spec pipe-shape Pipes shapes)

(mem water-height functions)

(domain-spec water-height tanks states numbers)

(units-of water-height 3 meters)

(mem Pump-Press functions)

(domain-spec Pump-Press Pumps numbers)

(units-of Pump-Press 2 pressure-units)

(if (and (cross-section \$p \$cs)

(cd-of \$p \$cd)

(/ \$cd \$cs \$pc))

(pipe-charact \$p \$pc))⁸

```

(mem pipe1 pipes)
(mem pipe2 pipes)
(mem pump1 pumps)
(mem j-wc-a junctions)
(mem j-wc-b junctions)

(if (mem $p pipes)
    (pipe-shape $p straight))9

```

B.3 Run#1-1 Data

This subappendix elaborates the description of the first demonstration, summarized in Section 7.2. SubAppendix B.3.1 first shows a transcript of the actual run. Subsequent subappendices present the specific details suggested by parts of the transcript. SubAppendix B.3.2 describes the set of kernel facts returned by *FindKernel*. SubAppendix B.3.3 presents the actual facts proposed for each instantiation of each abstraction considered (rkk, kk, lkk and ckk). In addition to these conjectures themselves, it also includes their entailments — i.e., the addition consequences found by forward-chaining from them. (This is the information used to constrain selection of possible values for the existential variables.) SubAppendix B.3.4 summarizes the results, listing the various abstraction instances which are returned by *ComAbs*.

B.3.1 Trace of NLAG's Actual Run

This subappendix presents an actual transcript of the *NLAG* system, during Run#1-1. At the end of this run is a list of points, elaborating several of the issues which arose.

Much of this trace information is included only for explanation and documentation. They are controlled by various tracing flags. If all were turned off (e.g., during

⁹This states that all pipes are straight.

```

(if (pipe-shape $p straight)
    (cd-of $p 1))

(if (and (mem $b pumps)
         (port $b 1 $j1)
         (port $b 2 $j2)
         (Pump-Press $b $v))
    (Pressure-drop $j1 $j2 $s $v))

(if (and (mem $b tanks)
         (port $b 2 $j)
         (water-height $b $s $w)
         (press-from-ht $w $p))
    (pressure $j $v))

(if (and (mem $b tanks)
         (port $b 1 $j))
    (pressure $j $s 0.0))

```

B.2.4 Facts about the Target Problem, *PT*

```

(mem s0 states)
(cross-section pipe1 1000.0)
(cross-section pipe2 1000.0)
(flow-rate j-wc-a pump1 s0 -50.0)
(port pump1 2 j-wc-a)
(port pipe1 1 j-wc-a)
(port pipe1 2 j-wc-b)
(port pipe2 1 j-wc-a)
(port pipe2 2 j-wc-b)
(port pump1 1 j-wc-b)

```

⁸Two comments: [1] This rule is but a crude approximation. [2] Notice there are no primitive facts stored which describe the pipe-character of any specific pipe. Thanks to this rule, this system can derive its values as they are needed.

a production run), the user would only see the questions asked. A nice intermediate setting allows *NLAG* to print out only those questions and the abstraction currently being considered.

To describe the information shown below: Everything typed by the user appears underlined. My subsequent comments appear in *this font*, against the right side of the page.

The -> appearing on the far left is Franz Lisp's prompt. Interspersed below is the value of the `total-tasks` variable, which tells how many tasks have been run at this point, before executing the current task. It is a list of dotted pairs. The first element of each is tag for a particular type of task and the second is a count of how many tasks of this type have been run. (Point Trace2, at the end of this trace, presents a summary of the various task types.)

This trace differs from the description in the text in a few minor ways: [1] essentially everything shown below is in lower-case; [2] every token, including abstractions and procedure names, appear in the same font; and [3] some names are slightly different. As three representative samples, `flow-rate` refers to `FlowRate`, `pipe-charact` to `PipeCharacter`, and `ohmslaw` to `Ohms`.

Finally, I did perform some cosmetic modifications to the actual output; basically to reformat it to fit nicely on the page. (I also explicitly mention the few places where some irrelevant part of the dialogue is flushed.)

This is run Run#1-1, first performed 10/X/84; rerun 2/V/85. To distinguish it: It uses Generator#1, and so ranks highest the relations subclass* and mem. It does include the H_{MGA} heuristic. The initial theory has only the four abstractions shown on page 140; in particular, it does not include rlkk, rckk, lckk or rlckk.

-> (nlag)

The (nlag) routine performs some overhead, and then calls Truep on the expression

`(nlag <AH> (query <PT>) $fs $ft $th),`

where <AH> is the analogical hint and <PT> is the target problem. It returns bindings for the final three variables: \$fs corresponding to the source fact (e.g., (rkk current voltage-drop resistance resistors)), \$ft, to the target fact (e.g., (rkk flow-rate pressure-drop pipe-character pipes)) and \$th, to the resulting theory (e.g., "t8"). This theory holds the conjectures needed to satisfy this target fact, i.e., the leaf residue of the target fact.

Notice that MRS designates variables by names which begin with "\$".

About to run the task

```
(find-kernel ((current . flow-rate))
             (query (flow-rate j-wc-a pipe1 s0 $fr)
                   $677 $678))
```

NLAG just entered the Find-Kernel module, on this particular set of arguments. The analogical hint, "((current . flow-rate))", and target problem "(query (flow-rate j-wc-a pipe1 s0 \$fr))" appear prominently.

Timing: 4

```
((bcdisp . 4))
```

The figure on the top line (beside "Timing:") tells the number of tasks run so far. We see here that the overhead has taken 4 steps. (Point Trace1 explains why this is used as a measurement of time.) The next line is a break-down of the tasks actually executed. So far, all of the tasks have been "bcdisp". Point Trace2 explains all of the different tasks.

About to run the task

```
(inst-source (p1107 p1205 p1108 p1206 p1105 p1203)
            electric ($674 . $676))
```

Find-Kernel has finished, and passed its result, the list (p1107 p1205 p1108 p1206 p1105 p1203), to the next process, Inst-Source. This list represents the kernel of relevant source facts; SubAppendix B.3.2 explicates these propositions.

Timing: 37

```
((bcdisp . 33) (succeed . 4))
```

* * * Abstraction: rkk * * *

About to run the task

```
(inst-target ((current . flow-rate))
  (query (flow-rate j-wc-a pipe1 s0 $fr))
  rkk (current voltage-drop resistance elect-devices)
  $675 $th)
```

NLAG has proposed the rkk abstraction. Here, at the start of the 51th step (see below), it starts the Inst-Target task.

Timing: 51

```
((bcdisp . 41) (succeed . 5) (ff-disp . 3) (fc-only . 2))
```

[Conj-Bc1] is conjecturing that (kirchoff1 flow-rate) is true!

... into theory t5

Conj-Bc1 is a submodule of Inst-Target. Here, it is considering the conjecture (kirchoff1 flow-rate). It stores these proposals in temporary theories; this first one is stored in the theory "t5". The contents of this temporary theory (as well as the others) appear in Subappendix B.3.3.

[Conj-Bc1] is conjecturing that (kirchoff2 ?2) is true!

... into theory t6

As discussed in Subsection 6.2.3, CTruep uses dummy existential variables, such as this "?2", during its first sub-step, CTruep-1.

[Conj-Bc1] is conjecturing that (conserved-thru flow-rate ?4) is true!

... into theory t7

[Conj-Bc1] is conjecturing that (ohmslaw flow-rate ?2 ?3 ?4) is true!

... into theory t8

Now for the real work: instantiating these three existential variables. Each term represents an argument of the rkk relation. Associated with each is a collection of propositions implied by the various conjectures. These propositions are stored in the theory t8 (which includes the theories t5 through t7). See SubAppendix B.3.3.

About to run the task

```
(resolve-exists t8
  (($TH$ . t8) ($704 . ?2) ($706 . ?4) ($705 . ?3) (t .
  (query (flow-rate j-wc-a pipe1 s0 $fr))
  $th $703))
```

Timing: 76

```
((bcdisp . 61) (succeed . 6) (ff-disp . 3) (fc-only . 2)
  (conj-bc . 4))
```

The conj-bc tasks are called by the CTruep process. Each indicates one conjecture which has been posited.

Initial values for ?3: (pump-press water-height pipe-shape cd-of pipe-charact
cross-section flow-rate pressure-drop pressure drop-
of port units-of / * - +)

This ?3 existential variable serves as a place holder for rkk's third argument. Using the proposition (mem ?3 functions) as a generator (it is shown on the next line), we see there are 16 possible values. This includes various "irrelevant" functions (e.g., arithmetic ones) which are soon eliminated. (Run#1-2 is based on another possible generator. While this generator avoids these particular extraneous candidates, it does consider others. See Subsection 7.2.2.)

Using (mem ?3 functions)

Legal values for ?3: ((cross-section . 2) (pipe-charact . 2) (cd-of . 2)
(pump-press . 2))

After a quick pruning, only these 4 possible values remain for ?3. The number following each entry tells the number of sought facts which were primitively stored for this possible value. This is later used to help order these entries. (See page 127 in Subsection 6.2.3 and Note:9-1.)

Initial values for ?4: (devices water-devices pumps tanks tubes pipes thin-
pipes orifices)

Using (subclass* ?4 devices)

Legal values for ?4: ((orifices . 1) (thin-pipes . 1) (pipes . 1) (tubes
. 1) (tanks . 1) (pumps . 1) (water-devices . 1)
(devices . 1))

We see above that all 8 values for ?4 (rkk's fourth argument) remain.

Initial values for ?2: (pump-press water-height pipe-shape cd-of pipe-charact
cross-section flow-rate pressure-drop pressure drop-
of port units-of / * - +)

Using (mem ?2 functions)

Legal values for ?2: ((pressure-drop . 6))

We are almost to the end of ComAbs processing. It has found both the abstraction rkk and this set of possible instantiations. The array below shows the number of entries generated for each existential variable followed by the number which pass the first consistency check, noted by 1-Consist below. (This is described on page 125 of the text. This data also appears in Table 7-2.)

| Var | Generated | 1-Consist |
|-------|-----------|-----------|
| ?3 | 16 | 4 |
| ?4 | 8 | 8 |
| ?2 | 16 | 1 |
| Total | 2048 | 32 |

This leaves only a scant 32 possible binding lists for this rkk abstraction, each formed by appending FlowRate to the front of an entry from the cross-product of these sets.¹⁰

¹⁰The fact that each entry in this table is a power of 2 is a curious coincidence. In earlier runs, with slightly different data, this was not the case.

Next, Inst-Target tests each of these 32 quadruples. The ones which pass this test are handed to Verify for another series of pruning tests. (These tests are described in Subsection 6.1.1.)

##1: (+)

About to run the task

```
(verify (query (flow-rate j-wc-a pipe1 s0 $fr))
 (t8 (?3 . cross-section) (?4 . pipes) (?2 . pressure-drop))
 (rkk flow-rate pressure-drop cross-section pipes))
```

This (rkk flow-rate pressure-drop cross-section pipes) target abstraction instance is the first entry which ComAbs returns. We now see if Verify approves.

The (t8 (?3 . cross-section) (?4 . pipes) (?2 . pressure-drop)) notation refers to the theory based on the temporary theory t8, but in which the existential variable ?3 is "aliased" to the constant cross-section, etc. The "##n" before each Verify invocation is included to facilitate subsequent references to this particular instantiation. The token (+) has been appended afterward to indicate that this instantiation led to an answer to the target problem.

Timing: 352

```
((bcdisp . 255) (succeed . 47) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))
```

NLAG does have a crude algebraic simplified and problem solving facility. Point Trace3 mentions that this problem solver is not that smart; this particular answer was supplied by the user.

In Solve (1)

```
ExVarsList = ((?3 . cross-section) (?4 . pipes) (?2 . pressure-drop))
```

Timing: 364

```
((bcdisp . 267) (succeed . 47) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))
```

Found soln: (flow-rate j-wc-a pipe1 s0 25.0)

So this first \vdash analogical inference, (rkk flow-rate pressure-drop cross-section pipes), also passes \vdash_{PT} 's test: i.e., it does produce a solution to the (flow-rate j-wc-a pipe1 s0 \$fr) query. As Verify has already confirmed that this new conjecture is consistent, it needs only get the user's acceptance to be added to the theory. Below it asks the user if he approves of each member of the leaf residue:

Is (kirchoff1 flow-rate) reasonable? y

Verify is asking for my approval of this individual conjecture. Typing y means that I do.

Is (kirchoff2 pressure-drop) reasonable? y

Is (conserved-thru flow-rate pipes) reasonable? y

Is (ohmslaw flow-rate pressure-drop cross-section pipes) reasonable? n

Had I said y, NLAG would have returned this answer. I said no because this particular proposition is not true, of course. This causes Verify to fail on this instantiation. By back-tracking, NLAG then asks ComAbs for the next instantiation.

Timing: 374

((bcdisp . 276) (succeed . 48) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##2: (+)

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . cd-of) (?4 . pipes) (?2 . pressure-drop))
(rkk flow-rate pressure-drop cd-of pipes))

Timing: 396

((bcdisp . 298) (succeed . 48) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

In Solve (2)

ExVarsList = ((?3 . cd-of) (?4 . pipes) (?2 . pressure-drop))

Timing: 431

((bcdisp . 331) (succeed . 50) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Found soln: (flow-rate j-wc-a pipe1 s0 25.0)

Conjecture (kirchoff2 pressure-drop) already OKed.

Notice Verify is smart enough to know that this proposition has already been approved, and so does not even ask. The "(kirchoff1 flow-rate)" ex-conjecture is not even mentioned here, for reasons given in Point Trace5.

Conjecture (conserved-thru flow-rate pipes) already OKed.

Is (ohmslaw flow-rate pressure-drop cd-of pipes) reasonable? n

Once again, I say n as this proposition is not correct.

Timing: 440

((bcdisp . 339) (succeed . 51) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##3: (+)

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . cross-section) (?4 . tubes) (?2 . pressure-drop))
(rkk flow-rate pressure-drop cross-section tubes))

Timing: 452

((bcdisp . 351) (succeed . 51) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

In Solve (3)

ExVarsList = ((?3 . cross-section) (?4 . tubes) (?2 . pressure-drop))

Timing: 464

((bcdisp . 363) (succeed . 51) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Found soln: (flow-rate j-wc-a pipe1 s0 25.0)
 Conjecture (kirchoff2 pressure-drop) already OKed.
 Is (conserved-thru flow-rate tubes) reasonable? y

The above fact is true, if irrelevant.

Is (ohmslaw flow-rate pressure-drop cross-section tubes) reasonable? n
 Timing: 473

((bcdisp . 371) (succeed . 52) (ff-disp . 3) (fc-only .
 2) (conj-bc . 4) (fcdisp . 41))

##4: (+)

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
 (t8 (?3 . pipe-charact) (?4 . pipes) (?2 . pressure-drop))
 (rkk flow-rate pressure-drop pipe-charact pipes))

Timing: 481

((bcdisp . 379) (succeed . 52) (ff-disp . 3) (fc-only .
 2) (conj-bc . 4) (fcdisp . 41))

In Solve (4)

ExVarsList = ((?3 . pipe-charact) (?4 . pipes) (?2 . pressure-drop))

Timing: 526

((bcdisp . 422) (succeed . 54) (ff-disp . 3) (fc-only .
 2) (conj-bc . 4) (fcdisp . 41))

Found soln: (flow-rate j-wc-a pipe1 s0 25.0)

Conjecture (kirchoff2 pressure-drop) already OKed.

Conjecture (conserved-thru flow-rate pipes) already OKed.

Is (ohmslaw flow-rate pressure-drop pipe-charact pipes) reasonable? n

Timing: 535

((bcdisp . 430) (succeed . 55) (ff-disp . 3) (fc-only .
 2) (conj-bc . 4) (fcdisp . 41))

NLAG has found the correct answer on this 4th iteration. At this point, only 8 questions have been asked, and only 535 tasks have been run. Had I typed y, NLAG would take 4 more steps before returning this correct answer, therefore requiring a total of 539 steps.¹¹ In the interest of finding all possible solutions, I said n instead.

##5:

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
 (t8 (?3 . cross-section) (?4 . thin-pipes) (?2 . pressure-drop))
 (rkk flow-rate pressure-drop cross-section thin-pipes))

Timing: 543

((bcdisp . 438) (succeed . 55) (ff-disp . 3) (fc-only .
 2) (conj-bc . 4) (fcdisp . 41))

¹¹Of these, 318 steps (60.0%) were spent in the *Resolve-Exists* module. See Table B-1.

Using bindings ((?3 . cross-section) (?4 . thin-pipes) (?2 . pressure-drop))

Found nil answer to (flow-rate j-wc-a pipe1 s0 \$fr).

All four of the previous proposed \vdash analogies qualify as legal \vdash_{PT} analogies, since each leads to an answer to the initial query. Here, this fifth binding set did not: i.e., it qualifies only as a \vdash analogy, not as a \vdash_{PT} one.

Timing: 560

((bcdisp . 455) (succeed . 55) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##6:

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . cross-section) (?4 . orifices) (?2 . pressure-drop))
(rkk flow-rate pressure-drop cross-section orifices))

Timing: 568

((bcdisp . 463) (succeed . 55) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Using bindings ((?3 . cross-section) (?4 . orifices) (?2 . pressure-drop)):

Found nil answer to (flow-rate j-wc-a pipe1 s0 \$fr).

Timing: 585

((bcdisp . 480) (succeed . 55) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##7:

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . pump-press) (?4 . pumps) (?2 . pressure-drop))
(rkk flow-rate pressure-drop pump-press pumps))

Timing: 593

((bcdisp . 488) (succeed . 55) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Using bindings ((?3 . pump-press) (?4 . pumps) (?2 . pressure-drop)):

Found nil answer to (flow-rate j-wc-a pipe1 s0 \$fr).

Timing: 624

((bcdisp . 519) (succeed . 55) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##8: (+)

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . cd-of) (?4 . tubes) (?2 . pressure-drop))
(rkk flow-rate pressure-drop cd-of tubes))

Timing: 644

((bcdisp . 539) (succeed . 55) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

In Solve (8)

ExVarsList = ((?3 . cd-of) (?4 . tubes) (?2 . pressure-drop))

Timing: 679

((bcdisp . 572) (succeed . 57) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Found soln: (flow-rate j-wc-a pipe1 s0 25.0)

Conjecture (kirchoff2 pressure-drop) already OKed.

Conjecture (conserved-thru flow-rate tubes) already OKed.

Is (ohmslaw flow-rate pressure-drop cd-of tubes) reasonable? n

This is the last question asked.

Timing: 688

((bcdisp . 580) (succeed . 58) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##9:

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . cd-of) (?4 . thin-pipes) (?2 . pressure-drop))
(rkk flow-rate pressure-drop cd-of thin-pipes))

Timing: 696

((bcdisp . 588) (succeed . 58) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Using bindings ((?3 . cd-of) (?4 . thin-pipes) (?2 . pressure-drop)):

Found nil answer to (flow-rate j-wc-a pipe1 s0 \$fr).

Timing: 713

((bcdisp . 605) (succeed . 58) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

##10:

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t8 (?3 . cd-of) (?4 . orifices) (?2 . pressure-drop))
(rkk flow-rate pressure-drop cd-of orifices))

Timing: 721

((bcdisp . 613) (succeed . 58) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

Using bindings ((?3 . cd-of) (?4 . orifices) (?2 . pressure-drop)):

Found nil answer to (flow-rate j-wc-a pipe1 s0 \$fr).

Timing: 738

((bcdisp . 630) (succeed . 58) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))

The fc-find-next subroutine, seen below, marks the end of NLAG's focus on the rkk abstraction. (I.e., all of the possible rkk abstraction instances have been considered.) Fc-find-next now seeks another abstraction which can be instantiated in the source domain.

To summarize the results so far: ComAbs produced a total of 10 legal \vdash analogies, of which 5 are useful (i.e., only 5 satisfied \vdash_{PT} 's requirements). These are numbered 1, 2, 3, 4 and 8. As shown below, NLAG has used 761 deductive steps so far. The first 37 were spent on overhead and in the Find-Kernel subroutine. The remaining 724 were devoted to this rkk abstraction. Notice that about 401 of these 724 steps (about 55%) involved instantiating the variables. (Point Trace4 derives this number.)

About to run the task

```
(fc-find-next (p1107 p1205 p1108 p1206 p1105 p1203)
              electric ($2172 . $2174) $2195 !16)
```

All of the arguments on the second line are overhead.

Timing: 761

```
((bcdisp . 653) (succeed . 58) (ff-disp . 3) (fc-only .
2) (conj-bc . 4) (fcdisp . 41))
```

* * * Abstraction: kk * * *

Inst-Source next proposes the abstraction, kk. Point Trace6 discusses why kk was chosen in this position. Note:6-2 discusses why Inst-Source bothers to consider kk at all, after rkk had failed.

About to run the task

```
(inst-target ((current . flow-rate))
             (query (flow-rate j-wc-a pipe1 s0 $fr))
             kk (current voltage-drop)
             $675 $th)
```

Timing: 779

```
((bcdisp . 657) (succeed . 59) (ff-disp . 10) (fc-only
. 8) (conj-bc . 4) (fcdisp . 41))
```

[Conj-Bc1] is conjecturing that (kirchoff2 ?2) is true!

About to run the task

```
(resolve-exists t9
              (($TH$ . t9) ($1346 . ?2) (t . t))
              (query (flow-rate j-wc-a pipe1 s0 $fr))
              $th $1345)
```

Timing: 792

```
((bcdisp . 668) (succeed . 60) (ff-disp . 10) (fc-only
. 8) (conj-bc . 4) (fcdisp . 41) (check-out . 1))
```

Initial values for ?2: (pump-press water-height pipe-shape cd-of pipe-character cross-section flow-rate pressure-drop pressure drop-of port units-of / * - +)

Using (mem ?2 functions)

Legal values for ?2: ((pressure-drop . 8))

There is only a single existential variable to consider, and only a single possible value it might assume. (Notice this kk target instantiation was a subgoal of the earlier rkk abstraction target instantiation task. The Super-Cache facility allowed us to store these intermediate results, making this search trivial. See Point MRS1 in Subsection 6.2.4.)

```

to run the task
verify (query (flow-rate j-wc-a pipe1 s0 $fr))
      (t9 (?2 . pressure-drop))
      (kk flow-rate pressure-drop))

```

This t9 theory is identical to the earlier t6, as expected.

```

: 898
((bcdisp . 739) (succeed . 76) (ff-disp . 10) (fc-only
 . 8) (conj-bc . 4) (fcdisp . 60) (check-out . 1))
bindings ((?2 . pressure-drop)):
nil answer to (flow-rate j-wc-a pipe1 s0 $fr).

```

```

: 973
((bcdisp . 814) (succeed . 76) (ff-disp . 10) (fc-only
 . 8) (conj-bc . 4) (fcdisp . 60) (check-out . 1))

```

Onto the third abstraction.

```

to run the task
c-find-next (p1107 p1205 p1108 p1206 p1105 p1203)
           electric ($2172 . $2174) $2195 !16)

```

```

: 985
((bcdisp . 826) (succeed . 76) (ff-disp . 10) (fc-only
 . 8) (conj-bc . 4) (fcdisp . 60) (check-out . 1))

```

Abstraction: ckk * * *

```

to run the task
inst-target ((current . flow-rate))
            (query (flow-rate j-wc-a pipe1 s0 $fr))
            ckk (current voltage-drop capacitance capacitors)
            $675 $th)

```

```

: 995
((bcdisp . 830) (succeed . 77) (ff-disp . 13) (fc-only
 . 10) (conj-bc . 4) (fcdisp . 60) (check-out . 1))

```

[Bc1] is conjecturing that (kirchoff2 ?2) is true!

[Bc1] is conjecturing that (capac-law flow-rate ?2 ?10 ?11) is true!

into theory t11

```

to run the task
resolve-exists t11
              (($TH$ . t11) ($1824 . ?2) ($1826 . ?11) ($1825 . ?10) (t . t)
              (query (flow-rate j-wc-a pipe1 s0 $fr))
              $th $1823)

```

```

: 1013
((bcdisp . 845) (succeed . 78) (ff-disp . 13) (fc-only
 . 10) (conj-bc . 5) (fcdisp . 60) (check-out . 2))

```


Initial values for ?10: (pump-press water-height pipe-shape cd-of pipe-charact
cross-section flow-rate pressure-drop pressure drop-
of port units-of / * - +)

Using (mem ?10 functions)

Legal values for ?10: ((pressure . 3) (water-height . 3))

Initial values for ?11: (devices water-devices pumps tanks tubes pipes thin-
pipes orifices)

Using (subclass* ?11 devices)

Legal values for ?11: ((orifices . 1) (thin-pipes . 1) (pipes . 1) (tubes
. 1) (tanks . 1) (pumps . 1) (water-devices . 1)
(devices . 1))

Initial values for ?2: (pump-press water-height pipe-shape cd-of pipe-charact
cross-section flow-rate pressure-drop pressure drop-
of port units-of / * - +)

Using (mem ?2 functions)

Legal values for ?2: ((pressure-drop . 8))

##12:

About to run the task

(verify (query (flow-rate j-wc-a pipe1 s0 \$fr))
(t11 (?10 . water-height) (?11 . tanks) (?2 . pressure-drop))
(ckk flow-rate pressure-drop water-height tanks))

Timing: 1301

((bcdisp . 1053) (succeed . 119) (ff-disp . 13) (fc-only
. 10) (conj-bc . 5) (fcdisp . 99) (check-out . 2))

Using bindings ((?10 . water-height) (?11 . tanks) (?2 . pressure-drop)):

Found nil answer to (flow-rate j-wc-a pipe1 s0 \$fr).

Timing: 1318

((bcdisp . 1070) (succeed . 119) (ff-disp . 13) (fc-only
. 10) (conj-bc . 5) (fcdisp . 99) (check-out . 2))

Onto the fourth and final abstraction.

About to run the task

(fc-find-next (p1107 p1205 p1108 p1206 p1105 p1203)
electric (\$2172 . \$2174) \$2195 !16)

Timing: 1345

((bcdisp . 1097) (succeed . 119) (ff-disp . 13) (fc-only
. 10) (conj-bc . 5) (fcdisp . 99) (check-out . 2))

* * * Abstraction: lkk * * *

t to run the task

```
(inst-target ((current . flow-rate))
  (query (flow-rate j-wc-a pipe1 s0 $fr))
  lkk (current voltage-drop inductance inductors)
  $675 $th)
```

ng: 1352

```
((bcdisp . 1101) (succeed . 120) (ff-disp . 14) (fc-only
. 11) (conj-bc . 5) (fcdisp . 99) (check-out . 2))
j-Bc1] is conjecturing that (kirchoff2 ?2) is true!
j-Bc1] is conjecturing that (induct-law flow-rate ?2 ?13 ?14) is true!
into theory t13
```

t to run the task

```
(resolve-exists t13
  (($TH$ . t13) ($1957 . ?2) ($1959 . ?14) ($1958 . ?13) (t . t)
  (query (flow-rate j-wc-a pipe1 s0 $fr))
  $th $1956)
```

ng: 1370

```
((bcdisp . 1116) (succeed . 121) (ff-disp . 14) (fc-only
. 11) (conj-bc . 6) (fcdisp . 99) (check-out . 3))
ial values for ?13: (pump-press water-height pipe-shape cd-of pipe-charact
cross-section flow-rate pressure-drop pressure drop-
of port units-of / * - +)
```

g (mem ?13 functions)

```
1 values for ?13: ((pressure . 3) (water-height . 3))
```

```
ial values for ?14: (devices water-devices pumps tanks tubes pipes thin-
pipes orifices)
```

g (subclass* ?14 devices)

```
1 values for ?14: ((orifices . 1) (thin-pipes . 1) (pipes . 1) (tubes
) (tanks . 1) (pumps . 1) (water-devices . 1) (devices . 1))
```

```
ial values for ?2: (pump-press water-height pipe-shape cd-of pipe-charact
cross-section flow-rate pressure-drop pressure drop-
of port units-of / * - +)
```

g (mem ?2 functions)

```
1 values for ?2: ((pressure-drop . 8))
```

t to run the task

```
(verify (query (flow-rate j-wc-a pipe1 s0 $fr))
  (t13 (?13 . water-height) (?14 . tanks) (?2 . pressure-drop))
  (lkk flow-rate pressure-drop water-height tanks))
```

Timing: 1658

((bcdisp . 1324) (succeed . 162) (ff-disp . 14) (fc-only
 . 11) (conj-bc . 6) (fcdisp . 138) (check-out . 3))

Using bindings ((?13 . water-height) (?14 . tanks) (?2 . pressure-drop)):

Found nil answer to (flow-rate j-wc-a pipel s0 \$fr).

Timing: 1675

((bcdisp . 1341) (succeed . 162) (ff-disp . 14) (fc-only
 . 11) (conj-bc . 6) (fcdisp . 138) (check-out . 3))

Are there any other abstractions to consider?

About to run the task

(fc-find-next (p1107 p1205 p1108 p1206 p1105 p1203)
 electric (\$2172 . \$2174) \$2195 !16)

Timing: 1703

((bcdisp . 1369) (succeed . 162) (ff-disp . 14) (fc-only
 . 11) (conj-bc . 6) (fcdisp . 138) (check-out . 3))

nil

At this point, NLAG gives up, having tried everything it can. Notice it did in fact find the correct answer in its search, which I vetoed.

-> (show-info)

Timing: 1708

((bcdisp . 1373) (succeed . 162) (ff-disp . 14) (fc-only
 . 12) (conj-bc . 6) (fcdisp . 138) (check-out . 3))

All told, this search required 1708 tasks. A total of 13 \sim analogies were considered, only 5 of which produced an answer. These are listed in Sub-Appendix B.3.4. A total of 6 distinct conjectures were considered for the 4 different abstractions.

Trace1. Why not use CPU Time?

There are several reasons why this run does not show CPU times. The most important reason is that it is really not important. This system is but a pilot implementation, one whose *raison d'être* is simply to demonstrate the nature of analogical inference; a functional specification is all I wanted. Since it is not meant as a production grade system, I did not mind writing it in Franz LISP, and running it on a relatively slow VAX 11/780 running UNIX^(tm). The second reason is that one can readily estimate the timing cost, using the estimate of 500 Logical Inferences Per Second. (This pertains to MRS running in MacLisp on a DEC20 running TOPS20 [Gen85]). This means *NLAG* would reach the desired answer in a bit over 1 CPU second (about 539/500 second), and full run would require about $1708/500 \approx 3.5$ CPU seconds.

Trace2. Types of Tasks:

The *NLAG* process used various types of tasks, whose names appear in the total-tasks list seen throughout this trace. This point describes these tasks.

bcdisp is the standard task used by the backward-chainer, *BC*.

fcdisp is the standard task used by the straightforward forward-chainer, *FC*.

succeed is used to indicate the successful completion of some earlier task and to pass along its answer. (These top three are all described in [Rus85].)

ff-disp, *fc-only* are used by the *Inst-Source* forward-chaining module. (*Inst-Source* also uses additional *bcdisp* tasks.)

conj-bc tasks are employed by *CTruep*. These are used to postulate that some proposition holds. (*CTruep* also uses additional *bcdisp* tasks.)

check-out tasks are employed by *CTruep* when trying to re-use data *SuperCached* earlier.

Trace3. Constraint Solver:

The problem solver associated with constraint information does a fair amount of algebraic simplification in its attempt to solve a system of linear equations.

However, there are many places when it knows there is a solution, but is not clever enough to know what that solution is. (E.g., , In the situation shown in the trace above, it has 5 linear equations and 5 unknowns.) In these situations, it simply asks the user for assistance. In this presentation, this asking step has been flushed and the correct answer, simply volunteered.

Trace4. Number of *Resolve-Exists* Steps:

It may not be obvious how to compute the number of *Resolve-Exists* steps. There are 276 deduction steps from the start of *Resolve-Exists* to the start of the first *Verify* call; all are in the service of *Resolve-Exists*. When this *Verify* invocation fails, *Resolve-Exists* was asked to determine the next candidate. To a first approximation, *Resolve-Exists* began immediately after the final question was asked (at step 377), and returned its answer at the start of the ##2 *Verify* call, at step 399. Hence, *Resolve-Exists* required an additional 22 steps to find this second candidate.

Similarly, we can determine that *Resolve-Exists* took 12 steps to find the third abstraction instance, and then 8 steps to find each of the fourth, fifth and ninth instances. To find the sixth abstraction instance, we assume the *Resolve-Exists* took over just when the *Solve* procedure "Found nil answer to ..." for the ##5th proposed abstraction instance, i.e., it began on step 563. Hence, *Resolve-Exists* required 8 steps here. Likewise, it took 8 steps to find seventh and tenth abstraction instance, and 20 to find the eighth. It also used the 21 steps after the failure of ##10 (on step 741) and the start of the *fc-find-next* on step 764.

We can use this analysis to find the number of steps required by *Verify*. Table B-1 summarizes these numbers. The totals are used in Table 7-3.

Table B-2 provides corresponding figures for the other three abstractions. For the *kk* abstraction: *Resolve-Exists* uses the 106 steps to find candidate ##11 (from step 795 through step 901) plus the 12 steps (from 944 through 956) to conclude there are no other instances. Similar numbers can be seen for the *ckk* and *lkk* abstractions.

Trace5. Why is (kirchoff1 flow-rate) not Mentioned?

| Abst | ##? | Resolve-Exists | Verify | $\ _{PT}?$ |
|-------|-----|------------------|-----------------|------------|
| rkk | 1 | 276 ¹ | 22 ² | + |
| | 2 | 22 ³ | 44 | + |
| | 3 | 12 | 21 | + |
| | 4 | 8 ⁴ | 54 | + |
| | 5 | 8 | 17 ⁵ | |
| | 6 | 8 | 17 | |
| | 7 | 8 | 31 | |
| | 8 | 20 | 44 | + |
| | 9 | 8 | 17 | |
| | 10 | 8 | 17 | |
| | ?11 | 23 ⁶ | | |
| Total | | 401 | 284 | |

Notes

- 1 from *Resolve-Exists* step 79 through the start of the first *Verify*, at step 355.
- 2 from start of the first *Verify* (step 355) through the last question, at step 377.
- 3 from that last question (step 377) through the start of the second *Verify*, at step 399.
- 4 if we add up these top four instances of *Resolve-Exists*, we derive the 318 figure used earlier.
- 5 from last question of ##4 (step 538) through *Solve*'s NIL answer, at step 563.
- 6 This reflects *Inst-Targets* unsuccessful search for another target instantiation of the rkk abstraction. (This is also why this row is tagged "?11".) We attribute all of it to *Resolve-Exists* (even though a few steps are probably spent on *CTruep-I*'s behalf).

Table B-1: Deductions for *Resolve-Exists* and *Verify*, for the rkk Abstraction

| Abst | ##? | Resolve-Exists | Verify | \vdash_{PT} ? |
|------|-----|----------------|--------|-----------------|
| kk | 11 | 106 | 75 | |
| | ?12 | 12 | | |
| | | 118 | 75 | |
| cck | 12 | 288 | 17 | |
| | ?13 | 27 | | |
| | | 315 | 17 | |
| lkk | 13 | 288 | 17 | |
| | ?14 | 28 | | |
| | | 316 | 17 | |

Table B-2: Deductions for *Resolve-Exists* and *Verify*, for the other Abstractions

This is because (kirchoff1 flow-rate) has already been added to the theory, as a result of the y response given to the question asked at around deduction step number 377. Of course, this same can be said of the (kirchoff2 pressure-drop) fact: it has been accepted as well. Why then is this second case (dealing with kirchoff2) mentioned here, but the first case (dealing with kirchoff1) is not? The problem arises because this second conjecture first appears as (kirchoff2 ?2); that is, it is burdened with an existential variable as an argument. This makes it more difficult to determine the derivability of this fact *a priori*.

Trace6. Why Pick kk Next?:

As the above trace shows, the abstraction kk is selected before the more specific cck and lkk abstractions, even though all three were in the running. As the H_{JK} “set of support” measure ranks all of these candidates equally, the H_{MGA} is allowed to decide. It chooses the most general abstraction, kk, first. (Of course, this is after rkk had been tried and (apparently) failed. Subsection 5.4.2 explains why kk is even considered in this situation.)

B.3.2 Source Kernel

```

p1107: (if (conserved-thru $tt $load)
          (constraint (+ $c1 $c2 0.0)
                     (and (mem $x $load)
                          (port $x $i $j1)
                          (port $x $k $j2)
                          (< $i $k))
                          ($tt $j1 $x $s $c1)
                          ($tt $j2 $x $s $c2)))

p1205: (conserved-thru current elect-devices)

p1108: (if (ohmslaw $tt $ct $rt $load)
          (constraint (* $c $r $vd)
                     (and (port $x 1 $j1)
                          (mem $x $load)
                          (port $x 2 $j2))
                          ($rt $x $r)
                          ($ct $j1 $j2 $s $vd)
                          ($tt $j1 $x $s $c)))

p1206: (ohmslaw current voltage-drop resistance elect-devices)

p1105: (if (kirchoff1 $tt)
          (constraint (num-= (+ . $vs) 0.0)
                     (and (bagof $d (port $d $i $j) $ds)
                          (form-list $d $ds ($tt $j $d $s +NewVar+) $curs)
                          (finals $curs $vs))
                          ($tt $j $x $s $e) . $curs)))

p1203: (kirchoff1 current)

```

Notice that (kirchoff2 voltagedrop) is not included here. This is because NLAG does not actually do the entire computation of finding the current; instead

it uses various heuristics to guess the needed facts. Since this statement is the only omitted justification, we see this quick heuristic worked quite well.

B.3.3 Conjectures Considered

This subappendix presents the set of conjectures which are considered, organized in terms of the temporary theories involved. That is, each theory houses a particular conjecture and various entailments. The list below shows the six different theories mentioned in the above run. In each, one particular conjecture spawned all of the entries; it is listed first. Each theory also specifies which other theories it includes.

At the end of this subappendix, we provide a brief description of some of the more obscure propositions.

Theory t5:

This is the topmost temporary theory. It directly includes the permanent wcl theory, which, recall, which describes the target problem.

p1371: (kirchoff1 flow-rate)

p1520: (if (useconstraints (flow-rate \$723 \$722 \$725 \$719)) (flow-rate \$723 \$722 \$725 \$719))

p1558: (constraint (num= (+ . \$735) 0.0) (and (bagof \$736 (port \$736 \$737 \$738) \$739) (form-list \$736 \$739 (flow-rate \$738 \$736 \$740 +NewVar+) \$741) (finals \$741 \$735)) (flow-rate \$738 \$742 \$740 \$743) . \$741)

Theory t6:

This temporary theory includes t5. (Since inclusion is transitive, it therefore includes wc1, and all of wc1's parents as well.)

p1472: (kirchoff2 ?2)
p1489: (defined-below ?2 lels)
p1541: (arity ?2 4)
p1543: (domain ?2 1 junctions)
p1545: (domain ?2 2 junctions)
p1547: (domain ?2 3 states)
p1549: (domain ?2 4 numbers)
p1536: (domain-spec ?2 junctions junctions states numbers)
p1532: (mem ?2 functions)
p1474: (kk flow-rate ?2)

Theory t7:

This temporary theory includes t6.

p1479: (conserved-thru flow-rate ?4)
p1520: (if (useconstraints (flow-rate \$723 \$722 \$725 \$719)) (flow-rate \$723 \$722 \$725 \$719))
p1522: (constraint (+ \$726 \$727 0.0) (and (mem \$728 ?4) (port \$728 \$729 \$730) (port \$728 \$731 \$732) (< \$729 \$731)) (flow-rate \$730 \$728 \$733 \$726) (flow-rate \$732 \$728 \$733 \$727))

Theory t8:

This temporary theory includes t7.

p1483: (ohmslaw flow-rate ?2 ?3 ?4)

p1520: (if (useconstraints (flow-rate \$723 \$722 \$725 \$719)) (flow-rate \$723 \$722 \$725 \$719))

p1518: (if (useconstraints (?2 \$723 \$724 \$725 \$721)) (?2 \$723 \$724 \$725 \$721))

p1516: (if (useconstraints (?3 \$722 \$720)) (?3 \$722 \$720))

p1514: (constraint (* \$719 \$720 \$721) (and (port \$722 1 \$723) (mem \$722 ?4) (port \$722 2 \$724)) (?3 \$722 \$720) (?2 \$723 \$724 \$725 \$721) (flow-rate \$723 \$722 \$725 \$719))

p1489: (defined-below ?2 lels)

p1491: (defined-below ?3 lels)

p1493: (defined-below ?4 lels)

p1511: (subclass* ?4 devices)

p1512: (mem ?4 classes)

p1504: (arity ?3 2)

p1506: (domain ?3 1 ?4)

p1508: (domain ?3 2 numbers)

p1501: (domain-spec ?3 ?4 numbers)

p1497: (mem ?3 functions)

p1485: (rkk flow-rate ?2 ?3 ?4)

Theory t11:

This temporary theory includes t10. The theory t10 is not shown, since it is identical to t6.

- p1520: (if (useconstraints (flow-rate \$723 \$722 \$725 \$719)) (flow-rate \$723 \$722 \$725 \$719))
- p1518: (if (useconstraints (?2 \$723 \$724 \$725 \$721)) (?2 \$723 \$724 \$725 \$721))
- p1688: (if (useconstraints (?10 \$1840 \$1843)) (?10 \$1840 \$1843))
- p1686: (constraint (* \$1837 \$1838 \$1839) (and (port \$1840 1 \$1841) (mem \$1840 ?11) (port \$1840 2 \$1842)) (?10 \$1840 \$1843) (minus \$1843 \$1837) (?2 \$1841 \$1842 \$1844 \$1845) (flow-rate \$1841 \$1840 \$1844 \$1839) (d-dt ?2 \$1841 \$1842 \$1844 \$1846))
- p1652: (capac-law flow-rate ?2 ?10 ?11)
- p1489: (defined-below ?2 lels)
- p1659: (defined-below ?10 lels)
- p1661: (defined-below ?11 lels)
- p1684: (mem ?11 classes)
- p1683: (subclass* ?11 devices)
- p1674: (arity ?10 3)
- p1676: (domain ?10 1 ?11)
- p1678: (domain ?10 2 states)
- p1680: (domain ?10 3 numbers)
- p1669: (domain-spec ?10 ?11 states numbers)
- p1665: (mem ?10 functions)
- p1654: (ckk flow-rate ?2 ?10 ?11)

Theory t13:

This temporary theory includes t12. This t12 is identical to t6, shown above.

- p1520: (if (useconstraints (flow-rate \$723 \$722 \$725 \$719)) (flow-rate \$723 \$722 \$725 \$719))
- p1518: (if (useconstraints (?2 \$723 \$724 \$725 \$721)) (?2 \$723 \$724 \$725 \$721))
- p1741: (if (useconstraints (?13 \$1973 \$1970)) (?13 \$1973 \$1970))
- p1739: (constraint (* \$1970 \$1971 \$1972) (and (port \$1973 1 \$1974) (mem \$1973 ?14) (port \$1973 2 \$1975)) (?13 \$1973 \$1970) (?2 \$1974 \$1975 \$1976 \$1971) (flow-rate \$1974 \$1973 \$1976 \$1977) (d-dt flow-rate \$1974 \$1973 \$1976 \$1972))
- p1705: (induct-law flow-rate ?2 ?13 ?14)
- p1489: (defined-below ?2 lels)
- p1712: (defined-below ?13 lels)
- p1714: (defined-below ?14 lels)
- p1715: (mem ?14 classes)
- p1736: (subclass* ?14 devices)
- p1727: (arity ?13 3)
- p1729: (domain ?13 1 ?14)
- p1731: (domain ?13 2 states)
- p1733: (domain ?13 3 numbers)
- p1722: (domain-spec ?13 ?14 states numbers)
- p1718: (mem ?13 functions)
- p1707: (lkk flow-rate ?2 ?13 ?14)

Explanation: This comment explains some of the propositions seen above. Many are easy to understand. These includes the propositions whose relation symbol is subclass, subclass*, arity, domain, domain-spec, mem, etc.; each does exactly what its name implies.

Others have discussed already. These include propositions whose relations are kirchoff1, kirchoff2, kk, ohmslaw, etc. (This list is not comprehensive.)

Propositions of the form (constraint . $\langle p \rangle$) (e.g., p1558) are used to encode

constraints. See Point MRS2 in Subsection 6.2.4.

Each proposition of the form (if (useconstraints $\langle p \rangle$) $\langle p \rangle$) (e.g., p1520) indicates that there are some constraints which deal with the proposition $\langle p \rangle$, and hence one way of solving it is to use the constraint solution method, *Solve*. Again, see Point MRS2 in Subsection 6.2.4, especially Footnote 14 on page 132.

A proposition of the form (defined-below $\langle term \rangle$ $\langle th \rangle$) (e.g., p1489) encodes the fact that the term $\langle term \rangle$ is defined in a theory which is below the theory $\langle th \rangle$. This information is used heavily by the Generator#2. (See Subsection 7.2.2.)

B.3.4 All Analogical Inferences Found — Run#1-1

| | | | | |
|-----|--|-------------|------|-----|
| 1: | rkk(flow-rate, pressure-drop, cross-section, pipes |)+ | 374 | |
| 2: | rkk(flow-rate, pressure-drop, cd-of, | pipes |)+ | 440 |
| 3: | rkk(flow-rate, pressure-drop, cross-section, tubes |)+ | 473 | |
| 4: | rkk(flow-rate, pressure-drop, pipe-charact, | pipes |)* | 535 |
| 5: | rkk(flow-rate, pressure-drop, cross-section, thin-pipes) | | 560 | |
| 6: | rkk(flow-rate, pressure-drop, cross-section, orifices |) | 585 | |
| 7: | rkk(flow-rate, pressure-drop, pump-press, | pumps |) | 624 |
| 8: | rkk(flow-rate, pressure-drop, cd-of, | tubes |)+ | 688 |
| 9: | rkk(flow-rate, pressure-drop, cd-of, | thin-pipes) | | 713 |
| 10: | rkk(flow-rate, pressure-drop, cd-of, | orifices |) | 738 |
| 11: | kk(flow-rate, pressure-drop |) | 973 | |
| 12: | ckk(flow-rate, pressure-drop, water-height, tanks |) | 1318 | |
| 13: | lkk(flow-rate, pressure-drop, water-height, tanks |) | 1675 | |

These abstraction instances are listed in the order in which they were proposed. The number at the end represents the number of the task at which *NLAG* gave up on this this proposed abstraction instance (i.e., either after this analogical inference proved not useful or after final question had been posed). The entries marked with + were useful analogical inferences, and the one tagged with an asterisk, *, is the "correct" answer.

B.4 Data from Other Runs

This subappendix describes the other runs. SubAppendix B.4.1 describes the answers found during Run#1-2, see Subsection 7.2.2. Subappendix B.4.2 presents the data from Run#2a-3 and Run#2a-4, see Section 7.3. The next two parts present the additional additional facts added to the knowledge base for the abstraction sensitivity studies, see Section 7.4. SubAppendix B.4.3 describes the bank of irrelevant abstractions added for Run#2b-2 and SubAppendix B.4.4 describes the relevant abstractions added for Run#2b-3. (These are also used during the Run#3-xb runs.) SubAppendix B.4.5 presents the answers found during Run#2b-3 and Run#3-2b. (Here, all of the abstractions described in SubAppendices B.4.3 and B.4.4 are present.)

B.4.1 All Analogical Inferences Found — Run#1-2

This subappendix describes the analogies found during Run#1-2. Here, *NLAG* used Generator#2, which used defined-below as its primary generator; all other conditions for this second run were the same. Notice exactly the same set of values are returned, but they appear in a slightly different order. Also, the times requires were slightly greater. The data are shown below:

| | | | |
|-----|--|----|------|
| 1: | rkk(flow-rate, pressure-drop, cross-section, pipes |)+ | 525 |
| 2: | rkk(flow-rate, pressure-drop, cross-section, orifices |) | 560 |
| 3: | rkk(flow-rate, pressure-drop, cd-of, pipes |)+ | 614 |
| 4: | rkk(flow-rate, pressure-drop, cross-section, thin-pipes) | | 639 |
| 5: | rkk(flow-rate, pressure-drop, pipe-charact, pipes |)* | 703 |
| 6: | rkk(flow-rate, pressure-drop, cross-section, tubes |)+ | 732 |
| 7: | rkk(flow-rate, pressure-drop, pump-press, pumps |) | 785 |
| 8: | rkk(flow-rate, pressure-drop, cd-of, orifices |) | 810 |
| 9: | rkk(flow-rate, pressure-drop, cd-of, thin-pipes) | | 835 |
| 10: | rkk(flow-rate, pressure-drop, cd-of, tubes |)+ | 887 |
| 11: | kk(flow-rate, pressure-drop |) | 1189 |
| 12: | ckk(flow-rate, pressure-drop, water-height, tanks |) | 1691 |
| 13: | lkk(flow-rate, pressure-drop, water-height, tanks |) | 2201 |

As before, this is the order in which these abstraction instances were proposed. The number at the end represents the number of the task at which *NLAG* gave up on this proposed abstraction instance. The useful analogical inferences are marked with +, and the "correct" answer, with *.

Here, every existential variable used the same generator proposition. As an example:

```
Initial values for ?2: (s0 junctions wire-like drop-of induct-law capac-law ohm-
slaw conserved-thru lkk ckk rkk kirchoff2 kirchoff1 kk port
press-from-ht pump-press water-height pipe-shape cd-of pipe-
charact cross-section flow-rate pressure-drop pressure pressure
units curved straight orifices thin-pipes pipes tubes tanks
pumps devices water-devices ohms coulombs volts amperes volt-
batt inductance capacitance resistance current voltage-drop
voltage inductors capacitors wires resistors batterys elect-
devices j-wc-b j-wc-a pump1 pipe2 pipe1)
```

Using (defined-below ?2 lels)

Table B-3 presents the number of deductions associated with each call to *Resolve-Exists* and *Verify* for this run; this information corresponds to Tables B-1 and B-2.

| Abst | ##? | Resolve-Exists | Verify | \approx_{PT} ? |
|------|-----|----------------|--------|------------------|
| rkk | 1 | 427 | 22 | + |
| | 2 | 18 | 17 | |
| | 3 | 10 | 44 | + |
| | 4 | 8 | 17 | |
| | 5 | 10 | 54 | + |
| | 6 | 8 | 21 | + |
| | 7 | 22 | 31 | |
| | 8 | 8 | 17 | |
| | 9 | 8 | 17 | |
| | 10 | 8 | 44 | + |
| | ?11 | 27 | | |
| | | 554 | 284 | |
| kk | 11 | 169 | 75 | |
| | ?12 | 12 | | |
| | | 181 | 75 | |
| ckk | 12 | 445 | 17 | |
| | ?13 | 23 | | |
| | | 468 | 17 | |
| lkk | 13 | 445 | 17 | |
| | ?14 | 24 | | |
| | | 469 | 17 | |

Table B-3: Deductions for *Resolve-Exists* and *Verify*, using Generator #2

B.4.2 Relevant Data Added — Runs#2a-x

Table B-4 presents the timing data from Run#2a-3 and #2a-4. Table B-5 lists the answers returned during Run#2a-4.

| | Run#2a-3 | Run#2a-4 |
|------------------------|-----------------|----------|
| <i>Find-Kernel:</i> | 33 | 33 |
| <i>Inst-Source:</i> | 14 | 14 |
| <i>CTruep-1:</i> | 22 | 25 |
| <i>Resolve-Exists:</i> | 50 ¹ | 277 |
| <i>Verify:</i> | 53 | 55 |
| TOTALS: ² | 180 | 412 |

Notes:

- 1 Even without existential variable, there is still work to be done. This subtask is responsible for adding in the entailments of each conjecture.
- 2 In both cases, this includes the additional 8 steps of overhead: an initial 4 steps required to start up the overall NLAG process, plus a final 4 steps to "pop" back to the top level.

 Table B-4: Number of Deductions for Run#2a-3 and #2a-4

B.4.3 Irrelevant Abstractions Added — Run#2b-2

Various additional abstractions were added by loading in additional theories. After describing the theories (see below), this subappendix describes the contents of the Algebra theory.

algebra holds facts about algebraic structures. For example, this is where the Group axioms appear, together with descriptions of its component parts, e.g., identity. (This theory includes over 130 propositions.)

number holds additional facts about numbers, sets of numbers (e.g., reals), numeric operators, e.g., +, and interconnections, e.g., (domain * 2 reals) and (monoid reals * 1). (This theory includes over 100 propositions, as well as some 60 justification-links.)

ALGEBRA facts:

- (domain-spec closed binary-operations classes)
- (domain-spec commutative binary-operations)
- (domain-spec associative binary-operations)

| | | | |
|----------------------|---|----|-------|
| 1: | rkk(flow-rate, pressure-drop, pipe-charact, pipes |)* | 408 |
| 2: | rkk(flow-rate, pressure-drop, cd-of, pipes |)+ | 480 |
| 3: | rkk(flow-rate, pressure-drop, pump-press, pumps |) | 519 |
| 4: | rkk(flow-rate, pressure-drop, cd-of, tubes |)+ | 583 |
| 5: | rkk(flow-rate, pressure-drop, cd-of, thin-pipes) | | 608 |
| 6: | rkk(flow-rate, pressure-drop, cd-of, orifices |) | 633 |
| 7: | kk(flow-rate, pressure-drop |) | 895 |
| 8: | ckk(flow-rate, pressure-drop, water-height, tanks |) | 1,228 |
| 9: | lkk(flow-rate, pressure-drop, water-height, tanks |) | 1,553 |
| (Total tasks: 1,618) | | | |

Table B-5: Answers Returned for Run#2a-4

(mem identity functions)

(domain-spec identity binary-operations classes things)

(domain-spec invertible binary-operations classes)

(mem invert-op functions)

(domain-spec invert-op binary-operations unary-operations classes)

(mem inverse functions)

(domain-spec inverse binary-operations binary-operations classes)

(domain-spec distributive binary-operations binary-operations)

(mem monoid abstract-relns)

(domain-spec monoid classes binary-operations things)

(mem group abstract-relns)

(domain-spec group classes binary-operations things)

(mem abelian-group abstract-relns)

(domain-spec abelian-group classes binary-operations things)

(mem ring abstract-relns)

(domain-spec ring classes binary-operations things binary-operations)

```

(mem commutative-ring abstract-relns)
(domain-spec commutative-ring classes binary-operations
                                     things binary-operations)

(mem ring-id abstract-relns)
(domain-spec ring-id classes binary-operations
                                     things binary-operations things)

(mem commutative-ring-id abstract-relns)
(domain-spec commutative-ring-id classes binary-operations things
                                               binary-operations things)

(mem field abstract-relns)
(domain-spec field classes binary-operations
                                     things binary-operations things)

(if (and (mem $x functions)
         (arity $x 3))
    (binary-operations $x))

(if (and (restriction $op . $lst)
         (every $i $lst (= $i $d)))
    (closed $op $d))

(if (and (domain $op 1 $d)
         (bagof $m (domain $op $i $m) $s)
         (every $i $s (= $i $d)))
    (closed $op $d))

(if (closed $op $s)
    (all $a $b in $s (mem ($op $a $b) $s)))

(if (commutative $op)
    (= ($op $a $b) ($op $b $a)))

(if (associative $op)
    (= ($op ($op $a $b) $c) ($op $a ($op $b $c))))

```

```

(if (identity $op $s $id)
  (all $a in $s (= ($op $a $id) $a)))

(if (and (known (identity $op $s $id))12
  (subclass $c $s)
  (mem $id $c) )
  (identity $op $c $id))

(if (and (invertible $op $s)
  (identity $op $s $id))
  (all $x in $s (= ($op $x $y) $id)))

(if (and (invert-op $o $i $s)
  (identity $o $s $i))
  ($o $x ($i $x) $1))

(if (and (invert-op $o $i $s)
  (subclass* $c $s))
  (invertible $o $c))

(if (inverse $o $i $d)
  (if ($o $x $y $z) ($i $z $y $x)))

(if (inverse $o $i $d)
  (invertible $o $d))

(if (inverse $o $i $d)
  (if (and ($o $x $y $z) (mem $y $d)) ($i $z $y $x)))

(if (distributive $o1 $o2)
  (and (= ($o2 $a ($o1 $b $c)) ($o1 ($o2 $a $b) ($o2 $a $c)))
    (= ($o2 ($o1 $b $c) $a) ($o1 ($o2 $b $a) ($o2 $c $a))))))

```

¹²The known operator is part of the initial MRS system. This clause is true if a proposition unifying with (identity \$op \$s \$id) has been primitively stored. This additional constraint prevents MRS's back-chainer from going into infinite loops.

And the abstractions,

```

(if (and (identity $op $s $i)
         (closed $op $s)
         (mem $s classes)
         (binary-operations $op)
         (mem $i $s)
         (associative $op))
    (monoid $s $op $i))

(if (and (invertible $op $s)
         (monoid $s $op $i))
    (group $s $op $i))

(if (and (group $s $op $i)
         (commutative $op))
    (abelian-group $s $op $i))

(if (and (abelian-group $s $o1 $i1)
         (associative $o2)
         (binary-operations $o2)
         (distributive $o1 $o2)
         (closed $o2 $s))
    (ring $s $o1 $i1 $o2))

(if (and (ring $s $o1 $i1 $o2)
         (commutative $o2))
    (commutative-ring $s $o1 $i1 $o2))

(if (and (ring $s $o1 $i1 $o2)
         (identity $o2 $s $i2))
    (ring-id $s $o1 $i1 $o2 $i2))

(if (and (ring-id $s $o1 $i1 $o2 $i2)
         (commutative $o2))
    (commutative-ring-id $s $o1 $i1 $o2 $i2))

```

```
(if (and (commutative-ring-id $s $o1 $i1 $o2 $i2)
        (not (= $i1 $i2))
        (set- $s $i2 $r)
        (invertible $o2 $r))
    (field $s $o1 $i1 $o2 $i2))
```

All of these abstractions have been from formal system. There is also a less technical abstraction. (This is usually resident in genl-info.)

```
(mem historical-event abstract-relns)
(domain-spec historical-event events person time)
(if (historical-event $type $bywhom $when)
    (if (and ($bywhom $thing $whom)
            ($when $thing $date))
        (AliveAt $whom $date)))
```

B.4.4 Relevant Abstractions Added — Runs#2b-3 and #3-xb

These abstractions and related facts are usually resident in the LELS theory.

```
(if (and (rkk $tt $ct $rt $load)
        (induct-Law $tt $ct $indt $ind-load) )
    (rlkk $tt $ct $rt $load $indt $ind-load))

(if (and (rkk $tt $ct $rt $load)
        (capac-Law $tt $ct $indt $ind-load) )
    (rckk $tt $ct $rt $load $indt $ind-load))

(if (and (lkk $tt $ct $rt $load)
        (capac-Law $tt $ct $scapt $scap-load) )
    (lckk $tt $ct $rt $load $scapt $scap-load))

(if (and (rckk $tt $ct $rt $load $indt $ind-load)
```

```

      (induct-Law $tt $ct $capt $cap-load) )
      (rlckk $tt $ct $rt $load $capt $cap-load $indt $ind-load))

(mem rlkk abstract-relns)
(domain-spec rlkk functions functions functions classes functions classes)

(mem rckk abstract-relns)
(domain-spec rckk functions functions functions classes functions classes)

(mem lckk abstract-relns)
(domain-spec lckk functions functions functions classes functions classes)

(mem rlckk abstract-relns)
(domain-spec rlckk functions functions functions classes
              functions classes functions classes)

```

B.4.5 Results Found, Using all Eight Relevant Abstractions

This subappendix describes the analogies *NLAG* finds during Run#2b-3 and Run#3-2b, mentioned in Sections 7.4 and 7.5, respectively. In both cases, all eight lumped element linear system abstractions are present. The first table describes Run#2b-3, in which the H_{MGA} rule is turned on. (Hence rkk appears first.) The second table describes Run#3-2b, in which the H_{MGA} rule is turned off. (Here the most specific abstraction, rlckk appears first.)

(The “...” below always encode “flow-rate, pressure-drop”, and is used only to fit this information on a single line.)

Run#2b-3's Data:

| | | | |
|-----|---|-----|------|
| 1: | rkk(..., cross-section, pipes |) + | 374 |
| 2: | rkk(..., cd-of, pipes |) + | 440 |
| 3: | rkk(..., cross-section, tubes |) + | 473 |
| 4: | rkk(..., pipe-charact, pipes |) * | 535 |
| 5: | rkk(..., cross-section, thin-pipes |) | 560 |
| 6: | rkk(..., cross-section, orifices |) | 585 |
| 7: | rkk(..., pump-press, pumps |) | 624 |
| 8: | rkk(..., cd-of, tubes |) + | 688 |
| 9: | rkk(..., cd-of, thin-pipes |) | 713 |
| 10: | rkk(..., cd-of, orifices |) | 738 |
| 11: | rlkk(..., cross-section, pipes, water-height, tanks |) + | 1308 |
| 12: | rlkk(..., cross-section, tubes, water-height, tanks |) + | 1340 |
| 13: | rlkk(..., cd-of, pipes, water-height, tanks |) + | 1709 |
| 14: | rlkk(..., pipe-charact, pipes, water-height, tanks |) + | 1778 |
| 15: | rlkk(..., cd-of, tubes, water-height, tanks |) + | 1833 |
| 16: | rlkk(..., cross-section, thin-pipes, water-height, tanks) | | 2007 |
| 17: | rlkk(..., cross-section, orifices, water-height, tanks |) | 2034 |
| 18: | rlkk(..., pump-press, pumps, water-height, tanks |) | 2367 |
| 19: | rlkk(..., cd-of, thin-pipes, water-height, tanks) | | 2418 |
| 20: | rlkk(..., cd-of, orifices, water-height, tanks |) | 2453 |
| 21: | rckk(..., cross-section, pipes, water-height, tanks |) + | 3416 |
| 22: | rckk(..., cross-section, tubes, water-height, tanks |) + | 3718 |
| 23: | rckk(..., cd-of, pipes, water-height, tanks |) + | 3547 |
| 24: | rckk(..., pipe-charact, pipes, water-height, tanks |) + | 3616 |
| 25: | rckk(..., cd-of, tubes, water-height, tanks |) + | 3671 |
| 26: | rckk(..., cross-section, thin-pipes, water-height, tanks) | | 3845 |
| 27: | rckk(..., cross-section, orifices, water-height, tanks |) | 4872 |
| 28: | rckk(..., pump-press, pumps, water-height, tanks |) | 4205 |
| 29: | rckk(..., cd-of, thin-pipes, water-height, tanks) | | 4256 |
| 30: | rckk(..., cd-of, orifices, water-height, tanks |) | 4291 |

| | | | |
|-----|--|----|-------|
| 31: | rlckk(..., cross-section, pipes, water-height, tanks |)+ | 8875 |
| 32: | rlckk(..., cross-section, tubes, water-height, tanks |)+ | 8904 |
| 33: | rlckk(..., cd-of, pipes, water-height, tanks |)+ | 15028 |
| 34: | rlckk(..., pipe-charact, pipes, water-height, tanks |)+ | 15102 |
| 35: | rlckk(..., cd-of, tubes, water-height, tanks |)+ | 15162 |
| 36: | rlckk(..., cross-section, thin-pipes, water-height, tanks) | | 16427 |
| 37: | rlckk(..., cross-section, orifices, water-height, tanks |) | 16470 |
| 38: | rlckk(..., pump-press, pumps, water-height, tanks |) | 20493 |
| 39: | rlckk(..., cd-of, thin-pipes, water-height, tanks) | | 20546 |
| 40: | rlckk(..., cd-of, orifices, water-height, tanks |) | 20611 |
| 41: | kk(... |) | 22013 |
| 42: | ckk(..., water-height, tanks |) | 22347 |
| 43: | lkk(..., water-height, tanks |) | 22707 |
| 44: | lckk(..., water-height, tanks, water-height, tanks |) | 23247 |

The final figures were

Timing: 23758

((bcdisp . 22829) (succeed . 492) (ff-disp . 18) (fc-only . 16)
 (conj-bc . 10) (fcdisp . 386) (check-out . 7))

Run#3-2b's Data:

| | | | |
|-----|--|----|-------|
| 1: | rlckk(..., cross-section, pipes, water-height, tanks |)+ | 3130 |
| 2: | rlckk(..., cd-of, pipes, water-height, tanks |)+ | 8502 |
| 3: | rlckk(..., cross-section, tubes, water-height, tanks |)+ | 8575 |
| 4: | rlckk(..., pipe-charact, pipes, water-height, tanks |)+ | 8645 |
| 5: | rlckk(..., cross-section, thin-pipes, water-height, tanks) | | 8676 |
| 6: | rlckk(..., cross-section, orifices, water-height, tanks |) | 8707 |
| 7: | rlckk(..., pump-press, pumps, water-height, tanks |) | 14694 |
| 8: | rlckk(..., cd-of, tubes, water-height, tanks |)+ | 14802 |
| 9: | rlckk(..., cd-of, thin-pipes, water-height, tanks) | | 14833 |
| 10: | rlckk(..., cd-of, orifices, water-height, tanks |) | 14864 |
| 11: | rlkk(..., cross-section, pipes, water-height, tanks |)+ | 18004 |
| 12: | rlkk(..., cross-section, tubes, water-height, tanks |)+ | 18029 |
| 13: | rlkk(..., cd-of, pipes, water-height, tanks |)+ | 18404 |
| 14: | rlkk(..., pipe-charact, pipes, water-height, tanks |)+ | 18471 |
| 15: | rlkk(..., cd-of, tubes, water-height, tanks |)+ | 18526 |
| 16: | rlkk(..., cross-section, thin-pipes, water-height, tanks) | | 18706 |
| 17: | rlkk(..., cross-section, orifices, water-height, tanks |) | 18739 |
| 18: | rlkk(..., pump-press, pumps, water-height, tanks |) | 19076 |
| 19: | rlkk(..., cd-of, thin-pipes, water-height, tanks) | | 19123 |
| 20: | rlkk(..., cd-of, orifices, water-height, tanks |) | 19158 |
| 21: | rckk(..., cross-section, pipes, water-height, tanks |)+ | 19753 |
| 22: | rckk(..., cross-section, tubes, water-height, tanks |)+ | 19787 |
| 23: | rckk(..., cd-of, pipes, water-height, tanks |)+ | 19930 |
| 24: | rckk(..., pipe-charact, pipes, water-height, tanks |)+ | 19999 |
| 25: | rckk(..., cd-of, tubes, water-height, tanks |)+ | 20056 |
| 26: | rckk(..., cross-section, thin-pipes, water-height, tanks) | | 20208 |
| 27: | rckk(..., cross-section, orifices, water-height, tanks |) | 20257 |
| 28: | rckk(..., pump-press, pumps, water-height, tanks |) | 20650 |
| 29: | rckk(..., cd-of, thin-pipes, water-height, tanks) | | 20723 |
| 30: | rckk(..., cd-of, orifices, water-height, tanks |) | 20766 |

| | | | |
|-----|---|----|-------|
| 31: | rkk(..., cross-section, pipes |)+ | 21413 |
| 32: | rkk(..., cross-section, tubes |)+ | 21442 |
| 33: | rkk(..., cd-of, pipes |)+ | 21502 |
| 34: | rkk(..., pipe-charact, pipes |)* | 21566 |
| 35: | rkk(..., cd-of, tubes |)+ | 21618 |
| 36: | rkk(..., cross-section, thin-pipes |) | 21659 |
| 37: | rkk(..., cross-section, orifices |) | 21690 |
| 38: | rkk(..., pump-press, pumps |) | 21735 |
| 39: | rkk(..., cd-of, thin-pipes |) | 21766 |
| 40: | rkk(..., cd-of, orifices |) | 21795 |
| 41: | lckk(..., water-height, tanks, water-height, tanks) | | 22324 |
| 42: | ckk(..., water-height, tanks |) | 23149 |
| 43: | lkk(..., water-height, tanks |) | 23509 |
| 44: | kk(... |) | 23766 |

The final number of deductions was 23,786.

Appendix C

Glossary

This appendix defines many of the terms and symbols used in this dissertation. Subappendix C.1 first presents some of the notational conventions we follow. Subappendix C.2 then lists many of the symbols, terms and phrases, and supplies the relevant definition or pointer.

C.1 Notation and Conventions

This section describes some of the notation and conventions used throughout this dissertation. The particular conventions used just in Chapter 8 appears at the end.

Fonts:

Base-level facts (13)

Base level facts — *i.e.*, symbols the user can type and see — appear in this fixed-width font.

abcd (15)

Constants (including constant, function and relation symbols) are indicated by English letters in this *math* style.

$\$univ$ (140)

Universal variables are prefaced with a dollar sign, “\$”.

?exist (15)

Existential variables are prefaced with a question mark, “?”.

\vec{x} (64)

Refers to the vector $[x_1, \dots, x_n]$. Often used in reference to some list of instantiations.

σ (13)

Propositions are indicated by small Greek letters, e.g., σ , γ .

Σ (82)

Sets of propositions are indicated by capital Greek letters, e.g., Σ , Γ .

Group (55)

The names of abstractions appear in this **bold-face fixed-width font**; e.g., **Group**, **RKK**.

TRUEP (111)

The name of both subroutines and various internal processes appear in this *slanted fixed-width font*. E.g., *NLAG*, *ComAbs*.

\mathcal{C} (167)

These calligraphic symbols usually denote interpretations, especially \mathcal{I} , \mathcal{J} , \mathcal{K} and \mathcal{P} .

See the description of Partial Interpretations in Section 8.2.

The notation defined in Chapter 8 follows:

\mathcal{C} is syntactic symbol.

$\boxed{\mathcal{C}}$ is supposed to “be” the actual object in the world, rather than just a syntactic symbol (which designates some such object).

\mathcal{C}^+ is semantic, referring to a collection of objects. In particular, it refers to the positive extension of the symbol (or label) \mathcal{C} .

\mathcal{C}^- is semantic, referring to a collection of objects. In particular, it refers to the negative extension of the symbol (or label) \mathcal{C} .

\mathcal{C}^I is semantic, referring to a collection of objects. In particular, it refers to the unknown extension of the symbol (or label) \mathcal{C} . See Equation 8.3 on page 167.

${}_+\mathcal{C}$, ${}_-\mathcal{C}$, \mathcal{C} are each semantic, each referring to a collection of objects corresponding to ${}_+\mathcal{C}^P$, ${}_-\mathcal{C}^P$ and \mathcal{C}^P respectively, with respect to the understood interpretation, \mathcal{P} .

$\underline{\mathcal{C}}^I$ refers to the pair of positive and negative extensions of the symbol (or label) \mathcal{C} , with respect to the (partial) interpretation I , $\langle {}_+\mathcal{C}^I \quad {}_-\mathcal{C}^I \rangle$.

$\underline{\mathcal{C}}$ refers to the pair of positive and negative extensions of the symbol (or label) \mathcal{C} , $\langle {}_+\mathcal{C}^P \quad {}_-\mathcal{C}^P \rangle$, with respect to the understood interpretation, \mathcal{P} . When dealing with constants, it often describes the unique referent of that symbol.

C.2 Glossary of Terms

Standard Logical Symbols:

\vdash (A-36)

Refers to the syntactic operation of proving; $\Sigma \vdash \sigma$ means that σ is provable from Σ . It may refer to some incomplete or unsound derivation process. See the *IM* entry.

\models (14)

Refers to the semantic notion of logical implication; $\Sigma \models \sigma$ means that σ can be logically inferred from Σ . It is equivalent to any complete and sound derivation process.

$\not\models$ (13)

Refers to the *negation* of logical implication; $\Sigma \not\models \sigma$ means that σ can not be logically inferred from Σ . In general, the “slash” operator, $/$, negates the operation through which it is slashing.

$\&$ (19)

Refers to the logical “and” operation, *i.e.*, conjunction.

- \vee (A-6)
Refers to the logical "or" operation, i.e., disjunction.
- \neg (13)
Refers to the logical "not" operation, i.e., negation.
- \bigwedge_j (121)
Refers to conjunction over a set of indexed propositions.
- \bigcap_i (174)
Refers to intersection over a set of indexed sets.
- \equiv (33)
"Is (logically) equivalent to"
Used to compare two propositions; e.g., " $\rho \& \sigma \equiv \sigma \& \rho$."
- \iff (16)
"If and only if"
Used to compare two propositions; e.g., " $\rho \& \sigma \iff \sigma \& \rho$ ".
- \implies (20)
"Implies"
Used to compare two propositions; e.g., " $\rho \& \sigma \implies \rho$ "
Also, when \implies and \impliedby are used in equivalence proofs, they refer to the sufficiency and necessity conditions, respectively.
See the \impliedby entry.
- \impliedby (A-13)
"Is implied by"
Used to compare two propositions; e.g., $\rho \impliedby \rho \& \sigma$
Also, when \implies and \impliedby are used in equivalence proofs, they refer to the sufficiency and necessity conditions, respectively.
See the \implies entry.
- \square (A-13)
"Quo Eratis Demonstratus"
indicates that the proof is complete.

$\phi[x/y]$ (A-13)

Indicates lexical substitution of the symbol y for the symbol x within the formula ϕ .

\subset (29)

Refers to proper subset.

See the \subseteq entry.

\subseteq (31)

Refers to the subset, which need not be proper.

See the \subset entry.

$=$ (24)

Refers to equality in general. When comparing formulae and sentences, the expression $\rho(Z) = \sigma$ means that the formula ρ "expands" into the sentence σ when the term Z is substituted for ρ 's free variable. This use of " $=$ " is distinguished from the more general use of " \equiv ", which denotes logical equivalence.

$+$ (23)

The notation $\Sigma + \sigma$ abbreviates the larger set, $\Sigma \cup \{\sigma\}$. (Of course, Σ can be a set of axioms, i.e., a theory.)

$-$ (77)

The notation $\Sigma - \Gamma$ is the set difference between the sets Σ and Γ ; i.e., $\Sigma - \Gamma = \{\sigma \in \Sigma \mid \sigma \notin \Gamma\}$.

Special Symbols:

\sim (13)

Read "is like".

See $A \sim B$ entry.

\vdash (13)

Refers to the *general analogical inference* operator, defined in Figure 2-1.

\vdash_{PT} (39)

Refers to the *useful analogical inference* operator, defined in Figure 3-1.
See the \vdash entry.

\Vdash (52)

The *abstraction-based analogical inference* operator.
See the \Vdash_{PT} entry.

\Vdash_{PT} (52)

The *abstraction-based useful analogical inference* operator, defined in Figure 4-5.
See the \Vdash entry.

\downarrow_i (15)

Refers to the projection operator. See Equation 2.7 on page 15 and Definition 22 on page 169.

φ (13)

The symbol φ , with or without some subscript (e.g., φ_{RKK}) generally refers to some “analogy formula”. By convention, the analogues occupy its “ i^{th} ” argument.

φ_{RKK} (19)

Refers to a particular analogy formula; see Figure 2-3.
See also **RKK** entry.

\succ_{LC} (88)

Resembles the $\succ_{[Th]}^{LC}$ relationship, but with an implicit theory.
See the $\succ_{[Th]}^{LC}$ entry.

$\succ_{[Th]}^{LC}$ (90)

The expression $\sigma_1(A) \succ_{[Th]}^{LC} \sigma_2(A)$ abbreviates *LessConstraints_{sem}*($\sigma_1, \sigma_2 Th$).
See the *LessConstraints_{sem}* entry.

$\succ_{[Th]}^s$ (192)

The expression $\varphi_1 \succ_{[Th]}^s \varphi_2$ means that the unary formula φ_1 is more specific than φ_2 (based on information in the theory Th).
See Definition 30 on page 192.

\sqsubseteq (171)

Refers to the extension-subset relation.

See Definition 23 on page 171.

 $=_A$ (A-46)

The expression $I =_A J$ means the interpretations I and J are identical, except possibly for their respective assignments to the symbol A .

See Definition 36 on page A-46.

 I^S/A (A-46)

Refers to a partition of the set of interpretations S induced by $=_A$.

See Definition 37 on page A-46.

 $\stackrel{E}{\supseteq}$ (172)

The expression $I \stackrel{E}{\supseteq} J$ means that the interpretation I is a consistent extension of J .

See Equation 8.9 on page 172.

 $\Delta(Th, \sigma, \Psi)$ (61)

Means that the set of conjectures Ψ must be added to the theory Th to deduce the sentence σ .

See Definition 13 on page 61.

 $\Delta_{LR}(Th, \sigma, \Psi)$ (122)

Holds when Ψ is a leaf residue of the sentence σ with respect to the theory Th .

See Definition 21 on page 122.

Alphabetical Listing:

 $A \sim B$ (13)

Refers to our canonical analogical hint, "A is like B".

See the *analogical hint* entry.

AbelianGroup (152)

Refers to the abelian group abstraction defined in the Algebra theory; see SubAppendix B.4.3.

AbstForm(φ) (56)

Means that the formula φ is an abstraction formula.

See Definition 11 on page 56.

See also the *Abstraction Formula* entry.

abstraction (56)

An abstraction is an important pre-specified relation. Examples include **Group** and **RKK**. See Chapter 4.

abstraction instance (51)

Refers to an instantiation of an abstraction. *E.g.*, (group reals + 0) is an *abstraction instance*, of the abstraction **Group**. Each abstraction instance is equivalent to a perspective.

See the *Perspective* entry.

abstraction formula (56)

Refers to an atomic formula whose relation symbol is an abstraction.

See the *AbstForm* entry.

abstraction-based analogical inference (52)

Refers to the $\|_{PT}$ process which seeks a common abstraction between a pair of given analogues, conjecturing new facts about the target analogue as necessary. (Technically, this is the *abstraction-based useful analogical inference process*.)

See Figure 4-5.

abstraction-based analogy (57)

Refers to the result of the $\|_{PT}$ process; *viz.*, the analogically-derived new conjecture about the target analogy. Unlike analogies in general, this conjecture is always an instance of an abstraction.

(an) abstraction holds for a concept (80)

We say “*the abstraction S holds for the concept A*” if there is some instantiation of **S** which includes **A** as a parameter. *E.g.*, “**Group** holds for +”, as **Group**($\mathcal{R}, +, 0$) holds.

AH (110)

Refers to the Analogical Hint, e.g., $A \sim B$.

See the *Analogical Hint* entry.

Allowed(\mathcal{RW} , Th) (172)

Refers to the set of allowed interpretations.

See Definition 24 on page 172.

analogical hint (13)

A statement of the form “A is like B”, often represented as $A \sim B$.

analogical inference (13)

Any of a variety of plausible inferences.

Flavors include:

- General analogical inference (see Figure 2-1)
- Useful analogical inference (see Figure 3-1)
- Abstraction-based (useful) analogical inference (see Figure 4-5)

analogical inference situation (202)

Refers to the initial situation, from which an analogical inference can be drawn.

It includes a specification of the analogical hint and initial knowledge base; and possibly a specific problem to solve as well.

Analogous (22)

We say that the analogues A and B are *analogous* if they share a common formula.

See the *Analogy_F* entry.

Analogous_χ (58)

By convention, *Analogous_χ* is the “exists-form” of the corresponding *Analogy_χ*

— i.e., $Analogous_{\chi}(A, B, Th) \Leftrightarrow \exists \Sigma Analogy_{\chi}(A, B, Th, \Sigma)$.

analogy (17)

Often used in reference to the *Analogy_F* relation or the \vdash process. It can also

refer to the “result” of that process, i.e., $\varphi(a_1, \dots, A, \dots, a_n)$.

analogy formula (17)

The formula used to connect the analogues.

Analogy_{CA} (56)

“The common abstraction relationship”.

See Definition 12 on page 56.

Analogy_F (22)

The formula-based definition of analogy.

See Definition 2 on page 22.

Analogy_{Sem} (178)

The semantic definition of analogy.

See Definition 27 on page 178.

Analogy_T (A-11)

The theory-based definition of analogy.

See Definition 33 on page A-11.

assertion (25)

Refers to an element in the theory. It is synonymous with *proposition* or *fact*.

See the *fact* entry.

AtomicFormula(ϕ) (56)

Means that ϕ is a single relationship; i.e., the formula ϕ has no boolean connectives. (As opposed to *compound formula*.)

base interpretation (172)

Refers to the “real world” interpretation, \mathcal{RW} , as used in $Allowed(\mathcal{RW}, Th)$.

See the *Allowed* entry.

base-level fact (117)

Refers to facts about the particular domain, e.g., about hydraulics. By contrast, a *meta-level fact* is a fact about some propositions.

See the *meta-level fact* entry.

binding (100)

Refers to the instantiation of some parameter(s) in a formula.

buef (33)

Refers to the class of base unary existential atomic formulae. See Definition 9 on page 33.

buf (34)

Refers to the class of base unary atomic formulae.

See Definition 10 on page 34.

CKK (140)

Refers to the Capacitor abstraction defined in the LELS theory; see SubAppendix B.2.1.

cohesive (65)

A collection of facts is considered *cohesive* if a subset of these facts suggest the others.

ComAbs (111)

Refers to the module of the NLAG system which finds the common abstraction between the source and target analogues. See Subsection 6.1.2.

Common (13)

Refers to the condition imposed by the analogical inference process which insists that the new conjecture corresponds to some fact known about the source analogue (i.e., it means that the inference is an analogy).

See Figure 2-1 on page 16.

common abstraction (53)

S is a *common abstraction* to A and B if both $S(a_1, \dots, A, \dots, a_n)$ and $S(b_1, \dots, B, \dots, b_n)$ hold, where S is an abstraction and A and B occur in the corresponding position.

E.g., **Group** is a common abstraction linking $+$ and $*$, as **Group**($\mathcal{R}, +, 0$) and **Group**($\mathcal{R}_0, *, 1$) each hold.

See the *Analogy_{CA}* and \mathbb{H}_{PT} entries.

CommutativeRing (152)

Refers to the commutative ring abstraction defined in the Algebra theory; see SubAppendix B.4.3.

CommutativeRingId (152)

Refers to the commutative ring with identity abstraction defined in the Algebra theory; see SubAppendix B.4.3.

concept (1)

Refers to some syntactic symbol (or string of symbols); which, in turn, refer to some object in the world.

conjecture (1)

Refers to a proposed addition to the theory. Also called "proposed element of the data base".

Content (of representation) (187)

Refers to the actual information encoded, as opposed to the "form" of that information.

Consistent (13)

Refers to the condition imposed by all of the analogical inference processes which insists that the new conjecture be consistent with the facts present in the initial theory.

See Figure 2-1 on page 16.

Consistent(Σ) (60)

Holds when the set of propositions, Σ , is consistent.

constraint (132)

Each constraint is a type of proposition used to constrain the values of various terms.

See Point MRS2 in Subsection 6.2.4.

See also the *Solve* entry.

Current (8)

Refers to the quantity which relates to electron flow through an electric device, in the electrical domain.

 $DC(\Sigma)$ (207)

Refers to the deductive closure of the set of propositions, Σ .

deducible (14)

A proposition ρ is *deducible* from a collection of axioms, Σ , if $\Sigma \models \rho$.

 $Dom(f)$ (A-11)

Refers to the domain of the function f .

domain (9)

This term is used in three senses in this dissertation:

- [1] as in “source *domain*” or “target *domain*”
- [2] as in *domain* type specifications.
- [3] as opposed to “range” (related to Sense [2]).

EC (8)

Refers to the subset of the theory which pertains to electric circuits; i.e., $EC \subset Th_{CF}$, where Th_{CF} is that initial theory. For this application, EC is the *source theory*, as it describes the source analogue, Current.

See also the *FS* and *source theory* entries.

extension (167)

Extension with a “s” refers to the result of enlarging (extending) a set.

See also the *Extention* entry.

extention (167)

Extention with a “t” refers to the real world object(s) corresponding to some symbol.

See also the *Extension* entry.

fact (1)

A *fact* is a proposition, one which is an element of a knowledge base. This term refers to base-level data, as well as rules, constraints, meta-level information, etc. This term is synonymous with *assertion*. (This is as opposed to *conjecture* or *postulate*, which are known not be in the knowledge base, and to *proposition* or *clause*, which are neutral in this regard.)

FC (118)

The forward chain module of the MRS system; see [Rus85].

Field (56)

Refers to the field abstraction defined in the Algebra theory; see SubAppendix B.4.3.

"Find the flowrate" (9)

Refers to our standard target problem.

See *target problem* entry.

Find-Kernel (114)

Refers to the module of the NLAG system which finds the kernel from which source instantiations of the abstraction can be derived. See Subsection 6.1.3.

See also the *kernel* entry.

first-order fact (20)

Refers to facts whose arguments pertain to objects rather than function or relations. By contrast, *second-order facts* may quantify over functions; e.g., $\text{Kirchoff1}(\text{FlowRate})$ is a second order fact as the argument, *FlowRate*, is itself a function. However, $\text{Member}(5 \text{ Numbers})$ is a first order fact.

See the *second-order fact* entry.

FlowRate (8)

Refers to the quantity which relates to the flow through a hydraulics device, in the hydraulics domain.

Form (of representation) (187)

Refers to how some information is encoded within a formalism. In particular, this considers which facts are explicit (*prima facie*) and which are implicit (require deduction). (This phrase refers to the actual information encoded, as opposed to the "form" of that information; i.e., it is a property of the representation, not of the formalism.)

See the *representation* and *formalism* entries.

formalism (187)

Refers to a way of encoding facts. Hence fully indexed predicate calculus is one formalism; another is the *Unit:Slot = Value* encoding. (This differs from "representation".)

formula (13)

A well-formed syntactic expression with zero or more free variables. (A formula with no free variable is called a *sentence*.)

FS (8)

Refers to the subset of the theory which pertains to fluid systems; i.e., $FS \subset Th_{CF}$, where Th_{CF} is that initial theory. For this application, FS is the *target theory*, as it describes the target analogue, *FlowRate*.

See also the *EC* and *target theory* entries.

function (108)

A *function* is a mathematical entity. To distinguish a function from the code used to implement it, we refer to the latter as a *subroutine* or *program*.

gbuf (A-21)

Refers to a *general base unary formula*; see page A-21.

general analogical inference (16)

Refers to the process which adds independent facts to a theory Th , where these additions establish an analogical connection between the given analogues A and B. See Figure 2-1 on page 16.

generality (of a formula) (198)

Refers to the number of terms a formula can accept; more general formulae accept more terms than less general ones.

See the $\overset{s}{>}_{[Th]}$ entry.

Group (56)

Refers to the group abstraction defined in the Algebra theory; see SubAppendix B.4.3.

 H_{Abst} (54)

This heuristic is used by the analogical inference process to prune its search; here, to just consider abstraction formulae.

See Heuristic 1 on page 57.

 H_{CC} (78)

This heuristic is used by the abstraction-based analogical inference process to prune its search; here, to insist that all terms used to instantiate the abstraction come from the same theory.

See Heuristic 3 on page 79.

 H_{FC} (94)

This heuristic is used by the analogical inference process to order its search; here, to prefer analogies which require that fewer conjectures be postulated.

See Heuristic 5 on page 97.

 H_{FT} (101)

This heuristic is used by the analogical inference process to prune its search; here, to insist that all terms used to instantiate the abstraction be “findable”; i.e., be involved in some proposition relevant to the abstraction.

See Heuristic 6 on page 101.

 H_{JK} (74)

This heuristic used by the useful analogical inference process to focus its search;

here, to use the target problem to suggest which abstractions to consider.
See Heuristic 2 on page 76.

 H_{MGA} (92)

This heuristic is used by the abstraction-based analogical inference process to order its search; here, to prefer the more general abstractions.
See Heuristic 4 on page 93.

 H_{MSA} (193)

This heuristic is not used. It suggests ordering the search for common abstractions by preferring the most specific abstractions.
See Heuristic 7 on page 193.

HistoricalEvent (152)

Refers to the historical event abstraction defined in the GenIInfo theory; see SubAppendix B.4.3.

 I_{Close} (43)

Refers to the intuition that the source of a useful analogy is composed of facts which form a *coherent* cluster.
See the *coherent*, I_{Least} and I_{Most} entries.

iff (83)

Refers to the connective, "if and only if".
See the \iff entry.

 I_{Least} (43)

Refers to the intuition that a useful analogy should add as few new conjectures as possible.
See the *LessConstraints_{Sem}*, I_{Close} and I_{Most} entries.

IM (121)

Refers to some inference mechanism. For example, it can denote the particular implementation of MRS's backward-chaining system.
See the \vdash entry.

I_{Most} (43)

Refers to the intuition that a useful analogy should add as many new conjectures as possible.

See the $\overset{s}{>}_{[Th]}$, I_{Close} and I_{Least} entries.

independent (13)

A proposition ρ is *independent* of a collection of axioms, Σ , if $\Sigma \not\vdash \rho$ and $\Sigma \not\vdash \neg\rho$.

Inst-Source (114)

Refers to the module of the *NLAG* system which finds the source instantiations of the relevant abstractions. See Subsection 6.1.4.

Inst-Target (115)

Refers to the module of the *NLAG* system which finds the target instantiations of the relevant abstractions. See Subsection 6.1.5.

instance (45)

Synonymous with *instantiation*.

See the *instantiation* entry.

instantiation (22)

An *instantiation* of a formula is a binding list for its arguments. Hence, an *instantiation* of **Group** might be the binding list $[\mathfrak{R}, +, 0]$.

See the *source instantiation* and *target instantiation* entries.

instantiation (of φ) in the source domain (17)

Refers to the *source analogy sentence*.

instantiation (of φ) in the target domain (17)

Refers to the *target analogy sentence*.

instantiate the abstraction (in the source domain) (17)

Refers to a binding list of the abstraction which involves the source analogue.

instantiate the abstraction (in the target domain) (17)

Refers to a binding list of the abstraction which involves the source analogue.

interpretation (165)

Shorthand for *partial interpretation*. This forms a proper superset of *Tarskian Interpretations*. See Section 8.2.

 j_d^i (18)

Refers to the i^{th} junction associated with the device d .

justification (31)

The *justification* of a proposition, ρ , is any collection of propositions, $\{\sigma_i\}$, such that $\{\sigma_i\} \models \rho$. Used synonymously with *support set*.

See the *support set* entry.

kernel (113)

The *kernel* is a subset of facts from the initial theory. These are used to derive source instances of the relevant abstractions. The H_{JK} heuristic derives these kernel sentences from the target problem and analogical hint.

Kirchoff's Laws (8)

Originally referred only to a pair of facts in the electricity domain.

1. The algebraic sum of all potential drops around a close loop is 0.
2. The algebraic sum of all currents flowing into a junction is 0.

Subsequently generalized to deal with general "through" and "cross" variables. See **KK** entry.

KK (143)

Refers to the Kirchoff's Laws abstraction defined in the LELS theory; see Sub-Appendix B.2.1.

See also the *Kirchoff's Laws* entry.

knowledge base (9)

Used synonymously with *theory*.

\mathcal{L} (167)

Refers to a language (of a theory).

LCKK (153)

Refers to the Inductance and Capacitance abstraction defined in the LELS theory; see SubAppendix B.4.4.

 $LD(\sigma, Th, \Sigma)$ (121)

Holds when Σ is a leaf decomposition of the sentence σ with respect to the theory Th .

See Definition 20 on page 121.

 $leaf_{IM}(\delta, Th)$ (121)

Means that the sentence δ is not “decomposable” with respect to the theory Th and inference mechanism IM .

See Definition 19 on page 121.

leaf decomposition (121)

Refers to a decomposition of a given relation which consists of only leaf propositions.

See Subsection 6.2.3.

See the LD entry.

leaf proposition (121)

See the $leaf_{IM}(\delta, Th)$ entry.

leaf residue (121)

Refers to a type of residue, one consisting of only leaf clauses.

See the Δ_{LR} entry.

 $Learn_1$ (A-6)

Refers to the class of learning processes which add independent conjectures to a theory. Afterwards, the basic performance engine can solve new problems.

See Note:2-2.

Learn₂ (A-7)

Refers to the class of learning processes which learn to solve a particular set of problems more efficiently.

See Note:2-2.

learner (8)

This model of analogy deals explicitly with a *learner* acquiring additional information. The theory, *Th*, contains the collection of facts we assume he (or she or it) initially knows, before the analogical inference process.

learning by analogy (4)

Refers to the common term for "analogical inference".

LessConstraints_{sem}($\sigma_1, \sigma_2 Th$) (90)

Holds when the sentence σ_1 constrains the known world (as encoded by the theory *Th*) less that the sentence σ_2 .

See Definition 17 on page 90.

LexInclusion(s, " σ ") (24)

means that the symbol *s* is lexically included in the sentence σ .

See page 24.

LKK (140)

Refers to the Inductance abstraction defined in the LELS theory; see SubAppendix B.2.1.

LOOKUP(s) (112)

LOOKUP is one of MRS's basic querying subroutines (the other is **TRUEP**). It takes a proposition pattern as input and returns the facts which are primitively stored in the theory and match this pattern.

See the **TRUEP(s)** and *primitively stored* entries.

mem (127)

The MRS expression (**mem** $\langle a \rangle \langle c \rangle$) is true if the concept $\langle a \rangle$ is a member of the class $\langle c \rangle$.

meta-level fact (117)

Refers to facts about the some propositions, as opposed to *base-level facts* which pertain to a some “base” domain, e.g., about hydraulics.

See the *base-level fact* entry.

metaphor (185)

Refers to a linguistic phenomenon. The distinction between metaphor and analogy is not relevant to this dissertation in general.

MI(S) (100)

Refers to the set of sentences which are “materially implied” by **S**.

See Definition 18 on page 100.

Monoid (33)

Refers to the monoid abstraction defined in the Algebra theory; see SubAppendix B.4.3.

model (1) (63)

Used synonymously with schemata or paradigm, to refer to some structure which a top-down process can use to guide its search.

model (2) (84)

Here, model is used in the model theoretical sense; see [End72] and [Tar52].

more general formula (20)

The formula σ is *more general* than the formula ϕ if $\forall x.\sigma(x) \Rightarrow \phi(x)$. Hence, **Monoid** is more general than **Group**, as $\forall \vec{x}.\mathbf{Group}(\vec{x}) \Rightarrow \mathbf{Monoid}(\vec{x})$. (If the two formulae have different sets of formal variables, take the larger set. Hence **Field** is more general than **Group**.)

See also the *more specific formula* entry.

more specific formula (20)

The formula ϕ is *more specific* than the formula σ if $\forall \vec{x}.\sigma(\vec{x}) \Rightarrow \phi(\vec{x})$. This is the inverse of the *more general formula* relation; see that entry.

MRS (109)

Refers to the expert system building program on which *NLAG* was built.
See [Rus85].

mutatis mutandis (2)

Is Latin for "with changes as necessary".

NLAG (5)

The *NLAG* system is my implementation in the ideas presented in this thesis. In particular, it follows the $\|_{PT}$ model of analogical inference (presented in Figure 4-5 on page 57), augmented with the ordering and pruning rules discussed in Chapter 5. Chapter 6 presents further details of its structure.

Ohms Law (8)

Originally referred only to the electricity domain fact that the voltage drop across a device (e.g., a resistor) is the product of the current entering that device times its resistance. Subsequently generalized to deal with general "through", "cross" and "resistance" terms and "load bearing" elements.
See **RKK** entry.

partial interpretation (167)

A partial interpretation differ from a total (read "Tarskian") interpretation in that a partial interpretation need not completely specify the extensions of its symbols. See Section 8.2.

perspective (48)

A perspective is a collection of facts; in particular, it is a subset of the facts in an theory which all pertain to some common notion. E.g., the group axioms form one *perspective* of number theory.

prima facie (202)

A fact is considered *prima facie* if it is stored explicitly in the knowledge base. Equivalently, each such fact is *primitively stored*; i.e., it would be returned by

LOOKUP.

See the *primitively stored* and *LOOKUP(s)* entries.

primitively stored (111)

The *LOOKUP* subroutine returns just primitively stored information. (By contrast, other implicit information can be found only after some inferences are performed; the *TRUEP* subroutine returns these facts as well.)

See the *LOOKUP(S)* and *prima facie* entries.

problem statement (39)

Refers to some specific query; that is, a proposition to be *TRUEP*ed.

See the *target problem* and *TRUEP(s)* entries.

proposition (8)

A proposition is a syntactically well formed string of symbols. Propositions present in the knowledge base are called "assertions" or "facts". "Conjectures" are propositions which have been proposed for inclusion in the knowledge base. (By assumption, each conjecture is independent of the knowledge base.)

See the *fact* and *conjecture* entries.

PT (39)

Refers to the *target problem* of a useful analogy.

See the *target problem* entry.

PS (75)

Refers to the *source problem* associated with a useful analogy.

See the *source problem* entry.

 \mathfrak{R} (80)

Refers to the set of all real numbers.

 \mathfrak{R}_0 (A-8)

Refers to the set of all non-zero real numbers.

range (171)

Refers to the destination (or target) of a mapping.
See the *domain* entry.

RCKK (56)

Refers to the Resistance and Capacitor abstraction defined in the LELS theory; see SubAppendix B.4.4.

Relation(ϕ) (56)

Refers to the relation symbol used in the atomic formula ϕ .

representation (187)

Used synonymously with “theory”, and refers to the collection of information which describe some concepts.
(This differs from “formalism”.)

Ring (56)

Refers to the ring abstraction defined in the Algebra theory; see SubAppendix B.4.3.

RingId (152)

Refers to the ring with identity abstraction defined in the Algebra theory; see SubAppendix B.4.3.

RKK (51)

Refers to the Resistance abstraction defined in the LELS theory; see SubAppendix B.2.1.
See also Figure 4-3 on page 52.

RKK-EC (48)

Refers to the Resistance perspective of the electricity domain.
See Figure 4-2 on page 50.

RKK_{CC} (79)

Refers to the elaborated **RKK** abstraction, embellished to accept only those instantiations which come from a common context.
See Equation 5.7 on page 79.

RKK-Exp (A-27)

Refers to a modification of the **RKK** abstraction, one which begins to model non-laminar flow.

Equation A.12 on page A-27 defines the modified version of Ohm's Law.

RKK-Junc (A-27)

Refers to the elaborated **RKK**-abstraction, embellished to explicate the junctions.

See Equation A-3 on page A-28.

RLCKK (153)

Refers to the Resistance, Inductance and Capacitor abstraction defined in the LELS theory; see SubAppendix B.4.4.

RLKK (153)

Refers to the Resistance and Inductance abstraction defined in the LELS theory; see SubAppendix B.4.4.

residue (60)

Refers to the additional conjectures which must be added to a theory to permit a desired proposition to be proven.

See the Δ entry.

 \mathcal{RW} (172)

A particular privileged partial interpretation used by the partial interpretation semantics framework; as in $Allowed(\mathcal{RW}, Th)$.

See the *Allowed* entry.

SameTheory($a_1 \dots a_m$) (79)

Refers to the relation which holds when all of its arguments, $\{a_i\}_i$, are associated with the same theory.

InTheory(c, th) (127)

Refers to the relation which holds when the concept c is associated with the theory th .

satisfy (20)

We say a theory, Σ , *satisfies* a formula, ϕ , if there is some instantiation of ϕ which is not inconsistent with Σ ; i.e., if $\exists \{x_i\} \Sigma \not\models \neg\phi(x_1, \dots, x_m)$.

second-order fact (20)

Refers to facts whose arguments pertain to function or relations. Hence, `Kirchoff1(FlowRate)` is a second order fact as the argument, `FlowRate`, is itself a function.

See the *first-order fact* entry.

sentences (13)

Refers to syntactically well-formed formulae with no free variables. (Used synonymously with *proposition*.)

See the *formula* and *proposition* entries.

s-expression (131)

Refers to the data structures used in LISP. See [MAE*62] and [Rus85].

Skolem variable (15)

Refers to some term which is dependent on another, in the sense that `VoltageDrop` is dependent on `Current`, with respect to the **RKK** abstraction. In our system, reinstantiating an n -ary abstraction for the target analogue requires determining values for the other $n - 1$ *Skolem variables*.

Solve (132)

Refers to the subroutine which uses *constraints* to solve some query.

See the *constraints* entry.

source analogue (13)

Refers to the source concept of the analogy. By convention, it is represented by the symbol **B**.

See the *target analogue* entry.

source analogy sentence (17)

Refers to the instantiation of the analogy formula involving the source analogue;

usually written $\varphi(a_1, \dots, A, \dots, a_n)$.

See the *analogy formula* and *source analogue* entries.

source instantiation (of the analogy formula) (17)

Refers to the terms used in the particular instantiation of the analogy formula which involves the source analogue; i.e., it is the $[a_1, \dots, a_n]$ of $\varphi(a_1, \dots, A, \dots, a_n)$.

See the *source analogy sentence* entry.

source of the analogy (17)

Refers to the source analogy sentence.

See the *source analogy sentence* entry.

source problem (77)

Refers to a problem which deals with the source analogue which resembles the target problem (which deals with the target analogue). In current implement, it is generated by the H_{JK} heuristic.

See the H_{JK} entry.

source theory (8)

Refers to the collection of facts associated with the source concept; it is a subset of the initial theory. E.g., EC is the target theory associated with the target analogue Current.

See the *EC* entry.

subroutine (108)

Refers to a body of LISP code. The term "function" is used to refer to mathematic entities (which, of course, may be realized by some subroutine). *Subroutine* is synonymous with *procedure* and *module*.

$ST_{Th}(\sigma)$ (31)

Refers to the *support set* of the sentence σ from the theory *Th*.

See Definition 8 on page 31.

See also the *support set* entry.

support set (31)

Refers to the minimal subset of a theory which are sufficient to prove a sentence.

Used synonymously with *justification*.

See the *justification* entry.

T (167)

Refers to "the true".

target analogue (13)

Refers to the target concept of the analogy. By convention, it is represented by the symbol A.

See the *source analogue* entry.

target analogy sentence (17)

Refers to the instantiation of the analogy formula involving the target analogue; usually written $\varphi(b_1, \dots, B, \dots, b_n)$.

See the *analogy formula* and *target analogue* entries.

target instantiation (of the analogy formula) (17)

Refers to the terms used in the particular instantiation of the analogy formula which involves the target analogue; i.e., it is the $[b_1, \dots, b_n]$ of $\varphi(b_1, \dots, B, \dots, b_n)$.

See the *target analogy sentence* entry.

target of the analogy (17)

Refers to the target analogy sentence.

See the *target analogy sentence* entry.

target problem (39)

Refers to a problem which deals with the target analogue. In the context of the useful analogical inference process, the "goal of the analogy" is the addition of the new facts needed to return an answer to this query.

See the *PT* entry.

target theory (8)

Refers to the collection of facts associated with the target concept; it is a subset

of the initial theory. *E.g.*, FS is the target theory associated with the target analogue FlowRate.

See the *FS* entry.

Tarskian Interpretation (185)

Refers to the standard notion of interpretation, as contrasted with partial interpretations.

See the *Partial Interpretation* entry.

task (112)

Refers to a (sub)process which must be executed. Within the MRS system, tasks are usually placed on an agenda, and pulled off and run in an order determined by various meta-rules. In general, a task can be an arbitrary chore, which might be performed, *e.g.*, by the *NLAG* system or its writer.

term (A-11)

Refers to a syntactic expression which evaluates to some concept.

TermMap(*f*) (A-11)

Holds when *f* is a function which maps terms to terms.

Th (13)

Refers to the initial theory used in an analogy situation.

See the *theory* entry.

Th' (39)

Refers to a final theory, the result of an analogical inference.

See the *Th* entry.

Th_{CF} (17)

The particular initial theory used for our FlowRate~Current analogy situation.

See the *theory* entry.

theory (13)

Refers to a deductively closed, consistent collection of axioms. Used synonymously with "Knowledge Base".

TheoryOf (79)

The (MRS) function which maps each symbol into its "defining" theory. Used by H_{CC} .

See Heuristic 3 on page 79.

Trivial (26)

Defines when a formula is trivial, with respect to some theory. See Section 2.4.

Trivial_T (A-11)

Defines when a set of sentences is trivial with respect to some constant.

See the *Trivial_F* entry.

TRUEP(s) (112)

The MRS subroutine which determines whether a proposition is provable from the current theory.

W (167)

Refers to a possible world, especially in Chapter 5.

W[σ] (86)

Refers to the set of possible worlds in which $\Sigma + \sigma$ are true, for some defaulted theory Σ .

W_{Th}[σ] (90)

Refers to the set of possible worlds in which $Th + \sigma$ are true.

world (73)

Used synonymously with model (in second sense).

See the *model (2)* entry.

U (167)

Refers to the universe of discourse.

underlying theory (172)

Refers to the *Th* theory used in $Allowed(\mathcal{RW}, Th)$.

See the *Allowed* entry.

Unknown (13)

Refers to the condition imposed by all of the analogical inference processes which prevents the process from conjecturing any fact already deducible from the initial theory.

See Figure 2-1 on page 16.

Useful (39)

Refers to the condition imposed by the useful analogical inference process which insists that the new conjecture leads to an answer to the given target problem.

See Figure 3-1 on page 39.

useful analogy (40)

An analogy is considered *useful* if it is likely to suggest useful new conjectures,; and in particular, when those suggested conjectures help to solve the specific given problem.

useful fact (40)

In general, a fact is considered *useful* if it helps to solve standard problems.

Verify (111)

Refers to the *NLAG* module which verifies that the new conjectures are consistent with each other and the initial theory, are useful (*i.e.*, lead to an answer to the given problem) and are acceptable to the user.

See Section 6.1.1.

Z (A-9)

Refers to the class of integers.