The Automated Discovery of Universal Theories

by

Kevin T. Kelly

December 1986

Report CMU-PHIL-2

Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

THE AUTOMATED DISCOVERY OF UNIVERSAL THEORIES

by

Kevin T. Kelly

M.A., University of Pittsburgh, 1982

Submitted to the Graduate Faculty of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

1986

THE AUTOMATED DISCOVERY OF UNIVERSAL THEORIES

Kevin T. Kelly

University of Pittsburgh

This thesis examines the prospects for mechanical procedures that can identify true, complete, universal, first-order logical theories on the basis of a complete enumeration of true atomic sentences. A sense of identification is defined that is more general than those which are usually studied in the learning theoretic and inductive inference literature. Some identification algorithms based on confirmation relations familiar in the philosophy of science are presented. Each of these algorithms is shown to identify all purely universal theories without function symbols. It is demonstrated that no procedure can solve this universal theory inference problem in the more usual senses of identification. The question of efficiency for theory inference systems is addressed, and some definitions of limiting complexity are examined. It is shown that several aspects of obvious strategies for solving the universal theory inference problem are NP-hard. Finally, some non-worst case heuristic search strategies are examined in light of these NP-completeness results. These strategies are based upon an isomorphism between clausal entailments of a certain class and partition lattices, and are applicable to the improvement of earlier work on language acquisition and logical inductive inference.

Table of Contents

List of Figures

Foreword

This document is an attempt to integrate work on discovery methods from such diverse fields as philosophy of science, artificial intelligence, and recursion theory. I hope it partakes of the virtues of each of these disciplines: the motivational clarity of philosophy, the powerful generality of recursion theory, and the practical appeal of artificial intelligence. But there is no doubt that it partakes of their vices: the tedious argumentative hammering of philosophy, the bewildering notation and arcane results of recursion theory, and the myriad unmotivated choices made in artificial intelligence programming. Although the combination of these studies introduces so many distinct faults, I hope that their admixture ameliorates them, and that the the tedium of algorithm design, philosophical argument, and mathematical proof is vindicated by a new perspective on the scopes and limits of automatic discovery procedures.

For each thesis there is some person other than the author who is more than anyone else crucial for its completion. In my case, this person is Clark Glymour. Glymour's commitment of time and energy to his graduate students is unusual. But more important to me was is his willingness to immerse himself in my project and to extend it as it progressed. His careful attention contributed materially to the thesis, but it also did much to revive my enthusiasm for the topic when prospects seemed dim. In fact, his only unreasonable habit was to refuse to listen to mathematical proofs on the phone at night while washing his baby.

Ken Manders and Robert Daley were very helpful. Manders complemented Glymour by listening to proofs on the telephone at night. I was often surprised by his patience and his facility to understand my vaguely expressed hunches during these very long conversations.

Robert Daley's knowledge of computation theory, inductive inference, and computational complexity was also quite useful. Special thanks are due him for holding two consecutive seminars on the mathematical theory of inductive inference. These seminars provided a lucky chance for me to survey and to understand some of this new and difficult literature.

I would like to thank the balance of the committee, including Ken Schaffner, Richmond Thomason, and Jaime Carbonell, for their supportive attitude, their cooperation in arranging meetings promptly, and their interest in the topic.

I have also been assisted through conversations with Keith Wright, Richard Statman,

Teddy Seidenfeld, Wilfried Sieg, Tim Maudlin, and Richard Scheines. Richard Scheines and Martha Harty deserve thanks for providing about eighty percent of my social life for the last six years. I am sure it was not easy, but somebody had to do it.

Carnegie-Mellon University deserves credit for providing a reasonable research environment for the completion of this project. In particular, I would never have finished without the powerful and reliable computer network on campus.

Finally, I wish to thank the members of my family for their warmth and support, despite the infrequency of my calls and visits.

Introduction

A standard, philosophical approach to the theory of inductive inference is to define relations of confirmation, explanation, refutation, or evidential support that hold between evidence and hypotheses. Then *given* some evidence and an hypothesis, one can tell whether the evidence supports or refutes the hypothesis. Never mind where the hypothesis comes from. All that matters is that it bear a certain relationship to the evidence. Or to put it another way, there is no *logic* for discovering good hypotheses. The methodologist's task is merely to *evaluate* whatever conjectures people dream up.

It seems to me that this approach ignores much of what is interesting about inquiry. The goal of inquiry is the *generation* of knowledge. So there is a serious gap between a theory of hypothesis evaluation and a theory of inquiry. This gap between defining what a good hypothesis is and providing a full theory of inquiry is like the gap between knowing what a needle is and finding one in a haystack. In both cases, the latter concern is not trivial.

Moreover, it is false that there is nothing a methodologist could or should say about hypothesis generation methods. If one method is known to arrive at a suitable hypothesis in every world another one can--- and then some--- then *ceteris paribus*, the former method is better than the latter. Or if one can determine that a method is much more costly to use than another, but does not provide any better results, then the former method is worse than the latter one. And in either case, choosing an inferior method over a better one is irrational.

The aim of this thesis is not to invent the One True Method sufficient for all inquiry. There is no such thing. Nor is it to provide a normative theory in which any two methods are comparable. I don't expect to find one of these, either. But there are interesting methods that are not universally applicable, and there are norms that permit us to say that *some* methods are better than others. The purpose of this thesis is to explore some of these more limited norms and methods. Once we lower our expectations from the outlandish to the possible, we will see that there is a good deal to be said about the logic of discovery.

It would be unfair to sell a model kit whose "instructions" say only that one should "make the thing look like the picture on the box". What is required is an unambiguous, step-by-step *procedure* for turning the parts of the kit into the desired result. The same is true of discovery methods. They should specify unambiguous procedures for constructing hypotheses on the basis of input evidence.

Since hypothesis generation methods are procedures, they should be studied and evaluated as such. The science of procedures is computation theory. Outside the context of computation theory, there seems little to say about logics of discovery. No doubt, this is the motivation behind the usual epistemological disdain for the study. But from a computational point of view, there is far more to say than I am able to say in this thesis. The important questions about discovery become as pervasive, precise, and compelling as those in any other branch of philosophy.

But while computation theory is central to the logic of discovery, other disciplines are also important. Related topics include mathematical logic, model theory, probability and statistics, combinatorics, lattice theory, computational linguistics, the philosophical theories of confirmation and explanation, and the traditional studies of metaphysics and epistemology.

Overview

The traditional, philosophical consensus is that there is no logic of discovery and that even if there were, it is of no epistemological interest. In chapter one, I clear the ground by exposing these claims as the groundless dogmas that they are. I diagnose the weakness of the position as resulting from a failure to recognize that procedures must be evaluated from a computational point of view.

While philosophers were preoccupied with the pointlessness and impossibility of the logic of discovery, mathematicians, statisticians, computer scientists, and linguists were *doing* it. In chapter two, I focus on how this literature evaluates the methods it proposes. *Ceteris Paribus*, methods are better insofar as they are more general and less costly to compute. And a method is more general insofar as it is able to converge to an adequate hypothesis in a wider range of possible worlds.

In the subsequent chapter, I shift the focus from criteria for evaluating hypothesis generators to the generators themselves. In particular, we examine generation algorithms proposed by Pao, Angluin, Horning and Shapiro. Some patterns emerge. The object in designing a general hypothesis generator is to eliminate the test of hypotheses whose consideration adds nothing to the system's generality. But it is

not enough to *decide* to withhold an hypothesis from test, for just considering whether to test each hypothesis can take too long. Somehow, hypotheses that do not contribute to the system's generality must be ignored altogether. Particular algorithmic techniques for withholding tests and ignoring hypotheses altogether are examined in detail.

The first three chapters review the relevant literature and lay the general groundwork for the thesis. The subsequent four chapters attempt to achieve greater depth by narrowing the focus to a particular problem. The problem focused upon is a novel one: that of converging to a theory that entails every true universal sentence true in a given world on the basis of true evidence about particular individuals in this world. A generator is viewed as having converged to a world if for each true, universal sentence, there is some time after which all the device's conjectures entail it, and for each false universal sentence, there is a time after which no conjecture entails it. This is different from the usual sense of convergence, in which there must be some unique point after which all and only the true universal sentences are entailed by each conjecture.

In chapter four, the problem just described is introduced and its motivation is compared to that of similar problems. Then three solutions to the problem are proposed, each of which is based on a particular, syntactic theory of confirmation. One method, called HEMP, relies on C.G. Hempel's syntactic confirmation relation to solve the problem. Another solution relies on a generalization of Jean Nicod's instance confirmation condition. Finally, a third solution relies simply on the relation of consistency with the evidence. Each of these confirmation relations can be viewed pragmatically, as a cog in a general, inductive procedure.

In chapter five, I take up the question of what it is for an hypothesis generator to be efficient. The concept turns out not to be so simple as one might expect. I begin with a review of standard complexity theory. Then I examine some proposed approaches to the complexity of inference, and then discuss their merits and demerits. Next, I propose some theories of efficiency for solutions to the universal theory inference problem proposed in the previous chapter, and again discuss their formal merits and demerits. In the end, I must settle with something less than a fully general theory. Finally, the chapter addresses some crucial differences between the difficulty of generation problems and the difficulty of their corresponding decision problems. I propose an account of the difficulty of the former that explicates intuitions about good and bad solutions to problems that require the generation of very large sets.

The methods proposed in chapter four make no claims to efficiency. In chapters six and seven I undertake to shave the fat off of them. In chapter six I focus on techniques to avoid the consideration of equivalent variants of the same proposition, and in chapter seven I consider methods that rely on the entailment structure of the hypothesis language to avoid the consideration of useless hypotheses. The outcome of these two chapters is mixed. There are some easy ways to greatly improve inductive efficiency. But what is more surprising is that there are some easily characterized hypotheses that just can't be ignored without expending more effort to expunge them than they would absorb by sitting around and getting in the way.

So what do we learn from all this? First of all, the logic of discovery is not trivial. Nor need it be an assemblage of vague truisms. Nor need it be the study of human psychology. It its proper, computational element, it is as subtle and interesting as any other philosophical subject. Indeed, the logic of discovery is so rich with interesting, computational questions that this thesis should be viewed as no more than a minor scratch on its surface.

Chapter 1
Why No Logic of Discovery?

The success of logic and set theory in providing a more unified view of mathematical method encouraged the logical empiricists to apply the same formal approach to the understanding of empirical methods. This led to the many well-known probabilistic and logical accounts of justification, corroboration, and explanation. The logical empiricists were amply aware of the long tradition of philosophical attempts, beginning at least with Plato and Aristotle, to provide normative principles for *producing* reasonable beliefs from given evidence. They were also aware that their own accounts of verification, confirmation, and explanation fail to provide such constructive principles. Their response was a vigorous attack on the very philosophical *interest* of the "logic of discovery": the study of procedures guaranteed (or at least likely) to produce hypotheses that are reasonable with respect to given evidence.

The last five years have seen a revival of interest in the logic of discovery among philosophers of science [Nickles80], and there are still philosophers ready to raise the traditional anti-generationist arguments against this new interest. [Laudan80].[1] In particular, they pose two familiar objections to the philosophical study of hypothesis generating methods:

- The logic of discovery has no epistemological relevance.
- Even if it did, there are no adequate logics of discovery.

This chapter disputes the arguments for each of these claims. The reasons offered for the first thesis are either uncompelling or dependent on equivocations and faulty analogies. There are no arguments offered for the second thesis, only claims for it. I show that if the thesis is given a precise sense it is either obviously false or extremely difficult to prove. Indeed, the very accounts of confirmation offered by

---

[1] "Generationism" denotes the view that the logic of discovery, the enterprise of seeking procedures that reliably generate interesting, useful, explanatory, or confirmed hypotheses on the basis of given evidence, is philosophically relevant and interesting. Generationism is distinct from "inductivism", the view (opposed by Popper) that singular evidence can provide a reason to believe a general claim. Carnap, for example, was an inductivist but not a generationist. Finally, anti-generationism is the position that the the thesis of generationism is false.

the logical empiricists can be used to produce adequate logics of discovery, in their own sense of "adequacy".

The real moral of this essay is not, however, that every version of anti-generationism is untenable. It is rather that the philosophical investigation of the interest and scope of the logic of discovery raises substantive, computational questions which cannot even be formulated sensibly in abstraction from a precise, computation-theoretic setting.

1.1. On The Arguments that the Logic of Discovery is Philosophically Irrelevant

The contemporary popularity of the first objection is due in large part to its appearance in the section entitled "Elimination of Psychologism" of Popper's The Logic of Scientific Discovery. Popper argues in this passage that the title of his book does not denote.

> ...[T]he act of conceiving or inventing a theory seems to me neither to call for logical analysis nor to be susceptible of it. The question how it happens that a new idea occurs to a man---whether it is a musical theme, a dramatic conflict, or a scientific theory---may be of great interest to empirical psychology; but it is irrelevant to the logical analysis of scientific knowledge (Popper68); p. 31.

Popper's rationale for the irrelevance of the manner in which a theory is "conceived or invented" to "the logic of analysis of scientific knowledge" is that the latter study concerns *justification*, while the former study is concerned with actual causes in human cognitive processes.

> [The logic of analysis of scientific knowledge] is concerned not with *questions of fact* (Kant's *quid facti?*), but only with questions of *justification or validity* (Kant's *quid juris?*) (p. 31).

But it is simply false that the logic of discovery is restricted to the study of actual, causal processes underlying actual, human behavior. First, it is not confined to the study of *actual*, causal processes. Given a programming system, the hypothesis generation procedures specifiable in that system exist abstractly in the same sense that proofs in a given formal system exist. So the logic of discovery is an abstract study whose domain includes all *possible* procedures.

Second, the logic of discovery is concerned with the investigation of adequate hypothesis generation procedures. What adequacy comes to is a *normative* question. Desiderata include general applicability, rapid convergence, efficiency, and an ability to generate simple, explanatory, confirmed hypotheses in the short run. So the logic of discovery is a normative, abstract study.

Psychology, sociology, history, archeology, and anthropology, on the other hand, investigate the actual causal principles regulating cognition, social interaction, and bodily functioning. These empirical sciences are not about arbitrary, abstract procedures; nor are they normative. Therefore, the logic of discovery is not a sub-discipline of any of these sciences any more than, say, proof theory is. Of course, a psychologist may employ proof theory himself, or he may conjecture that human behavior sometimes approximates the inferential standards of proof theory, but proof theory is then his methodological tool rather than his principal object of study. Similarly, a psychologist may conjecture that human behavior sometimes approximates a given logic of discovery, but the logic of discovery *per se* is not his object of study: actual human cognition is.

Larry Laudan [Laudan80] differs from most anti-generationists, for he actually argues for the irrelevance thesis. One argument is that studies analogous to the logic of discovery are of no philosophical interest, so the logic of discovery is of no interest either.

> ...[O]ne must ask what is specifically *philosophical* about studying the genesis of theories. Simply put, a theory is an artifact, fashioned perhaps by certain tools (e.g., implicit rules of 'search'). The investigation of the mode of manufacture of artifacts (whether clay pots, surgical scalpels, or vitamin pills) is not normally viewed as a philosophical activity (p.182).

But this analogy is not merely unsuccessful: it is self-defeating. The process of "quality control" for artifacts like clay pots, surgical scalpels, or vitamin pills is just as rarely viewed as a philosophical activity. But hypothesis generation is to artifact manufacture as hypothesis testing is to "quality control" for artifacts. Hence, the logic of justification must not be a philosophical pursuit.

If there is a difference between hypotheses and vitamin pills so that quality control is philosophically interesting in the first case but not in the second, then this difference may make a difference in the case of hypothesis generation as well. An obvious candidate for the difference that makes a difference is that hypotheses are the raw material for *knowledge* and epistemology is, after all, the study of knowledge. Clay pots and vitamin pills, on the other hand, are not candidates for knowledge, so their manufacture *and quality control* do not constitute epistemological concerns.

In a more specific analogy, Laudan claims that philosophers of law are uninterested in "the mechanics of drafting a piece of legislation" (p.182). The intended analogy is that the mechanics of drafting a piece of legislation is to its legal normative force

as the mechanics of generating an hypothesis is to the hypothesis' justification. Since philosophers of law do not study the mechanics of drafting (actual?) legislation, but do consider the normative force of law, epistemologists should ignore the generation of hypotheses and study only the normative (justificatory) aspects of knowledge.

Admittedly, the study of the actual process of legislation is analogous to the study of actual, human methods of hypothesis invention. For example, the fact that Benjamin Disraeli rarely came to Parliament during the debates over the Merchant Shipping Act of 1876 is of no obvious philosophical interest. But, once again, the logic of discovery is not confined to the study of *actual* human discovery processes. For Laudan's analogy to be fair, the philosopher of law should be asked to decide whether a method which generates laws guaranteed to possess some property in which the philosopher of law is interested would be interesting. But this is exactly the sort of issue at stake in discussions of Arrow's paradox and other social choice problems. It is, in a more remote form, the sort of issue at stake in Rawls' account of the role of the original position in the justification of social institutions [Rawls64], [Sen70], [Arrow51].

The arguments considered so far all assume a falsehood: that the logic of discovery is an empirical, descriptive science. Laudan also provides a more interesting argument that does not involve this false assumption (p. 182). I paraphrase the basic structure of his argument as follows:

- The logic of discovery is "redundant and gratuitous" to the task of finding "a sound warrant for our claims about the world".
- What other epistemological relevance could it have? (i.e. it has none).
- (Hence, the logic of discovery is "redundant and gratuitous" to epistemology in general).

Are the premises credible?

1.1.1. Discovery and "Epistemic Warrant"

Laudan does not argue explicitly for the first thesis. He provides instead a discussion of the historical events that led to the abandonment of the logic of discovery in the Nineteenth Century. According to Laudan, epistemologists have tended to divide into two camps.

> ...[T]he *consequentialists*...believed that theories or claims could be justified by comparing (a subset of) their consequences with observation.

> If an appropriatedly selected range of consequences proved to be true, this was thought to provide an epistemic justification for asserting the truth of the theory.
>
> ...[T]he *generators*... believed that theories could be established only by showing that they followed logically (using certain allegedly truth-preserving algorithms) from statements which were directly gleaned from observation (p. 176).

The logic of generation flourished prior to the Nineteenth Century because everyone knew that hypothetico-deductivism is fallible, but the question of infallibility for generation procedures was open.

> ...[I]f one seeks infallible knowledge and if one grants the fallaciousness of affirming the consequent, then the only viable hope for a logic of justification will reside in the quest for a truth-preserving logic of discovery (p. 178).

But in the Nineteenth Century, complicated theories about unobservable objects were common, infallibilism was rejected, and hypothetico-deductivism flourished (p. 178). So the generationist program was abandoned because hypothetico-deductive tests seemed more easily applicable to theories postulating unobservable structures, and the demise of infallibilism undercut the motivation for seeking a generation method. Laudan derives a philosophical lesson from this historical discussion.

> The program for articulating an infallible logic of discovery never came to fruition; but that failure only partially explains its abandonment. Equally crucial here was the joint emergence of epistemic fallibilism and of *post hoc* logics of theory testing; developments which rendered redundant and gratuitous the logic of discovery so far as the epistemological issue is concerned. It remains redundant now (p. 182).

This passage seems to suggest that the logic of discovery is gratuitous because the hypothetico-deductive method of Herschel, Compte and Whewell represents the completion of epistemological inquiry, so the study of different methods of *any* sort is gratuitous. But this position is implausible in light of the many standard formal objections to hypothetico-deductive method [Glymour80].

So perhaps Laudan only intended that the logic of discovery is gratuitous to the epistemological *search* for an adequate theory of epistemic warrant that has not yet been found. Laudan describes the epistemological issue at stake as the problem of "how to provide a sound warrant for our claims about the world" (p. 176). This could mean that the task of the epistemologist is to provide a way to show that our claims about the world are warranted by our evidence. Or perhaps the epistemologist need only specify abstract conditions under which an hypothesis is

warranted by evidence, whether or not we know how to find out that it is. In the latter case, the proper object of epistemological study is an *abstract relation* of epistemic warrant (confirmation) between theory and evidence. For any given theory and any given evidence, this relation either holds or fails to hold whether or not anyone knows that it does. But in the former case, the proper objects of epistemological study are *procedures* for establishing that the abstract relation of warrant holds in a particular case.[2]

Laudan's consistent use of the phrase "*post hoc* logics of theory testing" strongly suggests that he is interested in test procedures rather than in abstract relations. Laudan admits that some discovery methods may be guaranteed to generate only warranted hypotheses. Any such procedure establishes the warrant of the hypotheses generated and is therefore of relevance to epistemology. Nevertheless, Laudan maintains that the logic of discovery is gratuitous to epistemological inquiry because *post hoc* test procedures *suffice* to provide epistemic access to the relation of epistemic warrant. But the mere fact that the study of test procedures would suffice does not imply that the study of discovery procedures is gratuitous. *Reductio* proofs suffice to derive all derivable theorems, but it would be silly to infer from this redundancy that direct proofs are gratuitous for purposes of demonstration. They are no more gratuitous than *reductio* proofs are.

Laudan's position may be more sophisticated. Perhaps he assumes tacitly that each adequate generation method consists of a test subroutine together with a procedure that enumerates hypotheses and applies the test to them. Since the test procedure already provides epistemic access to the underlying relation of epistemic warrant, any further work required to convert the test into a generation procedure is gratuitous to the practice of epistemology.

But although generation procedures *can* be written in this obvious manner, they *need not* be. Indeed, there are abstract relations whose obvious test and generation procedures are related in just the opposite manner: the test makes use of the generator, but the generator makes no use of the test. For a trivial example, let the relation $\Phi(x,y)$ be defined as $y=x^2$. The obvious generator is just a device that when given x calculates x^2. Clearly, this calculation need not effectively enumerate the natural numbers and compute a test $\Phi(x,y)$ for each y in the enumeration until the test says 'yes'. It can be computed quickly and efficiently, as by a pocket calculator. On the other hand, the obvious test for $\Phi(x,y)$ calculates x^2 when given x

---

[2] My sense of "procedure" is broad enough to include any formal system in which it can be derived that the relation holds in a given case.

and then checks to see whether the result of this calculation is identical with y. If so it says 'yes'. Otherwise it says 'no'. This test procedure uses the generation procedure as a subroutine.[3] Since generation procedures need not rely in an obvious way on explicit tests for epistemic warrant, there is no "extra" work involved in the logic of discovery that is gratuitous to the pursuit of epistemology.

My attack on Laudan's position exploits the computational symmetry between test procedures and generation procedures. Just as Grue can be defined in terms of Green and Green can be defined in terms of Grue, a generator can be built out of a test procedure, but a test procedure can *also* be built out of a generator. When the one sort of procedure squeezes through the door, the other is difficult to exclude. So someone defending Laudan's thesis might retreat to the ascetic position that the object of epistemological inquiry is just the abstract relation of epistemic warrant *simpliciter*. On this view, *any* procedural considerations are rejected as "merely pragmatic". So the logic of *post hoc* tests is as gratuitous to the pursuit of epistemology as the logic of discovery is.

But there is something unsettling about philosophical accounts of rationality that are aimed at "ideal agents" who are not subject to "merely pragmatic" computational limitations. After all, these accounts are clearly motivated by an attempt to show that agents whose evidence is somehow limited can nonetheless be justified in believing theories that greatly extend this evidence, even when these theories are false. If the limitations of the agents subject to these norms were not of central importance to their formulation, then epistemology would be trivially complete upon the enunciation of the elegant principle "Thou shalt always believe all truths". The worry that fumbling, finite humans must measure miserably against such a standard is the starting-point for all epistemological theorizing. But notice that finite computational ability and limited evidence-gathering ability both arise from the same general source: the bare finitude of the epistemic agent. To make normative concessions to limited evidence gathering ability while making no concessions whatever to computational tractability lacks motivation. The situation is analogous to that of a priest, who in compassion for a lame man, derives by long and tedious arguments that it is acceptable for him not to walk to church--- so long as he walks to an equally distant one instead. One wonders whether the priest's exercise is worth the effort.

The absurdity of the ascetic position shows up in the philosophy of mathematics

---

[3] More generally, any underlying relation that contains a total function that is computable in polynomial time gives rise to such an example.

as well. According to this view, the important, underlying property of mathematical claims is mathematical truth, so this is all that concerns the epistemologist. Proofs are mere, pragmatic crutches for finding out whether this property holds for a given sentence. Worse, the set of proofs of a system is usually taken to be a recursive set of finite strings so that mere, finite beings can check them in finite time. This restriction is essential, for infinitary derivation systems can be shown to be more powerful. Hence, provability is of interest only to engineers and psychologists. Since the scope of first-order logic is delimited by the notion of finite, mechanically verifiable proofs, Goedel's completeness and incompleteness theorems are of no epistemological interest either. In fact, not much of anything that is difficult or interesting about mathematics is of interest on this view. So much the worse for the view.

1.1.2. Generation and Epistemology

The second stage of Laudan's argument is his demand to know what the epistemological relevance of the logic of discovery is if it is not to provide epistemic warrant for hypotheses. Some suggested answers follow, but an exhaustive survey is not attempted.

Much of Reichenbach's career was devoted to the "pragmatic vindication" of the "straight rule" of induction [Reichenbach49]. Laudan acknowledges that the straight rule is a logic of hypothesis generation. Reichenbach held on decision-theoretic grounds that the selection of the straight rule as an inductive method is justified, but he explicitly denied that belief in the hypotheses it produces is justified (pp. 373, 472). So even Reichenbach saw a crucial epistemic role for the logic of discovery that is not parasitic on the justification of individual hypotheses.

Once foundationalism is abandoned, new possibilities open up for the logic of discovery. Consider Quine's proposal for a naturalistic epistemology [Quine69]. The examination of the causal processes whereby the "meager input" of the human apparatus causally determines its "torrential output" is not only relevant to naturalistic epistemology; it *is* naturalistic epistemology (p. 83).

Perhaps no current epistemological theory is more vitally bound to the logic of discovery than is Larry Laudan's. His account of rationality can be summarized in two principles:

- "...[T]he choice of one tradition over its rivals is a progressive (and thus a rational) choice precisely to the extent that the chosen tradition is a better problem solver than its rivals" [Laudan77] p. 109.

- "...[I]t is always rational to pursue any research tradition which has a higher rate of progress than its rivals (even if the former has a lower problem-solving effectiveness)" (p. 111).

Laudan thoughtfully provides an explicit place for generation procedures in the concept of a research tradition.

> ...[A] research tradition is a set of general assumptions about the entities and processes in a domain of study, and about *the appropriate methods to be used for investigating the problems and constructing the theories in that domain* (p. 81; my emphasis).

It is not difficult to see the relevance of discovery issues to Laudan's theory of rationality. Imagine two traditions A and B, whose domains and metaphysical assumptions are identical. The only difference between A and B is that the A has bought sophisticated software guaranteed to generate only confirmed, explanatory hypotheses while B casts its lot with "creative intuition". The empirical problem-solving capacity of tradition A will accelerate until rationality compels both the pursuit and acceptance of A. Hence, discovery procedures and their properties can formally dictate the rational selection of a research tradition in Laudan's framework.[4]

Not only is the logic of discovery crucial in Laudan's framework; it reveals that his criterion of rational pursuit is not a necessary condition. For assume that the members of B, in a desperate attempt to catch up with A, purchase some "fifth generation" software that can be proved to converge to a correct result over the common problem domain ten orders of magnitude more quickly than A's software can. Once this fact is known, one seems rationally compelled to pursue B even though the actual rate of progress of B does surpass that of A until the new software is installed on the B-computer, and the B-technicians learn how to use it properly. One needn't wait until B actually surges ahead. The logic of discovery guarantees *a priori* that it will.

1.2. On the Arguments against the Existence of Any Adequate Hypothesis Generation Method

According to anti-generationists, the entire preceding defense of the philosophical interest of the logic of discovery was in vain, for there are no such methods to study. The significance of this opinion is difficult to assess, however, because the anti-generationists have only vaguely specified what would count as an hypothesis

---

[4] This scenario does not resemble the examples Laudan draws from the natural sciences, but it does reflect the situation in more methodologically self-conscious disciplines. For example, some "research traditions" in sociology insist on constructing statistical models by hand, while others apply computerized logics of discovery to the same sorts of problems with an enhanced rate of progress [Glymour84].

generator; much less have they specified what an adequate generator might be. Perhaps the clearest account is Carnap's, although it is not *very* clear:

> C1: The question whether an inductive logic with exact rules is at all possible is still controversial. But in one point the present opinions of most philosophers and scientists seem to agree, namely, that the inductive procedure is not, so to speak, a mechanical procedure prescribed by fixed rules. If, for instance, a report of observational results is given, and we want to find a hypothesis that is well confirmed and furnishes a good explanation for the events observed, then there is no set of fixed rules which would lead us automatically to the best hypothesis or even a good one. * * * The same point has sometimes been formulated by saying that it is not possible to construct an inductive machine. The latter is presumably meant as a mechanical contrivance that, when fed an observational report, would furnish a suitable hypothesis, just as a computing machine when supplied with two factors furnishes their product. I am completely in agreement that an inductive machine of *this* kind is not possible [Carnap50].

The phrases "fixed rules", "mechanical contrivance", and "computing machine" indicate that Carnap thinks of a method as a computer program. Notice that his claim that no adequate generation method is *possible* means that no such program exists as an abstract object. This *existential* thesis is much stronger than the mere contingent hypothesis that we will never find such a method, whether or not there is one.

Carnap requires that the outputs of an adequate hypothesis generator be confirmed and explanatory. The current lack of consensus over the nature of explanation and confirmation might lead to doubts that Carnap's existential claim can be adjudicated persuasively. But it is still possible to argue that Carnap's existential anti-generationism is false (or at least unattractive) no matter what relations the relations of explanation and confirmation are.

Accordingly, I call any relation involving hypotheses and any other relevant factors a "suitability relation". Suitability may involve only an hypothesis and an observation report. It may also involve human interests, historical context, the success of competing research programs, or other "external" factors. But since nothing is assumed about these factors, think of n-tuples of factors as single objects. I call each such sequence a "situation". So even though suitability will be treated as a binary relation, one of the relata is a situation coding all relevant factors.

The adequacy of an hypothesis generator depends upon more than a relation of suitability for hypotheses, however. Carnap seems to require that an adequate method generate a suitable hypothesis for *any* possible situation. But this necessary

condition for adequacy is too unsympathetic to generationism to be taken seriously, for there may be some situation for which no suitable hypothesis exists. In such a situation, a method should be praised rather than condemned for refusing to make a conjecture.

A more sensible criterion of adequacy is that the device commit no errors of omission or commission in any situation. M is guilty of an *error of commission* in situation W according to suitability relation S if and only if M outputs h upon receiving its appointed W as input and h does not bear S to W. M is guilty of an *error of omission* in S-situation W according to S if and only if M produces no output upon receiving its appointed sublist of W as input and there is an h such that h bears S to W.

> Hypothesis generation machine M is *strongly adequate* for suitability relation S if and only if for every S-situation W, M is guilty of no error in W according to S.

Since fewer algorithms satisfy a stronger criterion of adequacy, this extremely strict condition is very generous to Carnap's case.

1.2.1. Anti-Generationism and Recursively Enumerable Hypothesis Suitability

The following easy result shows that the existential anti-generationist thesis is false if suitability is a recursively enumerable relation.

> *Fact 1:* For any recursively enumerable suitability relation S, there is an hypothesis generation machine which is strongly adequate with respect to S.

The mathematical shallowness of Fact 1 belies its philosophical significance for anti-generationism. For example, Carnap's c^* confirmation measure is computable over the first-order monadic predicate calculus. For any chosen confidence level k, the test whether $c^*(h,e)$ exceeds k is clearly computable. Hence, Carnap's own construal of "h is well-confirmed on e" is not only recursively enumerable, but recursive as well. By the method of Fact 1, we can easily construct a strongly adequate hypothesis generation machine for this sense of confirmation. Hempel showed that his own "satisfaction" relation of confirmation is computable. By Fact 1, there is also a strongly adequate hypothesis generation machine for Hempel's confirmation relation.

Strongly adequate machines for these relations could generate trivial hypotheses, for tautologies are always highly confirmed. Carnap required, therefore, that the

outputs of an adequate discovery device be explanatory as well as highly confirmed. Hempel addressed this concern with his measure of "the systematic power of a theory" [Hempel65]. The measure s(h,e) assigns some rational number in the interval [0, 1] to any first order sentence h and any singular sentence e. Whether or not s(h,e) exceeds a certain level k is a recursively enumerable relation.

The intersection of any two recursively enumerable relations is itself recursively enumerable. Hence Fact 1 guarantees the existence of a strongly adequate hypothesis generation machine for each of the following recursively enumerable suitability relations, where H(h,e) is Hempel's recursive confirmation relation:

- S_1(h,e) if and only if c^*(h,e) > k and s(h,e) > k'
- S_2(h,e) if and only if H(h,e) and s(h,e) > k

If Carnap and Hempel believed what they were writing, then for any situation, these machines accept evidence and output a well-confirmed and explanatory hypothesis in this situation if and only if such an hypothesis exists. In a word, these two distinguished anti-generationists were either denying a simple mathematical truth, or they did not take the results of their own formal work on suitability relations seriously.

There is reason to think that the latter possibility applies in Hempel's case. Hempel worried a good deal about the significance and confirmation of hypotheses with non-logical vocabulary extending that of the evidence. Since his own confirmation relation was not intended to handle such cases, any discovery procedure adequate with respect to this relation would be inadequate with respect to the "true" suitability relation, which must deal with "theoretical" hypotheses as well as empirical laws.

> An adequate rule of induction would therefore have to provide, for this and for every other conceivable case, mechanically applicable criteria determining unambiguously, and without any reliance on the inventiveness or additional scientific knowledge of its user, all those new abstract concepts which need to be created for the formulation of the theory that will account for the given evidence. Clearly, this requirement cannot be satisfied by any set of rules, however ingeniously devised; there can be no general rules of induction in the above sense; the demand for them rests on a confusion of logical and psychological issues [Hempel65].

The interesting claim here is that no hypothesis generation machine can generate suitable hypotheses containing non-logical vocabulary not occurring in the input evidence. But what if we were to build a machine which is strongly adequate with respect to a proposed suitability relation for hypotheses with theoretical terms?

Carnap's theory of reduction sentences in "Testability and Meaning" [Carnap36] was intended as a theory of meaning, but it has been suggested [Glymour80] that it be viewed as an account of confirmation for theoretical hypotheses. The paper's leading idea is that the test of a theoretical hypothesis h on given evidence e is performed by testing this hypothesis with respect to sentences e' in the language of the hypothesis which are related to e in a particular way. To render the idea a particular proposal, one requires four things:

1. An hypothesis language L_h,
2. An evidence language L_e,
3. A specific account of confirmation which is adequate for hypotheses whose nonlogical vocabulary does not extend the evidential vocabulary.
4. A class R, which is a subset of $\{f:P(L_e) \dashrightarrow P(L_h)\}$

Without belaboring what could only be inadequate motivation, I choose the following ingredients:

1. L_e is the atomic sentences and negations of atomic sentences of a first-order relational language with a finite set $Pred_e$ of predicates.
2. L_h is the sentences of a full first-order relational language without function symbols or identity, with a finite set $Pred_h$ of predicates, such that $Pred_h$ properly includes $Pred_e$.
3. Our basis confirmation theory will be Hempel's, for simplicity. We shall write "e confirms h on Hempel's criterion" as simply "H(e, h)".
4. A *reduction pair* for L_h and L_e is the universal closure of L_h-formulae
   $O_1 \dashrightarrow (O_2 \dashrightarrow T)$
   $O_3 \dashrightarrow (O_4 \dashrightarrow -T)$
   where the O_i are atomic formulas whose predicates are in $Pred_e$ and the T_i are atomic formulas whose predicates are in $Pred_h - Pred_e$. For any finite consistent subset e of L_e and any finite, consistent set of such reduction pairs r, define
   $f(r, e) = f_r(e) = \{s \in L_h: e, r \vdash s\} \cup e$
   Then define

$$R = \{f_r: r \text{ is a finite, consistent set of reduction pairs}\}$$

Choice (4) just amounts to Carnap's scheme of reduction pairs in "Testability and Meaning". Now we define the confirmation relation CT as:

$$CT(r,e,h) =_{df} H(f_r(e),h)$$

for finite, consistent r, e, and L_h sentence h.

Notice that $\vdash$ is recursive for a prenex normal form language with strictly universal quantifier prefixes and no function symbols. Since e and r are finite, f is

a recursive function. Notice also that e, r can entail only finitely many atoms if e, r are consistent. Hence, f is recursive, so for any such r, f_r is as well. Since H is a recursive relation, it follows that CT is also a recursive relation.

Also, notice that the power set of the set of all reduction pairs for L_e and L_h is finite, and hence R.E. Of course, L_h is R.E. in virtue of being a language. This leads to the following program:

> Read the given finite, consistent subset e of L_e. Enumerate all possible pairs $\langle r,h\rangle_i$ and test $CT(f_r(e), h)$ for that pair. We can also recursively check whether some predicate in $Pred_h - Pred_e$ occurs in h. If both tests succeed, then output the current pair. Otherwise proceed to the next pair $\langle r',h'\rangle_{i+1}$.

This program is guaranteed to "invent" a confirmed hypothesis with "theoretical" predicates, if there is one. If explanatory hypotheses are desired, Hempel's theory of systematic power can be dovetailed in, where r is included as part of the hypothesis.

Nothing depends upon favored interpretations of the predicates in $Pred_h - Pred_e$. These predicates are mere objects pushed about by the agent executing the program. Hence, it would be hyperbole to claim that this machine is supplied "theoretical concepts" in advance. The predicates in $Pred_e - Pred_e$ cannot be said to correspond to "concepts" *until* a particular pair $\langle r, h\rangle$ is inferred by the machine or its user.[5] Of course, Hempel would dispute that Carnap had distilled the essence of confirmation in "Testability and Meaning" [Hempel65] p. 188. Moreover, he might have been skeptical that philosophers could ever do so. He certainly expressed such doubts in closely related areas.[6] But methodological skepticism about philosophical method is no evidence whatever for existential anti-generationism of the sort expressed by Hempel in the above quotation. If suitability is recursively enumerable (whether we currently describe suitability correctly or not) then the questioned machine exists.

---

[5] Even then, it is possible that where e" is the L_o diagram of one's favorite structure, e U {h} U r has many models which agree with the favored one on their assignments to $Pred_e$, but not on the theoretical predicates occurring in h and r. That is, these predicates need not be implicitly defined by e U {h} U r, when the program outputs $\langle r, h\rangle$ on input e.

[6] "I feel less confident, however, about the possibility of restating the general idea in the form of precise and general criteria which establish sharp dividing lines...."(Hempel 1965; p. 102)

1.2.2. Anti-Generationism and Unsolvable Hypothesis Suitability

The proofs of machine existence in the previous section fail to apply when suitability is not recursively enumerable, for then no program for enumerating the suitability relation can be assumed in the construction of an adequate hypothesis generator. Thus, an anti-generationist might claim that suitability is not a recursively enumerable relation. But this maneuver yields small comfort, for lots of "very uncomputable" suitability relations have adequate logics of discovery.

First, it can easily be shown that there is a non-recursively enumerable suitability relation for which there is a strongly adequate hypothesis generation machine. relation R is said to be *co-R.E.* if and only if the relation is the universal complement of some recursively enumerable relation. There exist sets which are co-R.E, but which are not recursive. Such sets are not recursively enumerable. Nevertheless, there are strongly adequate generation machines for some co-R.E., non-recursive suitability relations. Specifically,

> *Fact 2:* There exists a co-R.E. but not R.E. suitability relation for which there is a strongly adequate hypothesis generation machine.

Although the proof of Fact 2 relies upon an odd suitability relation, notice that first order non-entailment is a co-R.E., non-recursive relation. For Popper, it is a necessary condition for corroboration that an hypothesis entail no known false fact. Hence, if all the other necessary conditions on suitability a Popperian might impose are properly co-R.E., their conjunction is as well. Nonetheless, some entailment-decidable fragment of the hypothesis language might still contain a suitable hypothesis in any situation which has a suitable hypothesis in the expanded language.

Intuitively, Fact 2 reflects that even the program for a strongly adequate hypothesis generation machine for S does not generally presuppose a program to enumerate S. The following fact gives a necessary and sufficient condition for sifting a strongly adequate hypothesis generator from an arbitrary suitability relation S.

> Say that S is an *hypothesis-extension* of S' if and only if S' is a subset of S and for any S-situation W, if there is an h in H such that h bears S to W then there is an h' in H such that h' bears S' to W.

> *Fact 3:* There is a strongly adequate hypothesis generation machine for suitability relation S if and only if S is an hypothesis-extension of some recursively enumerable relation S'.

It is well known in the theory of computability that "unsolvability" does not stop with co-R.E. sets, but rather ascends in an infinite hierarchy of "degrees of unsolvability". If one begins with a recursive relation, successive projections and complementations of this relation yield relations of progressively greater unsolvability. The co-R.E. sets are merely one small step outside the bounds of any intuitive notion of computability. The anti-generationist might be tempted to take advantage of this fact and define suitability with many alternating negations and existential quantifiers. Nevertheless, it is easy to show:

> *Fact 4:* For any nonempty unsolvability degree, there are infinitely many suitability relations S properly of that degree for which there is a strongly adequate hypothesis generation machine.

Merely leaping up the unsolvability ordering will not guarantee the anti-generationist his quarry. He must still argue that his arbitrarily unsolvable suitability relation is not an hypothesis extension of some R.E. relation. A combinatorial argument for this proposition would be that suitability is both unsolvable and single valued (i.e. there is at most one hypothesis suitable in any S-situation). In this case, the relation could not possibly be an h-extension of an R.E. relation, so by Fact 4, no strongly adequate machine for this relation could exist. But such a proposal would entail at most one suitable hypothesis in any situation, which is implausible.

In short, the unsolvability degrees are teeming with suitability relations for which there are adequate logics of discovery. These elementary facts do not conclusively disprove existential anti-generationism, but they do show that this thesis is not the safe, "throw away line" it was thought to be by Popper, Hempel, and Carnap. To believe it without sophisticated reasons is to take a great risk. To assert it without argument is not to philosophize, but to speculate.

1.3. Conclusion

Popper's arguments that the logic of discovery is psychology ignore the fact that the former is an abstract, normative study. Laudan's arguments for the thesis that the logic of discovery is philosophically gratuitous neglect the computational symmetry of generation procedures and *post hoc* test procedures. Finally, Carnap's formulation of the position that there is no adequate logic of discovery can be shown by trivial recursion-theoretic constructions to contradict his own epistemological theories. He seems to have failed even to consider the mathematical audacity of his claim.

The common cause underlying each of these failures is a cavalier inattention to the mathematical, computational background of the logic of discovery. Any serious discussion of the interest and feasibility of possible discovery methods must consider detailed and difficult questions concerning their power and complexity. The study of computational complexity is quite young, but the question of the fundamental limitations on the feasibility of inductive inference has already been addressed in the computer science literature [Angluin78]. The study of the power of inductive methods was originated, in a sense, by Hilary Putnam [Putnam63] and has been greatly extended by computer scientists over the last two decades [Angluin82]. Even a cursory examination of the results of these computational inquiries reveals a complicated pattern of successes and failures for the logic of discovery. This complex pattern cannot be captured by the naive, traditional pronouncements of anti-generationism.

Chapter 2
Elements of the Logic of Discovery

2.1. Hypothesis Generation and Computation

The study of hypothesis generation methods has been a central concern of philosophical speculation since the dawn of recorded thought. The "paradox" recorded in the Meno, for example, asks how we could discover the true definition of a Form without already knowing what the definition is. The paradox can be restated in terms of heuristic search for an adequate hypothesis: how can any search procedure be sure to find an adequate hypothesis when the property of adequacy is undecidable on the basis of the inputs to the procedure? That is, how can the search be terminated without the risk of stopping too soon or too late?[7]

Aristotle was also interested in methods that generate adequate hypotheses on the basis of evidence. For example, he presented a formal method for generating hypotheses about natures in his Posterior Analytics. His proposal was to employ a partially specified syllogistic demonstration to constrain the search for a "middle term" that completes it. Bacon conceived of a sort of public industry of empirical knowledge generation, and Descartes proposed a private, introspective approach to the generation of knowledge. Later ages brought Newton's rules, Mill's method's, Reichenbach's straight rule of induction, and statistical estimation techniques. Every one of these proposals describes a process for generating adequate hypotheses.

An important difference between philosophical theories of inquiry is that some appeal to special, unanalyzed and unexplained mental faculties (e.g. Plato's "recollection" faculty and Descartes "natural light") while others are intended as explicit, logical procedures to be studied, learned, and followed in a self-conscious manner (e.g. Aristotle's logic, Bacon's Novum Organum, the logics of Newton, Mill and Reichenbach, and statistical point estimation techniques). Only proposals of the

---

[7] Newell and Simon distinguish "well-formed" from "ill-formed" search problems. A well-formed search problem consists of a search space and a procedure for deciding whether a given point in the space is a goal state or not. If no such decision procedure is available, then the problem is ill formed. So the Meno Paradox may be viewed as the claim that the search for an adequate hypothesis on the basis of evidence is an ill-formed problem.

latter sort deserve to be called "methods", for it should be possible to follow a method without reliance on mysterious abilities. For example, the Cartesian formula

Begin
    Invoke the Natural Light;
    Conceive Clearly and Distinctly;
    Assert the content of the resulting conception
End.

fails to be an hypothesis generation method, for it is difficult both to figure out how to execute it and to tell whether it has been executed correctly. And even if one believes that he can execute Descartes' formula at will, the ability to do so may be arcane and unavailable to others. Any method worthy of the name should analyze the problem to be solved into a multitude of trivial instructions that demand no special faculties for their execution.

This concern for explicitness and determinacy is also found in the standard motivations for the theory of computation [Rogers67]. The primitive abilities demanded of an agent to follow the instructions of a computer program are trivial, repeatable, and entirely non-mysterious. The fact that they can be simulated by actual machines is the supreme demonstration of their explicitness and determinacy. Accordingly, I restrict my attention to hypothesis generation methods that are computer programs or that can be reformulated as computer programs with only minor effort.

A computer language (LISP, PASCAL, Turing machines, random-access machines, C) contains infinitely many distinct programs, each of which computes some input-output function. Strictly speaking, every one of these programs can be described as an hypothesis generation method. Some generate nothing, others lead to the generation of silly or ill-formed hypotheses, and still others lead to the generation of sensible ones. Some methods require astronomical resources to produce an output, and others do so quickly. We are obviously not interested in all of these methods. Rather, we are interested in the *good* ones.

2.2. Inductive Generality

Consider an ordinary, electric toaster. Now imagine that Einstein's field equations are written on a piece of bread, which is inserted into one of the toaster's slots. The method is started by writing one's available astronomical evidence on another slice of bread, stuffing this slice into the toaster's free slot, and depressing the lever. After a minute and a half, the toaster-method is sure to "conjecture" a slightly darkened version of the field equations "on the basis of" the evidence provided.

There is nothing unclear about how to follow the toaster method for space-time physics. Moreover, we believe the conjectured hypothesis is true, explanatory, and downright ingenious. Finally, the toaster method is extremely fast--- much faster than poor Einstein was in generating the same hypothesis from the same data. So why don't scientists sit back and rely on toasters?

The toaster method is a farce, that's why. It makes the same conjecture no matter what the evidence is, and in most cases, its conjecture is not warranted by or even relevant to the sentences written on its "input" slice of bread. Needless to say, if the toaster method had been started in a world in which the field equations are false, it's conjecture would have been false. And had it been run repeatedly on increasing evidence true of that world, it would have conjectured falsehoods for eternity. Or to put it another way, no matter how much evidence we provide the toaster with, it will never come to discriminate between worlds in which the field equations hold and those in which they do not.

The toaster example suggests two considerations relevant to the assessment of discovery procedures. The first is that in ignoring its evidence, the device is unlikely to produce an hypothesis suitable for the evidence provided, where suitability is some relation like confirmation and explanation between evidence and hypothesis. So to recall Carnap's demand, a good device might be expected to produce confirmed, explanatory, hypotheses with respect to the evidence provided. *Ceteris paribus,* a method is better insofar as its conjectures are more usually suitable with respect to the evidence. And if we can measure suitability, a method is better, *ceteris paribus,* insofar as its conjectures are more usually more suitable.

The other consideration raised by the toaster example is that the toaster's outputs would have been forever false in any possible circumstances in which the Einstein field equations are false. Truth is not the only semantic standard of evaluation of hypotheses we might consider. We may demand informativeness as well as truth. Or we might lower our demands and permit false hypotheses to be adequate. Popper's notion of *verisimilitude* or "near truth" in a world is an example of a semantic standard of hypothesis evaluation that falls short of truth.

Recall that suitability relations are *epistemic* criteria of hypothesis evaluation in the sense that the investigator has access to all the factors relevant to determining the suitability of any given hypothesis (e.g. confirmation, explanatory power, simplicity, consistency with background beliefs). But truth, verisimilitude, and informativeness about a world are *semantic* relations rather than epistemic ones, in the sense that

they relate an hypothesis to a possible world, and the investigator usually has no direct way to tell which possible world he is in. To distinguish them from suitability relations, I refer to all semantic criteria of hypothesis evaluation as *adequacy* relations.

In light of this terminology, a shortcoming of the toaster method is that it does not eventually converge to an adequate hypothesis in a very wide range of worlds. There are many ways to define eventual convergence to an adequate hypothesis, and each such definition results in a distinct *identification* criterion. For example, a device is said to EX_*-identify a world w just in case for each enumeration of the total evidence true in w, the device changes its mind only finitely many times in reading this enumeration before it fixes upon an hypothesis adequate for w. This sort of convergence interested Peirce in the nineteenth century, and is taken quite seriously in contemporary discussions of scientific realism [McMullin84] and in the philosophy of mind [Dennett85]. Some users of inductive devices may not care about posthumous discoveries. For them, we can place a finite ceiling n on number of mind-changes before convergence to an adequate hypothesis to obtain what is called EX_n-identification. A device BC_*-identifies a world just in case for any ordering of the evidence true in a world, the device conjectures only finitely many hypotheses not adequate for the world. BC_*-identification suggests a corresponding notion of BC_n identification, which restricts the number of mind changes to no more than n. If adequacy is measurable (as in the case of verisimilitude), rather than being an all-or-nothing relation between worlds and hypotheses, then we can say that a method identifies a world w just in case for each enumeration of the evidence true in w, and for each ϵ, there is a δ such that after δ many evidence sentences have been read, every subsequent conjecture is adequate to degree $1-\epsilon$.

If we can assume either that the evidence is sampled stochastically from a world or that inductive methods are themselves stochastic processes, then a device identifies a world if its probability of conjecturing an hypothesis adequate for the world is unity. Or if the hypotheses conjectured by a device are parameterized by real numbers, one might require merely that the expected value of the method (considered as a random variable) is the parameter value of an adequate hypothesis.[8] Finally, we might examine concepts of identification in which the evidence provided to the procedure is not always true of the world to be identified. Such evidence is often called *noisy*. In short, there are many precise explications of the notion of eventual inductive success in a world.

---

[8] This is the basic idea behind the concept of "unbiased estimators" in statistical theory.

Once we select a notion of identification, we can define, for any given method, the set of all worlds this method can identify. Intuitively, this set of possible worlds represents the method's *range of success.* One might also call this set the method's inductive scope, generality, strength, or power. A method may be said to be *more general* than another if its inductive scope includes the other's. *Ceteris paribus,* a method is *better* insofar as it is more general.

If an investigator who professes to employ some method seems to have converged to an adequate hypothesis we can explain his apparent success to some degree by appeal to the inductive strength of his method. *Ceteris Paribus,* this explanation is better insofar as his method is more general. Indeed, if the investigator's method is very weak (e.g. he relies on a toaster-method) it is tempting to say that he is merely *lucky* when he succeeds, or that the discovery is an accident ([Aristotle41], The Physics).

One can explain why an investigator conjectures a given hypothesis without any appeal to inductive scope. For example, one can demonstrate merely that the investigator is caused by various social and physical factors to make the conjecture he does. But to explain why a particular system produces a particular conjecture is *not* to explain why the system seems to have converged to an adequate hypothesis--- any more than to explain why a friend buys a car (he needs transportation) is to explain why the car he buys is shaped like a banana. Inductive scope helps to fill this explanatory gap.

It is difficult to avoid appeals to probabilistic language in discussions of the explanation of inductive success. Such language is official if we assume that inductive scopes are measurable sets of possible worlds. Then the measure value of the scope of a method is its probability of eventual success. Notice that the set of worlds a method can identify is distinct from the set of worlds in which it converges to some hypothesis. The probability of convergence to a particular hypothesis may be very high even when the probability of success is very small (recall the case of the toaster). Regardless of the measure chosen, the probability of success of a less general method cannot exceed the probability of success of a more general one. But the probabilities of success of two methods, neither of which is more general than the other, is measure-dependent.

2.3. Resource Consumption

Inductive scope is not the only factor crucial to the assessment of an hypothesis generation method. For example, consider two methods that always produce the same conjecture on the same evidence, such that the latter requires geologically more time to produce its conjecture than the former does. Spending more time to do the same thing is not as good as spending less, so the faster method is the better one. A faster procedure also seems more *intelligent*, for intelligence is, in part, a kind of "quickness" or efficient organization of one's cognitive resources.

Also, if one could trace the sequence of computational states directed by the two programs, one would detect a kind of "elegance" or "intelligence" in the computation of the quick one that is lacking in the slow one. The slow one performs many tests that insight into the mathematical structure of the problem it solves would show to be unnecessary and short-sighted. So *ceteris paribus*, the faster, more elegant program is the better one.

To recapitulate: we prefer, all things being equal, methods that are more general, that are less costly to pursue, and that more frequently produce suitable hypotheses. But the situation is much more complicated when we lower the *ceteris paribus* restriction.

First of all, our background knowledge is sometimes so skimpy that no possible method's scope includes all the possibilities we take to be serious ones. In such an event, we must weigh scopes against one another even when neither includes the other.

We may also be forced, on mathematical grounds, to put suitability in the balance against inductive scope. Epistemologists like Pierce, Reichenbach and (at times) Putnam justify suitability relations in terms of inductive scope, so no such conflict can arise for them. That is, if the observance of a proposed relation of suitability compromises inductive scope, then the relation is just no relation of suitability. But a confirmation theorist might hold that confirmation is *sui generis* and need not be "vindicated" by considerations of inductive generality. For him, suitability can easily be at odds with inductive scope, so there are difficult choices to be made when no method that conjectures only suitable hypotheses is as general as some method that does not. For example, Osherson Stob and Weinstein have shown that there is an inductive scope that can be attained by some effective discovery method, but that cannot be attained by any effective discovery method that is also a Bayesian in the short run [Osherson86]. In this case, a choice must be made between greater inductive generality in the long run and Bayesianism in the short run.

Inductive generality may also interact with computational resource consumption in the assessment of discovery methods. We can expect, for example, that no method of a certain generality can be performed as quickly as some method of reduced generality. This does not imply that the faster program is more elegant or efficient than the slower one, for distinguishing among more possibilities on the basis of evidence may be expected to require more effort than distinguishing among fewer possibilities. As in high-diving contests, the *degree of difficulty* of the inductive task must be weighed against the speed in which it is completed. In the world of computation, this means that a slow solution to a difficult problem can be considered as efficient, elegant, and beautiful as a fast solution to an easy one. Nobody wants an inefficient procedure of any generality, but it is not at all obvious that a fast procedure with a small scope should be preferred, in general, over a slow one with a large inductive scope when both procedures are efficient.

2.4. The Methodologist as Product Designer

One response to the complicated interactions of the normative desiderata just discussed is to formulate a theory that can weigh, for example, the value of a certain decrease in inductive scope against the value of a certain increase in speed. But I doubt that there is a general norm of this sort to be discovered. In some situations speed is more valuable than inductive scope. For example, the possible, natural languages may be a small subset of all possible languages. So a child need not have a broad inductive scope to acquire the language of the culture he is assigned by the luck of the draw. But he must be capable of learning any natural language very quickly if he is to prosper in the society he finds himself in. So Chomsky's famous "rationalism" comes to no more than the thesis that speed is more valuable than broad inductive scope in the task of language acquisition.

In others situations, however, the reverse is true. For example, a social scientist must often face a staggering range of plausible hypotheses about the causes of social phenomena, but social science, as a whole, is not under any absolute time constraints to find a true hypothesis regarding such causes (grant proposals notwithstanding). So inductive scope is valued more highly than speed in this case. In the language acquisition case it can be argued that the space of possible languages is constrained by the inductive scope of the language learning mechanism children are supplied with from birth, so the small inductive scope of this learning mechanism is guaranteed to be suited to its task by a sort of "preestablished harmony". In unrestricted scientific inquiry, however, there is no such guarantee---unless one is willing to posit (along with Descartes, Spinoza, and Leibniz) that God has created the world in a way perfectly suited to man's peculiar intellectual abilities.

In light of the natural variation in the respective values accorded to suitability, inductive scope, and computational speed, I view the methodologist more as a product designer than as a judge who settles all conflicts among competing, epistemic values. A product designer does not adjudicate among conflicting values in the marketplace. Rather, he creates new markets by optimizing conflicting values in different directions. For example, rapid acceleration, impressive fuel economy, and high impact resistance cannot all be achieved in the same automobile. But one can build a sports car that is as efficient and safe as possible, an economy car that is as safe and sporty as possible, or a family sedan that is as sporty and efficient as possible. Each product finds its own market niche, and no customer is the less rational for buying the product best suited to his needs.

But if the market's desires are unsatisfiable (everybody wants a military tank that can win a drag race--- without shooting or crushing the competition) the firm's advertising should point this out so that customers can sort their unsatisfiable demands into those they hold most dear, and those they are willing to sacrifice. In this way, each customer can reformulate his demands, and can be sold an item that most nearly optimizes them.

In keeping with the industrial design metaphor, the task of the methodologist is twofold. First, he should expose unsatisfiable demands on discovery methods as the confusions that they are. Everybody loses when the customer cannot possibly be satisfied. Second, for any satisfiable set of demands, the methodologist should try to design a solution that optimizes them. For example, there may simply be no procedure with a given inductive scope that always conjectures suitable hypotheses. But if there is one, then the methodologist should find one that is near-optimal in resource consumption. And if resource consumption must increase with inductive scope, then the methodologist should develop fast and narrow (sports car) methods as well as slow but secure (family sedan) methods. Neither sort of method is intrinsically better than the other, any more than a sports car is intrinsically better than a family sedan.

Shifting perspective from that of the High Priest to that of a product designer has some advantages. Perhaps the greatest is that the methodologist's burden is transformed thereby from an amorphous, hopeless task to a relatively clear, mathematical one. But it is a mathematical task that still has a strong philosophical motivation, for a demonstration that a norm cannot possibly be satisfied is a good reason to junk it, along with all the vicious and outlandish demands of tyrants and petty dictators.

Needless to say, the kind of study I am proposing can also have significant practical import. Nor is this potential foggy and distant. A.I. "expert systems" are finding real industrial applications analyzing chemical constitution, diagnosing soybean diseases, searching for oil deposits, making medical diagnoses, optimizing computer programs, and recommending custom lubrication blends for huge, industrial machines. Many of these programs already employ a rudimentary subroutine for inductive inference. Such programs can provide immediate and useful field applications for advances in the logic of discovery.

2.5. Related Disciplines

Now that the logic of discovery has been described in a positive manner, needless confusion can be avoided by distinguishing it from more established disciplines that have distinct, but overlapping, aims. Such disciplines include statistics, cognitive psychology, the history of science, the philosophy of science, computational linguistics, computation theory, and artificial intelligence.

2.5.1. Statistics

Statisticians evaluate hypothesis generation methods under the rubric of *parameter estimation*. In parameter estimation problems, the possible worlds to be distinguished may be thought of as distinct populations. The evidence consists of samples drawn randomly from the population according to an assumed sampling distribution. The hypotheses are assumed to be parameterized by the real numbers. An inductive device is taken to be a function from finite population samples to particular parameter values. A standard criterion of identification assumed in the parameter estimation literature is that the expected value (with respect to the sampling distribution) of the parameter conjecturing device (considered as a random variable) is the distribution's true parameter value. An estimator that identifies a world in this sense is said to be *unbiased* for the world.

There are also interesting philosophical questions regarding the relationship of generative methodology to Bayesian epistemology. Bayesian norms provide coherence constraints on an agent's degrees of belief. But there are many relationships a generated hypothesis could have to a rational system of beliefs. An obvious aim would be to produce an hypothesis with a nearly maximal probability conditional on the total evidence available.

J.J. Horning [Horning69] has proposed a method that acheives just this aim in

generating the grammar of a language from positive examples of well-formed strings. He defines an effective *a priori* distribution over grammatical hypotheses, as well as an explicit method for calculating the likelihood of a string from a grammatical hypothesis that generates it. His procedure always conjectures a maximally probable hypothesis in light of the total evidence. But he also takes a device to identify a language just in case its probability of conjecturing a maximally probable, true hypothesis approaches unity as the evidence increases. Finally, he begins to address issues of efficiency by finding ways to avoid considering hypotheses that cannot possibly carry maximum probability in light of the evidence.

Bayesian epistemology is a sophisticated, normative theory that provides a compelling account of how *a priori* preference and hypothesis suitability with respect to the evidence should interact. But its interest would be enhanced through the study of methods by which a finite system might *use* its degrees of belief to *construct* a maximally probable hypothesis in light of new evidence. An ability to construct probable hypotheses is crucial to the communication of scientific progress even if an individual Bayesian could get along without it. Horning's thesis is a step toward such a method. The details of his work will be examined in more detail in the next chapter.[9]

2.5.2. Cognitive Psychology and "Concept Learning"

The logic of discovery, as I conceive it, is a normative, formal discipline. Psychology, on the other hand, is the empirical science whose aim is the prediction and explanation of human behavior. It is perfectly possible for a psychologist to explain actual inferential behavior by appeal to causal mechanisms without ever evaluating the performance of these mechanisms. But given the paucity of direct evidence about how such inferences are actually produced, psychologists often explain actual behavior by appeal to good methods that would lead to this behavior [Bruner56]. A method is a good one if its generality and resource consumption are suited to the inductive needs and computational limitations one might expect humans to have. So the relationship between psychology and normative methodology is tighter than one might expect. This affinity does not imply that the study of generation methods is psychology. Rather, one popular approach to

---

[9] Another question is whether generality is compromised by Bayesian method. As we have seen, Osherson, Stob and Weinstein [Osherson86] have defined a set of worlds that are in the scope of some effective discovery method but that are not in the scope of any Bayesian method. This proof may lead to a fertile investigation of the relationship of inductive generality to Bayesian method as the definitions and proof techniques are further refined to yield more informative and natural results.

psychology is normative.[10]

Psychological interest in inductive inference is concentrated in the study of *concept learning*. The concept learning literature flourished in the nineteen-fifties. It is not surprising, therefore, that the possession of a concept is "operationalized" as the subject's ability to respond "yes" to examples of the concept and "no" to counterexamples. But even though the psychological criterion for having learned a concept is somewhat crude, a respectable variety of learning tasks have been considered. For example, Bruner, Goodnow, and Austin [Bruner56] entertain problems in which the target concept is definable in terms of the evidence vocabulary and problems in which it is not. They consider problems in which many features of the examples and counterexamples are irrelevant, and problems in which they are not. In some problems the order of presentation of an instance is relevant to class membership and in others it is not. They also consider problems in which the simplest adequate hypothesis is a conjunction or a disjunction.

In some problems a payoff matrix for proper classification is appealed to in order to explain apparent biases in classification. For example, someone told that a certain kind of blip on a screen is an invading enemy bomber will tend to "recognize" more blips as bombers than someone who is told that blips of this kind are enemy spy planes and all other blips are friendly planes. In the first case, missing a bomber is much worse than sighting too many, while in the second case, the cost of sighting too many or too few blips is the same.

Some of the concept learning literature is more explicitly computational. An example of this sort of proposal is Earl Hunt's Concept Learning System [Hunt66]. Hunt's system reads descriptions in a monadic predicate language of objects that have or do not have a certain "target" property, and then produces a quantifier-free, monadic definiens in the vocabulary of the instances as a conjectured definition of the target property.

Hunt observed that his algorithm is guaranteed to conjecture a true definiens when every possible state description has been read, either as a positive or as a negative instance. So he was clearly interested in the breadth of scope of his method. He also performed some experiments comparing human performance to that of his program, but he was careful to point out that although his interests began with the modeling of human cognition, they widened to a general, performance-oriented interest in effective inductive methods.

---

[10] Quine has recommended that epistemology is psychology--- the study of how the evidence causes conjectures in humans. It is therefore ironic that many psychologists explain conjecturing behavior by appeal to the rationality of methods that might have produced the actual behavior.

2.5.3. The History of Science

Many philosophers who claim interest in the logic of discovery these days are interested in the history of science.[11] But according to my conception of the logic of discovery, the study of past discoveries is hardly mandatory or even central to the discipline.

On the other hand, the methods we employ do interest us just because we use them, so there is a special reason to analyze their properties. One might also expect that the past success of actual scientific practice is good evidence that scientists' methods are good ones. But caution is advisable, for individual scientists may simply be lucky in particular cases. Although the methodological significance of Kuhn's work on scientific revolutions has been overdrawn, his opinion that actual scientific inquiry is fragmented into paradigms that die primarily through the deaths of their proponents may not be. Actual scientists can carry rigid, *a priori* commitments to an hypothesis to the grave, and may be more similar to toasters than we would like to think.

Even if single instances of success in history are poor evidence for the generality of the method actually employed, showing that a scientist's method is general can constitute an explanation of the actual scientist's success. In this way, the logic of discovery can contribute to the force of explanations of discovery episodes in the history of science.

Social scientists also attempt to explain actual discovery events, but in social, rather than conceptual, terms. One social scientist has gone so far as to explain the genesis of mechanics by appealing to the early breast-feeding habits of the infant Newton and to his father's socio-economic status [Manuel68]. Conceptual historians are often disappointed by such sociological explanations. Perhaps the sociological explanation is disappointing because it explains only the fact that a particular hypothesis is conjectured, rather than the fact that the conjecture produced by the method is adequate. For example, there is no obvious connection, causal or logical, between the inverse square law and Newton's ability to breast feed. It would be more satisfying to know that the social forces operating on Newton caused him to behave as a relatively powerful and efficient discovery system. Breast-feeding habits are not sort of stuff of which such an account is made. Newton's vast familiarity with mathematics, along with his desire to beat the Royal society at its own game, would seem more relevant to his eventual success.

---

[11] For a representative sample of such papers, see [Nickles80].

2.5.4. The Philosophy of Science

Chapter one reviewed the anti-generationist position in philosophy, but it did not discuss the positive work of Hans Reichenbach and Hilary Putnam in the logic of discovery. Although some of their respective views are justifiably defunct, their general approach to the evaluation of discovery methods is not. In the following paragraphs, I attempt to winnow the wheat from the chaff.

Reichenbach

Hans Reichenbach held that the paradigmatic task of the scientist is to assess the probabilities of events of various types. Reichenbach assumed Von Mises' frequentist theory of probability, according to which the probability of a type of event is the limiting relative frequency of events of this type in a given, infinite sequence of events. Since any limiting, relative frequency of a type of event is consistent with any observed, finite sequence of events, the assessment of objective probabilities is a non-trivial inductive problem.

Reichenbach addressed this problem by proposing the *straight rule* as a method for discovering the probabilities of events of different types. On any observed, finite sequence of events, the straight rule conjectures that the probability of event type T is no further than k from the observed relative frequency of events of type T (where k is fixed over all conjectures). So for example, if nine out of eighteen observed balls are black, then the straight rule conjectures that the limiting relative frequency of black balls in an infinite sequence of trials lies somewhere in the interval [0.5-k, 0.5+k].

Reichenbach's familiar "pragmatic vindication" of the straight rule is really an argument about inductive scope. A rule is taken to be successful in a sequence if all but finitely many of its conjectures are true. The inductive scope of the straight rule is identical with the set of all sequences in which the type of event under study has a limiting relative frequency. Moreover, no possible procedure whose conjectures state that the limiting relative frequency of a sequence is in a certain interval can identify a sequence that has no limit. Hence, the scope of the straight rule is maximal, given the hypothesis language assumed by Reichenbach. If one is vindicated in selecting an inductive method that is as general as any other, one is vindicated in choosing the straight rule as an inductive strategy.

One reason for the demise of the straight rule literature is an expectable disappointment in *pragmatic reductionism*, an attempt to define hypothesis suitability

in terms of maximal inductive scope. The proposal is that an hypothesis is suitable on given evidence just in case it is generated from this evidence by a method whose inductive scope includes that of any other method [Salmon66] p.87. It is easy to see that *any* hypothesis about limiting relative frequency is suitable in this sense, for an arbitrary, finite period of insanity has no impact on the inductive scope of a method: an unhealthy result.

But the triviality of this version of pragmatic reductionism does not imply that there is *no* interesting relationship between inductive scope and hypothesis suitability that a pragmatic reductionist might get his teeth into. For example, imagine two proposed suitability relations S and R. If it were to turn out that some method whose conjectures all satisfy S with respect to the input evidence is more general than any method whose conjectures all satisfy R with respect to the evidence, then a more liberal pragmatic reductionist might say that S is a *better* norm of evidential support than R.[12] And we can make such comparisons even if no method is more general than every other.

More importantly, the idea that *ceteris paribus*, a more general method is a better one, is entirely independent of pragmatic reductionism--- even in its more liberal guise. One can value truth and beauty without reducing beauty to truth or truth to beauty. Similarly, one can be interested in inductive generality and hypothesis suitability without reducing suitability to generality or *vice versa*.

Another shortcoming of the straight rule literature is Reichenbach's implausible attempt to distill the essence of empirical science as the problem of estimating limiting relative frequencies. But this inductive problem is artificial even from his own theoretical perspective. For example, Reichenbach thought the point of knowing a probability value is to ensure more success than failure making predictions. But if the target sequence is recursively enumerable, it would be far more useful to know a program that generates the sequence than to know the sequence's limiting relative frequency. The former knowledge leads to successful prediction of the occurrence of events of the type in question in every instance, while the latter guarantees only more success than failure after some finite time. But Reichenbach entertains no method that conjectures programs for enumerating sequences of events, and he offers no reason whatever for excluding them.

Furthermore, the straight rule's maximum generality (the property that "vindicates" it)

---

[12] Putnam assumes this more relaxed, pragmatic reductionism in his attack on Carnap's theory of logical probability (c.f. below).

is an artificial consequence of the fact that every competing method is assumed to conjecture only probability intervals. If we add one simple hypothesis to the hypothesis language (i.e. "The probability of event type T does not exist") then there is no nontrivially achievable inductive scope that includes every other such scope.[13]

But none of these objections to Reichenbach's frequentist epistemology constitutes a general objection to the interest of inductive scope. We can be interested in inductive generality without being pragmatic reductionists and without neglecting other, independent desiderata such as hypothesis suitability and efficiency. We need not assume that hypothesis generation problems are the only problems of interest that arise in practice or that it is useful to reduce all such problems to hypothesis generation problems. We need not study only a particular concept of identification. Nor need we assume that discovery methods may only conjecture probability intervals. Nor must we expect to find a method with a relatively rich hypothesis language that is as general as any other method. In short, interest in inductive scope need not imply narrow-mindedness.

Putnam

Recall the thesis that a suitability relation is better insofar as discovery methods that "rely on it"[14] are more general. Putnam [Putnam63] assumes something stronger in order to attack Carnap's familiar theory of *degrees of confirmation*: a suitability relation is *defective* if there is a world (recursive set) that some method can identify but that no method relying on this relation[15] can identify. Since every world can be identified by some method (the one that conjectures only hypotheses

---

[13] Notice that for each sequence there is a method that identifies it. Hence, if no method identifies every sequence, then there is no method that is more general than every other.

Methods conjecture either that the limiting relative frequency of a kind of event in an infinite sequence of events is within plus or minus k or they conjecture that no limit exists. A method identifies a sequence just in case all but finitely many of its conjectures are true (EX_* identification). It is non-trivial just in case it does not conjecture the whole unit interval.

Theorem: No nontrivial inductive rule can identify all sequences.

Proof: Suppose R can. Then R can identify any sequence in which the limiting relative frequency of K's is zero. So we begin by defining each place in σ as a non-K until R sees the light and conjectures an interval including zero. Immediately, we begin defining subsequent places in σ as K-events. Since R can identify all sequences, it eventually sees the light and conjectures an interval including 1. Since R is nontrivial, this hypothesis is distinct from its previous one. Continuing forever, we arrive at a sequence σ that R cannot BC_* identify Q.E.D.

[14] i.e. methods that produce only hypotheses that are suitable with respect to the input evidence.

[15] Carnap's theory has the odd consequence that the degree of confirmation of any universally quantified sentence is zero. If reliance on a c-measure were defined as conjecturing only hypotheses that have a certain measure value, then no method relying on a c-measure in this sense would identify any worlds. To give Carnap half a chance, Putnam defines "reliance" in an attenuated way: if its conjecture entails that the next event will be of a certain kind, the hypothesis that the next event will be of this kind has a sufficiently high degree of confirmation on the evidence seen so far.

adequate for this one world), the first requirement is trivial. The second requirement is not. But Putnam discovered a way to construct, for any given method that relies on Carnap's concept of suitability, a world that this method cannot identify. This is the first case (I am aware of) in which a general, negative result has been obtained about inductive scope in an epistemological context.

Putnam's negative result reveals an interesting interaction between a proposed suitability relation and inductive scope even if one does not infer from it that Carnap's theory is defective. But Putnam cannot resist.

> ....a good inductive judge can do things, provided he does *not* use "degree of confirmation" that he could not *in principle* accomplish if he *did* use "degree of confirmation". As soon as a scientist announces that he is going to use a method based on a certain "c-function,", we can exhibit a hypothesis (in fact, one consistent with the data so far obtained, and hence possibly true) such that we can say: if this is true, *we* shall find it out; but you (unless you abandon your method) will never find it out. [Putnam63] p.77.

Putnam proceeds further. Carnap's theory is not defective because of some mere flaw in design. Any theory remotely like it must be defective as well.

> ...it is easily seen that any method that shares with Carnap's the feature: what one will predict "next" depends *only* on what has so far been observed, will also share the defect: either what one should predict will not in practice be computable, or some law will elude the method altogether (one is *in principle* forbidden to accept it, no matter how long it has succeeded). (p. 773).

Actually, E. Mark Gold showed something stronger than this [Gold65]. Any method that is a function of the evidence alone fails to BC_*-identify the recursive sets, which are exactly the worlds assumed in Putnam's construction. This result covers ineffective methods as well as effective ones, and methods that do not rely on Carnap's method as well as those that do.

On the basis of his negative result, Putnam invents something most unusual: an *argument* against the study of hypothesis generation methods. It runs like this. Consider an hypothesis *selection* method. Hypotheses are "suggested" to the method, and each suggestion is placed at the end of a queue. The method conjectures its oldest suggestion until this suggestion is inconsistent with the evidence. Then the next hypothesis in the queue is conjectured, and so forth.

Notice that for *every* world, it is *possible* that the selection method identifies this world. But as we have seen, for *every* generation method that is a function of the evidence alone, *there is* a world that it cannot *possibly* identify. Therefore, the

selection method dominates every generation method, so it is irrational to choose any generation method over the selection method.

The anti-generationist import of this argument is obvious.

> ...we should take the view that science is a method or possibly a collection of methods for *selecting a hypothesis, assuming languages to be given and hypotheses to be proposed.* Such a view seems better to accord with the importance of the hypothetico-deductive method in science, which all investigators have come to stress more and more in recent years. (p. 783, my emphasis).

But this bomb is easily defused. Notice that the argument depends upon two distinct modalities, possible worlds of investigation and the possibility of an hypothesis being "suggested" in a given world. Putnam's dominance argument depends on a specious emphasis on the former modality at the expense of the latter. To expose this sophism, notice that the selection method *can* fail to identify *every* world, while any generation method with a non-empty inductive scope *cannot* fail to identify *some* worlds (in fact, it *must* identify every world in its scope, by definition). That is, any generation method with a nonempty scope "dominates" Putnam's selection method. And the extent of the domination increases as inductive scope increases. So by parity of reasoning, use of the selection method is irrational, and the sort of "hypothetico deductivism" urged by Putnam should be abandoned. The whole point of using a discovery method is to increase the odds of scientific success over those to be expected in sifting through the conjectures of monkies banging on typewriters. Since Putnam ignores this point entirely, it is not surprising that he sees little use for discovery methods.

2.5.5. Computational Linguistics and "Learnability Theory"

Noam Chomsky takes the object of linguistic theory to be the characterization of the "nature" of natural language. It seems reasonable to suppose that a language doesn't get to be a natural language unless the garden-variety human baby can learn it in a few years without formal instruction. That is, natural language is learnable. So any proposed characterization of natural language had better result in a class of languages that can be be learned from examples, for examples are the only kind of evidence the child receives.

In light of Chomsky's ambitions for linguistics, E. Mark Gold introduced a notion of inductive scope based on limiting, syntactic convergence. On the basis of his definition of inductive scope, he showed that if negative examples are excluded

from the evidence language, no method can BC^*-identify any class of languages that includes all finite languages and at least one infinite one [Gold67].[16] On the other hand, if negative examples are allowed, then the set of all primitive recursive languages is syntactically identifiable in the limit.[17] Gold also showed that no method, effective or not, can identify every recursive set. As we have seen, this result generalizes Putnam's.

Gold's techniques are quite analogous to Putnam's, but they had a much greater impact on linguists and computer scientists than Putnam's results had among philosophers. Although Gold's results are mathematically simple, they were electrifying. Here was a mathematical technique for demonstrating the inadequacy of proposed definitions of natural language. The limiting criterion of identification is extremely weak, but so long as the result is negative, the weakness of the criterion of success only enhances the power of the result.[18]

Gold has, since then, extended his interest to complexity theory (c.f. chapter five below). Gold's followers have studied numerous criteria of grammar adequacy and convergence. Jerome Feldman has investigated the possibility of converging to minimal-length grammars adequate for a target language, along with generalized concepts of convergence to be exploited later in this thesis. Horning's work, alluded to above, is also devoted to the language acquisition problem. Kenneth Wexler, a linguist, has recently written a large book on the subject of *learnability theory*, which is what the subject spawned by Gold is called by those who pursue it.

---

[16] Let C be a class that includes every finite language and one infinite one. Now let M be a device that claims to BC^*-identify every language in C. Let L be an infinite language in C. We construct an enumeration of L on which M makes infinitely many inadequate conjectures. Enumerate L in any manner. Repeatedly present the first string in L to M. Since M can identify every finite language, it can identify the language containing just the first string of L. So M must must conjecture a grammar for this language. Immediately feed the next string of L to M and repeat it. Since the first two sentences of L constitute a finite language, L must eventually conjecture a grammar for this language. Now add the third sentence of L and repeat it, and so forth. M clearly conjectures infinitely many grammars for distinct languages. But M is eventually fed a complete enumeration of the positive instances true of L. Therefore M fails to BC^*-identify L, and hence the scope of M does not include C.

[17] Enumerate the primitive recursive programs effectively. It is decidable whether a string is accepted or rejected by such a program. As each new instance is read, test the current program to see whether it accepts all positive instances and rejects all negative ones. If the program fails the test, reject it. When an adequate program is reached in the enumeration, conjecture it and read a new instance. Repeat the procedure.

If the target language is primitive recursive, there is some first program for it in the enumeration. Moreover, since the enumeration is complete, any program ahead of it in the enumeration is eventually refuted on some evidence. So after seeing finitely many evidential instances, the correct program is conjectured, and never revoked thereafter.

[18] Actually, this point is undercut by the stringency of Gold's requirement that the grammar generate exactly the language.

2.5.6. Computation Theory and "The Mathematical Theory of Inductive Inference"

In 1975, Manuel Blum wrote a seminal paper called *A Mathematical Theory of Inductive Inference* [Blum75]. Blum's paper treats the identification of recursive functions as opposed to languages, although all recursive languages can be thought of as recursive characteristic functions. And instead of employing grammars as hypotheses, Blum employs Goedel numbers of programs that compute the functions to be identified. But aside from these largely notational substitutions, Blum's work extends Gold's by introducing complexity measures into proofs of identifiability. Blum's interest in discovery matters is understandable, given that Minsky was his thesis advisor, and Minsky and Pappert wrote a book on the inductive scopes of "perceptrons" [Minsky69], which are computational models of neural nets with perceptual abilities.

The literature that has developed in light of Blum's paper is called *the mathematical theory of inductive inference* by those familiar with it. It differs from learnability theory both in increased mathematical sophistication and in a corresponding decrease in attention to applications and motivation.

Recent developments of this tradition include a systematic comparison of the inductive scopes obtainable according to criteria of convergent success that vary along several dimensions (e.g. probability p of converging to an adequate hypothesis, convergence to an hypothesis with n errors, convergence in n steps to an adequate hypothesis) [Case78], [Pitt84] as well as attempts to extend Blum's complexity theory to convergent computations (c.f. chapter five) [Daley84].

Another recent development in the lineage of Blum's paper is Ehud Shapiro's system for inferring logical axiomatizations adequate (by a certain criterion) with respect to a relational structure from sampled atoms true in the structure [Shapiro81]. His work unifies that of Gold and Blum with the extensive work on mechanical theorem proving by the "resolution method". Shapiro's work is the starting point of my own positive work, and will be described in detail in the next chapter.

2.5.7. Artificial Intelligence and "Machine Learning"

In artificial intelligence, the study of generation methods is called *machine learning.* A famous example of this genre is P.H. Winston's paper *Learning Structural Descriptions from Examples* [Winston75]. Winston's paper is something of an enigma. On the one hand, he claims that "simulation of human intelligence is not a primary goal of this work", p. 160. On the other hand, the paper is devoid of any sense of evaluation of the procedure presented. The hypotheses conjectured by the device are "semantic networks" which are labeled, directed graphs whose arcs represent relation symbols or copulas and whose vertices represent individuals or monadic predicates. Since no clear, semantic theory is provided for the networks, it is not obvious what inductive problem the machine has solved when it solves one.

The BACON project [Bradshaw80] is another familiar example of work in machine learning. The inductive scope of the BACON program includes a significant class of functions definable by short, polynomial expressions. Although their proposal is an interesting discovery method, Langley and Simon tend to advertise it as a simulation of actual human discoveries. They express the importance of computational tractability, but are more reticent about scope. Instead, they appeal to particular historical examples, and illustrate the steps their program undertakes in producing actual physical laws.

The DENDRAL project is yet another example of a program that embodies machine learning techniques. This project remains one of the few examples of an inductive system that employs a substantive background theory to constrain inductive inference over a theoretically interesting and practically useful class of alternatives. The program receives a raw mass–spectrogram and a chemical formula as input, and returns a conjectured class of stereo structures for the analyzed compound as output. A very clever feature of this program is Lederburg's graph enumeration algorithm which heavily restricts the space of molecular structures considered *a priori* without any danger of missing any live possibilities. Another interesting feature of the system is its reliance on syntactic relations of confirmation and explanation that are familiar in the philosophy of science literature.

There are too many different kinds of machine learning proposals to enumerate here [Carbonell83]. Like the traditional, philosophical literature on inductive inference, papers in machine learning tend to promote particular programs or to discuss difficulties encountered in attempting to design a program in a particular way. Given this focus on particular procedures, the machine learning literature's

strength lies in its attention to intuitively compelling applications for the procedures proposed, and in the nuts-and-bolts issues that arise in implementing these procedures. The corresponding weakness of the discipline is its inability to demonstrate that a problem has no tractable solution, or to show that a proposed program is optimal for the problem it solves. Both of these sorts of problems require theorems that quantify over all possible programs, while the techniques of workers in machine learning apply only to the design of particular programs.

A.I. in general and machine learning in particular has an ambiguous relationship with cognitive psychology. Its practitioners sometimes claim to propose psychological models. At other times, machine learning practitioners claim to be doing something that looks much more like the logic of discovery as I have described it. For example, Carbonell and Michalski write

> There is no reason to believe that human learning methods are the only possible means of acquiring knowledge and skills. In fact, common sense suggests that human learning represents just one point in an uncharted space of possible learning methods-- a point that through evolutionary process is particularly well suited to cope with the general physical environment in which we exist. More theoretical work in machine learning has centered on the creation, characterization, and analysis of general learning methods, with the major emphasis on analyzing generality and performance rather than psychological plausibility. [Carbonell83] p. 5.

This description of the machine learning literature is more reasonable, for the psychological credentials of much of this work are dubious, and experimental evidence is almost unheard of in the field. But it also suggests a bit more attention to "generality and performance" than is actually evident in the literature. At best, the ambiguous aims of machine learning might inspire an interdisciplinary literature in which considerations of generality and efficiency combine with genuine, psychological observation to inspire new psychological theories. At worst, the ambiguity may be (and has been) exploited to avoid both the formal evaluation of machine learning procedures and their assessment as good, empirical hypotheses about human cognition.

2.5.8. Miscellaneous

My enumeration of fields related to the logic of discovery is far from exhaustive. For example, electrical engineers pursue a curious form of the logic of discovery under the rubric of "pattern matching". The motivation for this work appears to have been to learn to decipher pictures coded as formal grammars to enable machines to process "bubble chamber" photographs produced at physical laboratories

automatically [Fu75]. Aside from the novel application, this massive literature is heavily indebted to learnability theory for theoretical insight. Another large literature concerns statistical "clustering" techniques for concept formation.

2.6. Problems of Inductive Generalization

From my discussion of the computational literature on the logic of discovery, it should be evident that inductive procedures have been studied in light of very different applications. The applications all share one characteristic, however. Some evidence is given about particular individuals and an hypothesis is demanded that extends the evidence received. That is, the device is expected to perform an *inductive generalization*. Inductive generalization is a kind of inductive inference, but not every inductive inference is an inductive generalization. For example, one may infer that whatever happens in a given system will happen analogously in an analogous system. But such analogical inferences are not generalizations in my sense of the term. Another inductive inference problem that is not obviously a generalization problem is that of extending a mathematical system in a useful manner. For example, consider the problem of extending the natural number system to the real number system, or of inventing analytic geometry [Manders86].

The problems reviewed in the previous section concern language learning, concept acquisition, discovering physical laws, automatic programming, and the induction of logical theories. A natural question is whether these generalization problems are all as different as they appear to be, or are mere notational variants of one another. And insofar as they are similar, is the similarity the result of a deep insight into the nature of inductive generalization, or is it merely a reflection of theoretical short-sightedness?

The answer to the first question is that many of these problems do tend to fit naturally into a single framework. The answer to the second question is that there are deep similarities among many of these problems that do seem to lack motivation. One point of the positive work in this dissertation is to help loosen up intuitions about generalization problems by exhibiting a problem that breaks free of the usual mold, but that may be analyzed by natural extensions of known techniques.

2.6.1. An Analysis of Generalization Problems

Most of the problems addressed in the literature on inductive inference can be analyzed in a uniform manner. First, there is *evidence*. This evidence is true or false *of* something, which I have already called a *possible world*. In grammatical inference problems, for example, the "possible worlds" are simply distinct languages. A string marked with a plus sign is called a "positive instance" and a sting marked with a minus sign is called a "negative instance". A positive instance is true of a language just in case the marked string is in the language, and a negative instance is true of a language just in case the marked string is not in the language. Finally, there may be a probability distribution over possible worlds. As was mentioned earlier, this distribution would provide a notion of the probability of eventual success for any given device. Intuitively, such a distribution might reflect the degrees of belief of an agent who is evaluating the performance of various methods.[19]

Given that we have worlds and evidence about these worlds, there is a question about how the evidence gets from the world into a given generation method. It is often assumed in formal theories of generalization (e.g. Gold, Blum) that the evidence fed to an inductive method is true and complete. But a moment's reflection on inductive problems faced in practice reveals that such high-quality evidence is encountered only rarely. Air resistance, stray cosmic rays, the power of suggestion and fallible apparatus are all examples of factors that can riddle the evidence with falsehood. And besides falsehood, evidence can remain incomplete, even in the limit. For example, due to the laws of general relativity, there are certain observations crucial to distinguishing substantive hypotheses about the topology of the universe that cannot be made in principle [Glymour77]. To learn the extent to which the ill effects of noise and incompleteness can be minimized, we must add to the generalization problem a sampling function that for any given world, produces a relatively noisy or incomplete evidence sequence for this world. And so that the relationship between noise and inductive scope can be investigated, a measure of the noise of an evidence sequence with respect to a world would be desirable. Despite the interest of noisy evidence, I do not address this sort of problem in this thesis.

But scientists are not always passive receptors of evidence. It would therefore be nice to provide an inductive device with an ability to "perform experiments" (i.e. to

---

[19] Curiously, I know of no such proposal of this sort in the computational literature on induction, but it is perfectly natural.

query the world about the truth of a given evidence sentence). Of course, there is no more guarantee that a question put to nature will receive a true answer than there is that evidence resulting from passive observations must be true. Whether or not the inductive agent has control over the evidence acquisition mechanism, the way the evidence is sampled in a given world may well be stochastic. If the inductive agent can perform experiments, the sampling distribution will be a complicated function of inductive procedures and the evidence they have already sampled. J.J. Horning has studied a simpler problem, in which each sampled evidence instance is independent from every other and is independent of the method receiving the evidence. There is obviously a lot of work to do to enhance the realism of the data acquisition aspect of generation problems, but I do not address these issues in this thesis.

In the Blum literature, the possible worlds are recursive functions, and the evidence consists of ordered pairs marked as positive and negative instances. A positive instance is true of a function just in case it is in the graph of the function, and a negative instance is true of a function just in case it is not in the graph of the function. No oracle is considered, and the evidence is always assumed to be noise-free. The BACON program solves a problem that falls approximately within the Blum paradigm. The program receives as evidence ordered pairs drawn from the graph of an n-ary polynomial function restricted to the rational numbers. An oracle is assumed, and the oracle may be noisy.

In Shapiro's setting for inferring logical theories, the possible worlds are first-order relational structures and the evidence consists of true atoms or their negations, with truth defined in the usual model-theoretic manner. Shapiro also assumes an infallible oracle for the evidence language.

In the concept learning literature, the possible worlds are sets of state descriptions, typically in a monadic predicate language without function symbols. The evidence consists of the same state descriptions, classified veridically as positive and negative instances. A positive instance is true of a set or "concept" just in case it describes only elements of the target set, and a negative instance is true just in case it describes only elements in the complement of the target set. It is assumed that no state description is satisfied by distinct objects that are, respectively, in and out of the target set. Some concept learners are supplied with oracles and others are not.

Now that we have worlds and evidence about them, it is time to consider worlds

and hypotheses adequate for them. Most of the problems in the computational literature assume that an adequate hypothesis for any world can be found in a fixed hypothesis language. In grammatical inferenee, hypotheses are formal grammars, and a grammar is usually taken to be adequate for a language just in case all and only strings in the language can be derived from the rules of the grammar.[20] As if there were not already enough plausible places to inject probabilities into generalization problems, hypotheses may be probabilistic as well. Horning, for example, has studied the inference of stochastic grammars for languages with assumed sampling distributions.

In the Blum literature, hypotheses are Goedel number encodings of computer programs.[21] A program (or its index) is adequate for a recursive function just in case it computes the function. Although Simon sometimes speaks as though the point of BACON is to do what human scientists would do, adequacy for a polynomial expression could be defined naturally as satisfaction in the structure of the real numbers by all and only the ordered pairs making up graph of the function under investigation.

In Shapiro's setting for inferring logical theories, an hypothesis is a finite set of first-order sentences, and such an hypothesis is adequate for a first-order relational structure just in case the hypothesis is true in the structure, and each element of the diagram of the structure is derivable in "few enough steps" from the hypothesis. So the hypothesis must be true and very informative about the structure.

In the concept learning literature, an hypothesis is a formula open in one variable that is satisfied by all and only the elements of the target set in the assumed, fixed relational structure. That is, the open formula must define the target set in the language of the assumed structure.

Finally, a problem comes equipped with an identification criterion and a *proposed scope* which is just some subset of the fixed universe of possible worlds. A method solves a generalization problem just in case it identifies every world in the problem's proposed scope by conjecturing sentences in the problem's hypothesis language on the basis of evidence provided in the manner specified by the problem.

To recapitulate, an inductive problem specifies

---

[20] For those readers who are unfamiliar with formal linguistics, the details are unimportant for the main point of this discussion. It will suffice to think of a grammar as an uninterpreted proof system in which some uninterpreted strings are derivable but others are not. For a fine presentation of the details see [Hopcroft79].

[21] Or indices for an acceptable numbering of the partial recursive functions [Rogers67], to be more pedantic.

- A universe of possible worlds
- A proposed scope (i.e. a set of possible worlds)
- A probability distribution over the proposed scope (optional)
- An hypothesis language
- A concept of hypothesis adequacy in a world
- An evidence language
- A concept of evidential truth in a world
- A (possibly noisy or stochastic) process for sequentially sampling evidence from a world
- A (possibly noisy) oracle for evidential truth (optional)
- A concept of the noise of an evidence sequence with respect to a world (optional)
- A concept of identification defined in terms of hypothesis adequacy

It is easy to see that each of these parameters can be "wiggled" to generate a broad range of variations of any given problem. For example, Gold examined the significance of eliminating negative instances from the hypothesis language. The proposed scope can be larger or smaller. In language acquisition problems, for example, smaller scopes are defined as those classes of languages generated by grammars of restricted forms. The concept of adequacy can be varied in many ways. Case and Smith [Case78], for example, have examined adequacy criteria for which programs that compute no more than n values of a function incorrectly are still adequate with respect to that function. Langley and Simon have investigated error tolerance in an empirical way by supplying the device with a parameter that makes the BACON program more or less responsive to sprinklings of noise. How this phenomenon relates to inductive scope remains to be seen. Finally, many different notions of identification have been investigated, including EX^*, EX^n, BC^*, BC^n, and all of these criteria with errors. Probabilistic identification has also been investigated in various ways by Horning [Horning69] and Pitt [Pitt84]. Pitt has investigated the interaction of the probability of identification with the number of errors consistent with hypothesis adequacy, with curious results.

2.6.2. Artificial Intelligence and Artificial Similarity

The analysis of inductive generalization problems highlights their similarities. They are analogous, for example, in uniformly assuming that each evidence sentence concerns only the properties of a unique individual. Another similarity is that each problem assumes that a possible world is a countable structure. A third similarity is that adequacy is always defined in terms of truth and a lower bound on the amount of information the hypothesis conveys. Finally, each of these problems is naturally describable as a generalization problem, as opposed to being a problem of hypothesis test, of choice among given hypotheses, of drawing an analogy, or of extending a mathematical framework to one that is easier to investigate (as in the progressive elaboration of the real number system) [Manders86].

Sometimes these similarities and limitations are urged as objections to the study of such problems. Perhaps the objections result from a fear that anyone interested in these problems must have a naively narrow and old-fashioned conception of science. Admittedly, there is a long tradition in philosophy of overstating the centrality of inductive generalization in scientific method. Plato, Aristotle, Bacon, Mill and Reichenbach are all guilty, to a degree, of this mistake. But I advocate pluralism rather than reductionism when it comes to types of methodological problems. After all, I took pains in the second chapter of this work to refute the dogma that only problems of hypothesis test are worthy of methodological investigation. It would be counterproductive to enshrine my own dogma in place of the defeated one.

Another objection to the study of generalization problems is that they are too simple to be worth considering, for the *really difficult* task is to "invent" new concepts that neither appear in the evidence nor are fixed in advance of the inductive process. But some commonly studied generalization problems cannot be solved without introducing what certainly seem to be novel theoretical concepts. In grammatical inference problems, for example, the only predicate[22] that either appears in the evidence or is assigned a fixed extension by the inductive problem is "is-a-sentence". But most interesting languages cannot be axiomatized without introducing further predicates like "is-a-noun-phrase" or "is-a-verb". The form of an introduced predicate is irrelevant. What is important is that the extension of the predicate is fixed not by the problem to be solved but by the conjectures invented by the inductive agent solving the problem. The grammatical concepts that must be introduced are not mere toys. A casual acquaintance with grammatical theory

---

[22] Technically, I refer to the non-terminal symbols in a context-free grammar as "predicates". This correspondence is natural, and in the next section, I show how to make it precise.

reveals that theoretical concepts like "noun-phrase", "verb-phrase", and "preposition" are thought of as genuine features of natural language and are employed in serious explanations of syntactic phenomena.

A weakened version of the last objection is that one may be able to define non-trivial generalization problems, but only the trivial, uninteresting ones are soluble. But if identifying each of infinitely many infinite worlds that cannot be identified without inventing arbitrarily many novel concepts is a non-trivial problem, then some familiar, non-trivial generalization problems are indeed soluble. Even as Hempel doubted that a machine could ever sensibly introduce theoretical terms, linguists and computer scientists were proposing algorithms to solve grammatical inference problems of exactly this sort [Feldman67], [Solomonoff64], [Horning69], [Pao78].[23]

Another objection to the interest of generalization problems is that they ignore the importance of background knowledge in constraining inductive inference. But this suspicion is mistaken in two ways. First, we can think of the worlds that are not in the proposed scope of a problem as being those in which the background knowledge of the evaluator of an inductive system is false. The worlds in the proposed scope of his problem are exactly the worlds representing open possibilities he takes seriously even in light of all he knows. Second, an inductive device that somehow "knows" what the background knowledge implicit in an inductive problem is will surely be better at solving it than a device that considers hypotheses adequate only in worlds not in the problem's proposed scope. Indeed, we can represent the background knowledge of a procedure as the complement of its inductive scope. The worlds in this set are those the machine does not take seriously, and background knowledge may be described as our standard for serious possibility [Levi83].

So generalization problems are of genuine interest despite their restrictive similarities, and despite the fact that their study needn't exhaust all methodological concerns. But many of the problems reviewed earlier are similar in another manner. Consider the following two properties of generalization problems:

1. Each hypothesis is adequate in at most one world.

2. Each hypothesis specifies a procedure for enumerating all and only the positive instances true in the (unique) world in which it is adequate.

---

[23] I do not wish to criticize Hempel's good will in examining the computational literature. He was quite familiar with the machine learning literature when he expressed his doubt. The problem, I suspect, is that he did not see how similar grammars are to the logical theories with which he was familiar. And it is just an accident that no explicitly logical program introduces theoretical terms in the sense required.

A grammar, for example, specifies a procedure for enumerating the unique language it "generates", and thereby specifies a procedure for enumerating the positive instances true of this language.

A program index, as conjectured by Blum's inductive inference machines, specifies a procedure for enumerating the graph of the (unique) function it computes, and the positive instances true of a function are just the elements of the function's graph. A polynomial function is obviously computable over the rationals, so the same may be said of the problem addressed by the BACON program.

Now recall the concept learning literature. If the hypothesis language is simple enough, an open sentence provides a mechanism for enumerating the state descriptions that satisfy it. In most cases, "concepts" are taken to be quantifier-free, in which this criterion is certainly met. So these simple concept-learning problems satisfy these conditions as well.

It is just a bit more involved to see that Shapiro's problems also satisfy these conditions. First, he says that an evidence language and hypothesis language form an *admissible pair* just in case for any structure, if an hypothesis is false in that structure, it must entail some evidence sentence that is also false in the structure. Then he refuses by fiat to consider any inductive problem whose evidence and hypothesis languages do not constitute an admissible pair. (Notice that this assumption excludes out of hand those problems in which the hypothesis includes "theoretical predicates" that do not occur in the total evidence.) So for any such problem, the requirement that adequate hypotheses be true is vacuous. For consider an admissible pair of languages. If the intersection of the deductive closure of an hypothesis with the evidence language is the complete evidential theory of a structure, then the hypothesis must be true in the structure.[24] The only remaining requirement is that the hypothesis entail exactly the evidence sentences true in the structure. But the completeness and consistency of first-order logic guarantees a procedure that can, for any given sentence, enumerate all the consequences of this sentence. So Shapiro's problems all satisfy the axiom in question.

---

[24] Shapiro does not really want to require admissibility of the evidence and hypothesis languages, however. In one of his examples, the hypothesis language is a first-order language with non-logical vocabulary {+,s(.),0}. The evidence language consists of atomic sentences over the non-logical vocabulary of the hypothesis language. But this pair is clearly inadmissible. Consider the hypothesis '(x)(x+0=0) & (x)(y)(z)(x+y=z ---> x+s(y)=s(z))'. Now consider a structure in which the usual natural numbers are augmented by five elements a,b,c,d,e that are not successors of any natural number, such that s(b)=d, s(c)=e, a+b=c, but a+d is not identical to e. The hypothesis entails every atomic sentence true in the structure just described, but it is false in this structure. What Shapiro really wants to require is that for any structure in which every domain element is denoted by a closed term of the evidence language, if an hypothesis entails exactly the evidence sentences true in a structure then the hypothesis is true in that structure. According to this criterion, his example languages are evidently admissible.

Finally, even the inductive problem assumed in Putnam's construction satisfies the two axioms. Each hypothesis amounts to an open formula of number theory that represents a recursive set.

It is striking that problems drawn from such different areas of application all satisfy axioms (1) and (2). It is tempting, therefore, to assume that all other generalization problems satisfy them as well. But these axioms express a very stringent lower bound on the predictive power of an adequate hypothesis. In logical terms, these axioms require that an adequate hypothesis entail every positive instance in the total evidence true of a world. But an induced theory can be interesting, explanatory, informative, and true without singling out a unique world or providing an enumeration procedure for the total evidence. In fact it can achieve these aims without entailing *any* of the true evidence.

For example, consider a problem in which the hypothesis language is a first-order language and the evidence language consists of the atoms of the hypothesis language. Assume that an hypothesis is adequate just if it is true, invalid, and of the form of the universal closure of an open formula with at least one free variable. So for example, "all ravens are black" would be an adequate hypothesis for a world in which it is true. This is clearly a generalization problem but it fails to satisfy both axiom (1) and axiom (2). First, an adequate hypothesis may be adequate for many worlds, and second, an adequate hypothesis need not entail any evidence, let alone all of it. "All ravens are black" is such an hypothesis.[25]

It might be objected that the problem just presented does not place stringent enough lower bounds on the strength of adequate hypotheses. This objection has some merit, but there are many ways to place non-trivial explanatory demands on adequate hypotheses without assuming axioms (1) and (2). In chapter five, I introduce a problem of this sort.

2.6.3. A Logical Perspective

Computer science flourishes in the proliferation of equivalent notations, and this fact is evident in our representative sample of inductive problems. Hypotheses can be grammars, open formulae, Goedel numbers, "semantic nets", logical theories, or finite-state automata. Possible worlds may be functions, sets of strings, relational structures, or sets of state descriptions. Evidence sentences may be ordered pairs drawn from the graph of a function, strings in a language, or atomic sentences true in a structure, or state descriptions of objects.

[25] It entails only disjunctions of atoms, but never any atoms.

Different representational schemes can make a real difference to issues of efficiency and control when employed as data structures in a program. But they tend to obfuscate the comparison of problems as contrasted with the procedures that solve them. Moreover, when an expressively weak formalism suggests no more powerful extension, there is a danger that its limitations will be missed. For example, concept learning theorists who study the inference of boolean combinations of "attribute values" seem to have missed the possibility that a concept's definition might require quantification, binary relations, and function symbols.

Logical languages are powerful (they can express uncomputable problems) and adaptable (the syntax and vocabulary can be adjusted in in a variety of ways). Moreover, logical systems come equipped with a superabundance of metatheoretical results from proof theory and model theory. Hence, there may be some interest in formulating inductive problems in a logical framework.

Shapiro's problems are already presented logically, with relational structures as possible worlds, and with purely universal Horn clauses as hypotheses. Simon's polynomial function definitions are also open first-order formulas, and the possible worlds to be inferred can be thought of as the structure of the real numbers augmented by the function defined. In the usual concept-learning problem, the hypothesis language is tantamount to a monadic formula open in one free variable.

Now consider the inference of *context-free* grammars. A context-free grammar may be expressed as a finite list of context-free production rules along with a distinguished *start symbol*, which is conventionally taken to be 'S'. A context-free production rule consists of the funny symbol '::=' flanked by a capital letter on the left side and an arbitrary, finite string of capital and lower-case letters on the right-hand side. For example, the context- free production 'S::=aBc' is read "S rewrites as aBc". A grammatical derivation of a string of lower-case letters is begun by writing down the start symbol. Thereafter, one writes down the result of substituting the right-hand side of a rule for some occurrence of the rule's left-hand side in the current last line of the derivation. The derivation is completed when no capital letter occurs in the current last line of the derivation. The set of all strings of lower case letters derivable from a grammar is the language *generated* by the grammar.

There is an obvious translation of context-free grammars into first-order axioms of a restricted sort. Consider the following example:

S::=SBa	(x)(y)[(S(x) & B(y))--->S((x*y*a)]
S::=b	S(b)
B::=bB	(x)[B(x)--->B(b*x)]
B::=S	(x)[S(x)--->B(x)].

The translation works as follows. The capital letter on the left side of a context-free rule becomes a monadic predicate on the right hand side of a universal conditional. The concatenation of distinct capital and lower case letters on the right-hand side of the rule corresponds to a term built up from an application of a concatenation functor '*' to distinct variables and constants, respectively. The constants are the translated lower case letters, themselves. The variables correspond uniquely to the capital letters on the right-hand side of the rule. The antecedent of the conditional is just the conjunction of the predication of each capital letter in the right-hand side of the rule to its uniquely corresponding variable. The last step is to take the universal closure of the resulting open formula. Notice that if no capital letter occurs on the right-hand side of a rule, the corresponding axiom has no antecedent (or equivalently, a tautologous one).

The sense in which a translation has been provided is just this. A string 'ab...z' is derivable from a context-free grammar just in case the atom 'S(a*b*...*z)' is entailed by the corresponding translation.[26]

Since the consequent of the translation of any context-free rule is atomic, such axioms are basic Horn sentences [Chang73], p. 328. But not every basic Horn sentence is such a translation. First, all predicates must be monadic. Second, only one function symbol is permitted, and this must be binary. Third, this function symbol occurs only in the consequent. Any Horn sentence that satisfies these three properties may be called a *context-free* hypotheses to highlight the analogy to context-free productions.

Once grammars are viewed as logical theories, there is a natural, logical reconstruction of the other elements of grammatical inference problems. The evidence, for example, consists of instances of atoms and negated atoms of the form 'S(a*b*c*a*c)', where S is the translation of the start symbol, and the closed term corresponds to the formal string that results when the constants themselves are concatenated.

---

[26] Indeed, there is an easy transformation of grammatical derivation lines into lines of a logical proof by refutation.

Possible worlds correspond to relational structures <U,L>, where U is the set of all finite strings on an alphabet of lower-case letters, and L is a subset of this set, which is assumed to interpret the predicate 'S'.

From this logical perspective, the grammatical inference problem raises familiar epistemological issues. For example, grammatical inference problems can be viewed realistically, instrumentalistically, or positivistically. The positivist version of the problem is the one just described, where there is no "fact of the matter" in the target "world" about what constitutes a noun-phrase. The meaning of the expression 'noun-phrase' derives entirely from its employment in a conjectured theory of the predicate 'is-a-sentence'. In the realist version of the problem, the structure <U,L> is augmented by other phrasal categories, and each predicate in the true grammar denotes its extension in the structure. In this case, there is a fact of the matter about how a sentence is to be analyzed, and the realist requires of any adequate hypothesis that it get these facts more or less right. Finally, an instrumentalist (e.g. [vanFraassen80]) may admit that there is a fact to the matter of phrasal structure, but a theory is deemed by him to be adequate so long as it gives the correct judgments of sentential well-formedness. Getting the "hidden" phrasal categories right is not required.

The correspondence between grammars and theories also illustrates epistemic situations ignored by philosophers of science. For example, consider a realist version of the grammatical inference problem in which the positive and negative instances are not just strings, but *bracketed strings*. A bracketed string is generated from a grammar by enclosing the right-hand sides of all the grammar's productions in parentheses, and by treating parentheses as lower-case letters in any derivation. So for example, consider the simple grammar

S::=Sa
S::=b.

The string 'baaaaa' is derivable from this grammar. We can obtain the corresponding, bracketed string '(((((b)a)a)a)a)' by writing down the corresponding, bracketed grammar

S::=(Sa)
S::=(b)

and by deriving bracketed string from it, employing the same sequence of production rule selections as in generating the unbracketed string.

The motivation for bracketed strings in the evidence is that a native speaker might be able to employ pauses and emphasis to indicate the bracketing which would

enhance the information available to a learner [Crespi-Reghizzi71]. In logical terms, the outermost bracketed string provides an instance of the sentence predicate 'S' as always. But the bracketing of a substring of a positive instance says more than that the string has *some* property. It says that the string is in some one of the finitely many phrasal categories of the target language. Let *R* be a second-order predicate whose extension is fixed as the set of all categories of the target language (e.g. the extensions of the predicates relevant to its actual derivation). So for example, the overall, logical translation of the positive instance

(((b)a)a)

is just

S(b*a*a)

(EX)[*R*(X) & X(b*a)]

(EX)[*R*(X) & X(b*a)]

As far as I am aware, philosophers of science have not examined inductive problems with this strange kind of second-order evidence. But they are commonplace in learnability theory.

The point of this exercise was to illustrate the analogies and disanalogies of problems by viewing them in a common, logical framework. We have seen that most of the hypothesis languages studied in the computational literature can be viewed very naturally as logical axiomatizations. Moreover, we have seen that innocuous assumptions about problems expressed in other formalisms (e.g. bracketed strings) can lead to interesting problems of inductive generalization that might have been missed from the logical point of view had we not considered how such problems are related to one another.

2.7. Chapter Summary

The purpose of the previous chapter was to dissolve the standard, philosophical objections to the principled study of hypothesis generation procedures. The purpose of this chapter was to portray the logic of discovery in a positive light. In this chapter, I sketched a normative study of hypothesis generation methods. Factors relevant to a method's evaluation include the suitability of its conjectures, its inductive generality, and the computational costs involved in its pursuit. I am pessimistic about any attempt to rank these desiderata across all applications. Rather, the methodologist should identify unsatisfiable combinations of these norms, and he should design a variety of optimal methods that may accentuate some virtues at the expense of others.

Next, the aims of the logic of discovery were contrasted with those of related disciplines. At the same time, the numerous contributions of these disciplines to our understanding of generation methods were discussed. The resulting survey of methods and problems suggested some strong analogies, which were reflected in an anatomy of generalization problems. Various, standard objections to the study of such problems were rejected, including the claim that such problems do not require that novel concepts be invented. Next, I showed that many problems in the literature are artificially similar in their imposition of unrealistic lower bounds on the strength of adequate hypotheses. Finally, I sketched the utility of model theory as a tool for comparing the difficulty of different generalization problems expressed in different formalisms.

The various topics addressed in this chapter are unified by two important themes. The first is pluralism. Generalization problems are not the only or even the most important inductive problems encountered in practice. Among generalization problems there are myriad variations. There are various criteria for evaluating inductive methods, and there need be no absolute adjudication among them or reduction of one to the other.

The second theme is that pluralism is *not* anarchy. *Ceteris paribus*, a faster and more general method that tends more often to produce suitable hypotheses is a *better* method. That no method is universal does not imply that no method is better than any other. And the fact that different problems demand different solutions does not imply that one method cannot be better than another for a given problem. Such norms may not be the grandiose one-liners that have been sought by philosophers for millennia. But they are precise, non-trivial, and best of all, available.

Chapter 3
How Discovery Methods Work

The previous chapter described the logic of discovery at a general level, and discussed criteria for the comparison and evaluation of hypothesis generation methods. But if nobody has any hunches how to design a good generation method, there is little point in staking out a discipline devoted to their study. As with flying saucers, we may as well wait until we run into a discovery method or think we can design one ourselves before we spend time disputing how to study the things.

But as a matter of fact, precise, solutions have been found for some inductive generalization problems, and the detailed examination of these methods suggests some techniques for designing new ones. The purpose of this chapter is to provide a taste of the kinds of nuts-and-bolts issues that arise in designing such methods. This taste is provided through the dissection of four discovery procedures that have been proposed in the computation-theoretic literature.

The methods to be examined are the automaton inference procedures of Pao [Pao69] and Angluin [Angluin81], the grammatical inference procedure of Horning [Horning69], and the model inference systems of Ehud Shapiro [Shapiro81]. Although these procedures address distinct problems in different ways, their designers share an explicit concern for inductive generality and efficiency. The tension between the joint aims of inductive generality and computational ease is what makes the logic of discovery interesting and difficult.

3.1. Trimming the Hypothesis Enumeration

Before proceeding to the methods of Pao, Angluin, Horning and Shapiro, it is useful to examine some informal distinctions between approaches to the improvement of a generation method's efficiency. To illustrate these distinctions, I appeal to a very simple but general kind of method called an *enumeration method* [Gold67]. An enumeration method generates a tape on which every possible hypothesis eventually occurs. When each evidence sentence is received, it

conjectures the first hypothesis on its tape that is consistent with the evidence it has seen so far. Enumeration methods can be powerful (so far as EX*-identification is concerned).[27] But despite their inductive power, enumeration methods seem ugly and inefficient.

The apparent ugliness of enumeration methods can be explained, in part, by the fact that they make no use of the results of previous tests to guide the selection of hypotheses for future tests. Control over the selection of the next hypothesis to test against the evidence is abdicated to the arbitrary, fixed order of the assumed enumeration. For example, there is no finite bound on the number of equivalent formulations of a failed hypothesis[28] that an enumeration method can test before producing its conjecture on given evidence. An enumeration method can also test arbitrarily many hypotheses entailing previously falsified hypotheses before it arrives at a conjecture. In general, enumeration methods do not exploit the semantic structure of the hypothesis language to restrict the test and consideration of hypotheses whose consideration is a waste of time.

3.1.1. Kinds of Hypothesis Neglect

While some hypotheses can be ignored in any circumstances without compromising inductive scope, others may be ignored only in light of the evidence so far received. For example, only one element of each equivalence class of hypotheses need be considered in any circumstances.[29] But hypotheses that entail a given hypothesis may be ignored only after it is discovered that the given hypothesis is refuted. Hence, it is natural to say that some hypotheses may be ignored *a priori* (i.e. no matter what) while others may be ignored only *a posteriori*, or in light of the available evidence.

---

[27] Assume that each world has an adequate hypothesis in the enumeration, and each hypothesis inadequate for a world can be shown to be inadequate on the basis of a finite set of evidence. Then after some finite amount of evidence is read, every inadequate hypothesis preceding the first adequate hypothesis in the enumeration will have been rejected. The first adequate hypothesis is never refuted, and is therefore conjectured forever after.

[28] In general, two hypotheses are equivalent in a problem if they are adequate for exactly the same worlds in the problem's proposed scope.

[29] Two hypotheses are taken to be equivalent in a problem if they are adequate for exactly the same worlds in the proposed scope of the problem.

3.1.2. Hypothesis Test vs. Hypothesis Consideration

Notice that there is an important difference between withholding an hypothesis from test and ignoring it altogether. First, imagine a modified enumeration procedure that works just as before, but prior to testing the next hypothesis, it checks whether this hypothesis is in a fixed, normal form.[30] If the hypothesis turns out not to be in normal form, it is rejected before it is tested against the evidence. This procedure withholds any non-normal hypothesis in its enumeration from empirical test *a priori*, with no attendant loss in inductive generality. But the procedure obviously *considers* every non-normal hypothesis it decides not to test.

Next, imagine an enumeration procedure that employs an enumeration of only the hypotheses that are in normal form, rather than of the entire hypothesis language. This procedure does not merely withhold non-normal sentences from test; it fails to consider them at all, with no accompanying loss in inductive scope.[31] Not only does it save the time that would have been spent in testing hypotheses. It saves the time and space that would have been involved in generating and storing them as well. Empirical tests can be very expensive[32] so avoiding unnecessary tests is good. But it is better (if feasible) to ignore hypotheses that need not be tested.

3.1.3. Description vs. Computation

It may seem simple to modify an enumeration method to ignore lots of hypotheses with no attendant loss in inductive scope: "ignore every hypothesis that entails a refuted one." But this proposal is no method in my sense, for it does not specify *how* to ignore the hypotheses in question. To *characterize* the set of hypotheses that may be ignored on given evidence is not to provide a general *procedure* that ignores this set (i.e. that enumerates the complement of the characterized set when it is fed the evidence as input). And designing such a procedure may be difficult or even impossible for computational reasons.[33] Each of the methods to be discussed below can be viewed as a genuine attempt to bridge the gap between merely *describing* the hypotheses that need not be tested and specifying *how* to avoid their test or consideration.

---

[30] A "normal form" of a language is just a decidable, expressively complete sublanguage of this language.

[31] Of course, it is assumed that the procedure that enumerates the normal form hypothesis does so directly, without considering non-normal hypotheses. This can certainly be done in some cases. For example, the sentences in disjunctive normal form can be enumerated by a procedure that does not enumerate all possible sentences and then delete the non-normal ones.

[32] Evidence for this claim is presented in chapter six.

[33] E.g. if the specified set is R.E. but non-recursive, then its complement cannot be enumerated recursively.

3.2. Pao's Method for Inferring Finite State Acceptors

3.2.1. The Regular Set Inference Problem

The algorithm to be considered appears in Pao's PhD. dissertation [Pao69]. Pao's hypothesis language is the set of all *nondeterministic, finite state automata.* A nondeterministic, finite state (NFS) automaton is a finite, directed graph whose vertices are called *states* and whose arcs are called *state transitions.* One state is designated as the *initial state* and some states are designated as *accepting states.* There is a set of *input symbols,* and each arc or state transition is marked with exactly one input symbol. An NFS automaton is said to *accept* a finite string of input symbols just in case the input string is an edge label sequence of a path from the initial state to some accepting state.

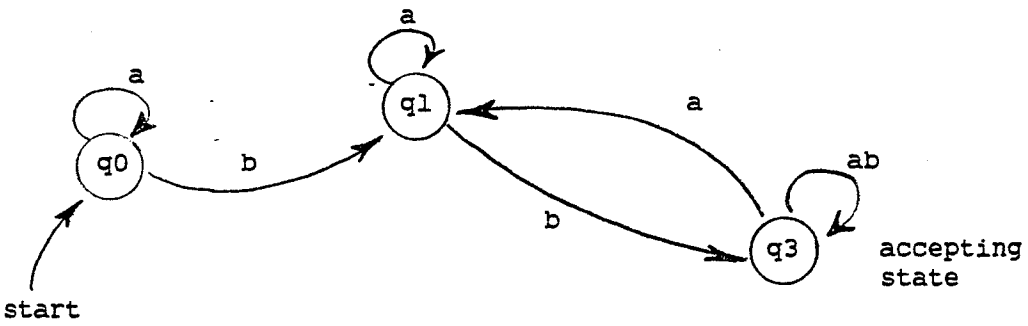


Figure 3-1: A Nondeterministic, Finite-State Automaton

Pao takes possible worlds to be *languages* or arbitrary sets of finite strings of input symbols. The *language of* an automaton is just the set of all strings the automaton accepts. Two automata are *equivalent* just in case they accept the same language. The problem's proposed scope is the class of all languages that are accepted by deterministic, finite state automata. These languages are called the *regular sets* by computer scientists. The evidence language consists of the set of all finite strings of input symbols. A veridical oracle for string membership in the target language is assumed. That is, an inductive device has the prerogative to ask the oracle about the membership of a given string in the target language, and it is guaranteed to receive the correct answer.

A transition is said to be *live* if it occurs in some accepting path, and is *dead* otherwise. A finite set of strings is *representative* of an automaton just in case the automaton accepts each string in the set and for every *live* transition in the automaton there is some string in the set such that the acceptance of this string

exercises the transition. A set is a *representative sample* of a language if it is a representative set for some acceptor of the language. Like a good work-out, a representative sample forces an automaton to "exercise" each of its muscles---except for the "vestigial" ones that are never involved in accepting strings.

Pao takes an inductive method to *identify* a language if and only if for every automaton that accepts this language and for each sample of the language that is representative of this automaton, the method can output a finite-state acceptor of the target language in finite time after asking only finitely many questions of the oracle.[34] So a method solves the problem just in case it identifies every regular set in this sense.

There is a countable infinity of distinct regular sets, and infinitely many distinct regular sets are themselves countably infinite. So there is nothing trivial about the proposed scope of the problem so far as its cardinality is concerned. Notice that Pao's identification criterion is not quite the same as EX*-identification. In EX*-identification, a proposed method need never stop making queries, so long as it converges to an adequate hypothesis after making finitely many questions have been asked. That is, the method need not "know that it knows" the correct answer. But according to Pao's criterion, the method must "know that it knows" and must stop making queries at this point. On the other hand, Pao exempts a method from failure if it has not been provided with a representative sample before it makes its first query. So any method is free to assume that it has been given a representative sample before it makes its first query. In EX*-identification, however, the evidence must eventually be representative, but there is no point at which a device may safely assume that it is.

Finite state automata correspond to context-free grammars of a special kind [Hopcroft79]. In particular, the states of an automaton correspond to unique, non-terminal symbols occurring in a grammar that generates the language the automaton accepts. Recall that any context-free grammar can be thought of as a universally quantified logical formula, such that the predicates in this formula correspond to non-terminal symbols in the grammar. By composing translations, we can also think of finite state automata as universally quantified logical sentences. As it turns out, the initial state of a finite state machine corresponds to the "is-well-formed" predicate and all the other states correspond to "theoretical" monadic

---

[34] Chomsky appears to have been among the first to propose this somewhat quirky identification criterion. It was later adopted by Feldman and Solomonoff. An analogous version is employed in Shapiro's logical inference system, and the exact problem proposed by Pao is recently resurrected in a paper by Angluin, which we shall examine shortly.

predicates defined contextually by the organization of the machine. So a procedure that conjectures automata on the basis of input strings actually introduces new "theoretical predicates" in the usual sense of Hempel and Carnap.

3.2.2. A Clunky Solution

By Pao's definition of identification, it is clear that no inductive method is ever penalized for assuming *a priori* that the original sample is representative of the target language. This free assumption is very powerful, for it places a finite bound on the number of hypotheses that must be considered. The reason is simple enough. Let N be the sum of the lengths of the strings in a finite set that is representative of the target language. If every live transition of some finite state acceptor for the target language is exercised in accepting a finite set of finite strings, then the number of live transitions in this acceptor cannot exceed N. Moreover, the result of deleting any dead transition from an automaton accepts the same language as the original automaton, for no accepting path is created or disrupted by eliminating a transition that occurs in no accepting path. Therefore, the language must be accepted by an automaton with no more than N transitions. But no connected graph with N transitions has more than N−1 vertices. Therefore, some acceptor of the target language has no more than N transitions and N−1 states. The set of all automata with no more than N transitions and N−1 states is finite, and can be generated mechanically.

So the only task that remains is to to eliminate all inadequate hypotheses from our finite set of alternatives on the basis of finitely many oracle queries. Since we "know" that an adequate automaton is in our constructed set, we "know" we have eliminated all inadequate automata when only equivalent automata remain in our set. It is decidable whether two automata are equivalent. It is also decidable whether a given automaton accepts a given string: just run the automaton.[35] So an obvious method is to enumerate all possible finite input strings and to ask the oracle about the status of each. Any time the oracle says 'yes', eliminate each automaton that does not generate it, and any time the oracle says 'no', eliminate each automaton that generates it. Eventually, every inadequate automaton must be eliminated, and only equivalent automata remain. We can recognize this situation, for the equivalence of finite-state automata is decidable. When all remaining automata are equivalent, we choose one of them (perhaps the one with the shortest description) as our conjecture. This procedure is clearly a general solution to the regular set inference problem.

---

[35] It must stop either when the input string is used up, or we arrive at a state where there is no transition for the currently scanned input symbol.

3.2.3. Pao's Solution

Pao's solution improves on our clunky one both by withholding hypotheses from test and by ignoring some hypotheses altogether. First of all, the obvious method just presented made no use of the structure of the representative sample. Pao corrects this oversight by means of what Feldman [Horning69]) calls an *ad hoc* acceptor for a given, finite sample. The *ad hoc* acceptor for a finite set of strings is constructed as follows. Assume an initial state. Now select a string from the sample. Add a state to the *ad hoc* machine and connect the initial state to it by a transition labeled with the first symbol occurring in the string selected. Add another state, and connect the previous state to this one by an arc labeled with the second symbol occurring in the selected string. Repeat this process until the end of the string is reached. Designate the last state added as an accepting state. The *ad hoc* machine accepts the first string in the sample. Now select a distinct string from the sample. Run the *ad hoc* machine constructed so far on this string. If the string is not accepted, then either the machine stops at some point before the string is entirely read, or the machine reads the entire string and does not end up in an accepting state. In the first case, repeat the above procedure, starting at the state where the machine choked, using the unread portion of the string it choked on. In the second case, label the state in which the machine finds itself when the input string is read as an accepting state. Repeat the overall procedure until no strings are left in the sample. In the end, the constructed machine must accept exactly the strings in the sample. The accompanying figure depicts an example of a sample and its corresponding, *ad hoc* acceptor.

Consider an arbitrary NFS automaton. Eliminate all dead transitions and then eliminate each state that is not connected to the initial state. Call the resulting automaton M. Consider a representative set for M, and form the *ad hoc* acceptor for this set. This *ad hoc* machine must be *homomorphic* to M (e.g. M is isomorphic to some result of identifying states in the *ad hoc* machine). Therefore, the *ad hoc* acceptor of a representative sample of a language is homomorphic to some acceptor for this language. So once the *ad hoc* acceptor of a representative sample is constructed, we know that some result of identifying some of its states is an acceptor for the target language. By testing only results of identifying states in the *ad hoc* acceptor, Pao's procedure *ignores* a vast number of hypotheses that were considered and tested by our clunky method.

But we must be careful not to hide a problem under the rug. Is it possible to generate just the automata that result from identifying states in the *ad hoc*

ab
abaa
abab
abbb

Figure 3-2: Sample

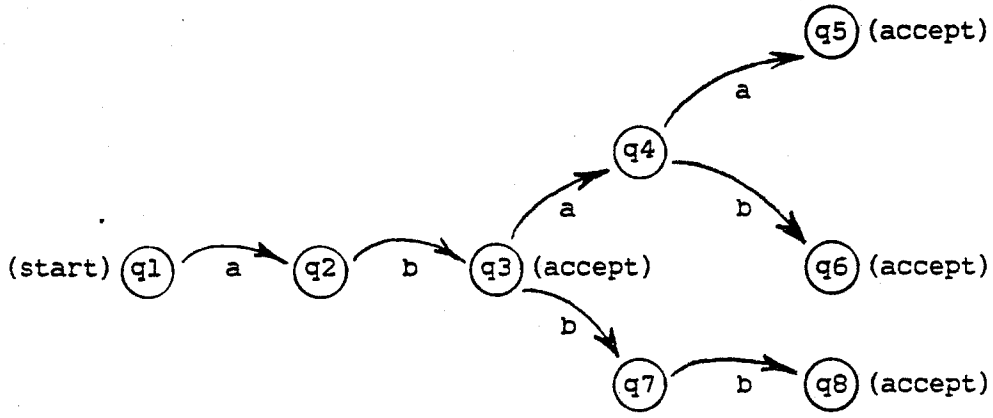


Figure 3-3: *Ad hoc* Acceptor

automaton without in some sense generating all the other automata of the same number of states and sorting the result? In this case, the answer is yes. Think of the set of states in the *ad hoc* acceptor. Any result of identifying states (and dragging the transitions along) corresponds to a unique *partition* of the set of states. The *ad hoc* automaton itself corresponds to the partition in which each state is in a distinct cell. The result of identifying states q and p in this automaton corresponds to the partition in which all states are in distinct cells except for q and p, which are in the same cell. So each result of identifying states in the *ad hoc* acceptor corresponds uniquely to some partition of its set of states.

The algorithm adopted by Pao to generate the set of all partitions of a given, finite set works as follows. Let S(n) be a set of integers from 1 to n, representing n distinct automaton states. The recursive function PART(n) takes a positive integer n as argument and is intended to take the partitions of S(n) as value. The definition of PART(n) is:

- PART(1) = {{{1}}};
- PART(n) = {P U {{n}}: PϵPART(n−1)} U {P−{p} U {p U {n}}: pϵP and PϵPART(n−1)}

So in the case of PART(3), we begin with {{1,2,3}} and {{1,2},{3}}. The first set in the above expression evaluates to

{{1,2},{3}}

{{1},{2},{3}}

while the second evaluates to

{{1,2,3}}

{{1,3},{2}}

{{1},{2,3}}.

By inspection, the union of these sets includes all the partitions of {1,2,3}. In a sense,[36] this function generates exactly the set of automata resulting from collapsing states in the *ad hoc* automaton, without wasting resources to consider other automata. So Pao's procedure can really be said to ignore lots of hypotheses tested by our "clunky" procedure. And in a sense, it does so *a posteriori*, or as a function of the particular sample provided.

But Pao does not stop here. She realizes that some results of identifying states in the *ad hoc* automaton need not be tested against the evidence given that others are known to have failed. For imagine the result of identifying two states in a given automaton. The collapsed automaton still has all the accepting paths the non-collapsed automaton has, plus all the new ones that arise from the state identification. So the language of the collapsed automaton includes the language of the non-collapsed one. Therefore, if an automaton fails to accept a string the oracle claims to be in the language, then any automaton that collapses into it fails as well. And if an automaton accepts a string that is not in the language, any result of collapsing its states must fail. So by deciding whether one automaton collapses into another, Pao's procedure can avoid gratuitous tests. In Pao's own words,

> Since we will use the above two principles, some f.s. [finite state] machines in W [the set of results of identifying states in the *ad hoc* machine]...will be eliminated before participating in any comparisons with other f.s. machines in W. Therefore, actually, we do not have to construct those f.s. machines.
>
> [Pao78], p. 12.

There is one final way in which Pao's system is better than our clunky one. Recall that the clunky procedure queries the oracle about every string. Queries can be very expensive. Computer peripheral devices are very slow compared to core operations. More metaphorically, scientific experiments require expensive apparatus

---

[36] To be made precise later, in chapter five.

and time consuming observations. Therefore, it would be advantageous to make every query count as an *experimentum crucis* between two "live" hypotheses remaining in our finite set of possibilities. Accordingly, Pao develops a clever technique for constructing, for any given pair of inequivalent automata, a string that is accepted by one but not by the other. The result of querying the oracle regarding such a string must constitute a crucial experiment between the given automata.

Let M, M' be inequivalent automata. We can construct a crucial string for these machines as follows. First, we form the *direct product* machine MM' from M and M'. The states of MM' are the ordered pairs <q,q'>, where q is a state of M and q' is a state of M'. There is a transition labeled c from <q,q'> to <p,p'> just in case there is a transition labeled c between q and p or between q' and p'. Finally, the start state of MM' is just the ordered pair of the respective start states of M and M'. The *relative complement* machine M−M' is just MM' with every state <q,q'> such that q is an accepting state and q' is not an accepting state declared to be an accepting state. M−M' accepts a string just in case M accepts the string and M' does not. So to generate a crucial string for M,M', all we need to do is to start at an accepting state in M−M', and to proceed backwards until the start state is reached. Reversing the sequence of transition labels passed along any such path results in a crucial string for M and M'.

Pao's system has three advantages over the clunky procedure. First, it exploits the structure of the input sample to ignore many automata that the clunky procedure tests against the evidence. Second, it utilizes the "collapses into" relation to withhold certain hypotheses from empirical test given that other hypotheses have already been tested. And finally, it relies on the structure of the hypotheses to construct crucial experiments, rather than blindly consulting nature about facts irrelevant to its live alternatives.

Despite these successes, there is room for improvement. Pao's method generates and stores the set of all partitions of a given, finite set of states. This is a smaller set than was generated by the clunky algorithm, but it is not a small set. For example, if the given sample contains just one string with five symbols, there would be fifty-two live hypotheses to generate and to decide among. For a string of length 10, the number of live hypotheses rises to one hundred fifteen thousand, nine hundred seventy-five. For a string of length fourteen, the algorithm would be responsible for paying attention to one hundred ninety million, eight hundred ninety-nine thousand, three hundred twenty-two partitions. Pao's algorithm blows its

gasket on extremely small samples. An obvious question, then, is whether this kind of effort is necessary to solve the problem, or is, rather, a contingent shortcoming of Pao's approach. As it turns out, Angluin's procedure is a significant improvement over Pao's.

3.3. Angluin's System for Inferring Minimal Regular Set Acceptors

3.3.1. The Minimal, Deterministic, Automaton Inference Problem

Angluin's identification criterion is different than Pao's. According to Angluin, an NFS automaton is adequate for a regular set just in case it is *deterministic* and has fewer states than any distinct (up to isomorphism), deterministic, finite state acceptor of this set.[37] A *deterministic* finite state (DFS) automaton is just an NFS automaton such that for each state and input symbol there is exactly one transition labeled with this input symbol from the given state to another (not necessarily distinct) state. So an NFS machine can differ from a DFS machine in two ways. First, it may have two distinct transitions from the same state that are labeled with the same input symbol. Second, some states may have no transition with a given label. These additional restrictions do not alter the proposed scope of Pao's problem, however, for any language accepted by a NFS automaton is also accepted by a DFS automaton.

Given this construal of hypothesis adequacy, Angluin defines identification as follows. A method can identify a regular set just in case for any sample of the language that is representative of its minimal DFS acceptor, the method outputs an adequate hypothesis after making at most finitely many queries to the oracle. In short, Angluin's problem is more strict than Pao's, in that a successful method must find not just any acceptor for the target language, but a minimal, deterministic one.

3.3.2. Angluin's Solution

While Pao's method begins with an *ad hoc* automaton and searches the ways of collapsing distinct states, Angluin's does just the opposite: it assumes that two states in the *ad hoc* automaton are equivalent until it can be demonstrated by means of a crucial test that they are not.

Angluin's method is based on the concept of *state equivalence*. Two states are

---

[37] That there is such a unique minimal state deterministic acceptor for each regular set is a consequence of the Myhill-Nerode theorem [Hopcroft79] p.67.

equivalent just in case any possible string of input symbols that leads to an accepting state from the one leads to an accepting state from the other, and conversely. Notice that the language accepted by an automaton is unchanged if equivalent states are identified.

Consider an arbitrary, representative sample of a minimal, DFS automaton. Since the acceptance of this set exercises every live transition in the machine, every live state in the machine (e.g. every state that lies in some accepting path) can be reached by some prefix of a string in the sample. So we may think of the prefixes of strings in the sample as names for states in the minimal automaton we seek, and we may be sure that every live state of the target automaton has such a name.

Since we have redundant names for all the states in the minimal automaton we are seeking, all we need to do is to determine which names denote the same state in this automaton. One way to do this is to assume that two strings designate the same state until we find a string that proves they are inequivalent. That is, strings σ and τ are assumed to name distinct states if there is a string γ such that the oracle says $\sigma\gamma$ is in the target language and $\tau\gamma$ is not, or *vice versa*. At first, one might despair that we could ever be sure that there is no such discriminating extension for two state names. After all, there are infinitely many possible extensions to consider. But not every problem that appears to be inductive actually is.

For technical reasons, the first thing we do is to proliferate state names by extending each of our given names with every possible input symbol. So for example, if the string 'aaba' is a state name, then we create the new names 'aabab', 'aabaa', 'aabac', and so forth, for each input symbol. For ease of reference, call the original set the *old set* and call the augmented set the *new set*. Each name in the new set is assigned a drawer into which we place strings that result in well-formed strings when tacked onto the name. For example, if the string $\sigma\tau$ is well-formed, then the string τ may be placed in the drawer of the string σ.

To begin with, we query the oracle about each result of extending a state name with the empty string. If the answer is yes, the empty string is placed in the appropriate state name's drawer. Otherwise, nothing is placed in the drawer. If the extension of a state's name with the empty string is well formed, then the state's name is well formed, and so a path ending at the state so named is an accepting path. Therefore the state so named is an accepting state. On the other hand, if the empty extension of a name is ill-formed, the state so named cannot be an

accepting state. So after this step is completed, the state names are distinguished into those that are accepting states and those that are not. Accepting states have the empty string in their drawers, and non-accepting states do not.

The next problem is to find an arbitrary string that cuts a new distinction among state names by extending some of them to well-formed strings and others to ill-formed strings. The trick is to look for two old names σ and τ and some input symbol b such that the drawers of σ and τ have identical contents but the drawers of σb and τb have distinct contents. (Recall that if σ and τ are old names then σb and τb must be in the new set. That's why we added the new names). Notice that if γ is a string in σb's drawer that is not in τb's drawer, then bγ must be in σ's drawer but not in τ's drawer. Since the contents of the drawers of σ and τ were identical, we have found a way to distinguish two previously undistinguished names, so we have split an equivalence class of names.

All that remains is to find out which other names in the new set should also have bγ added to their drawers. This can be done, as before, by extending each name with the string bγ and then asking the oracle. Once this is done, we find another distinguishing string, and so forth, until there is no pair of state names with identical drawer contents that have successor states with distinct drawer contents. At this point, all inequivalent states have been distinguished.[38]

An automaton equivalent to the minimal acceptor for the language is constructed from the final contents of the drawers in the following manner. Two state names are equivalent if their drawers have the same contents when the algorithm halts. The states of the conjectured automaton are just the equivalence classes of state names. A state is a halting state if one of its names has the empty string in its drawer, for this indicates that some accepting path terminates at this state. The initial state is the (unique) state named by the empty string, for it is the only state reachable by the empty string.[39] Finally, there is a transition labeled b from one state to another just in case some name of the latter state results from clamping b onto the end of some name for the former state.

---

[38] Actually, I have simplified Angluin's procedure somewhat by not accounting for the possibility that the target automaton has a dead state. The algorithm can easily be modified to drag a dead state name along, but the added tedium would have served no useful purpose in this discussion.

[39] The empty string must be a state name, for recall that the old names are just initial segments of the strings in a representative set. The empty string is a prefix of any string, so it is an old name.

3.3.3. A Comparative Assessment

The procedures of Pao and Angluin may both be thought of as searching the space of all possible partitions of a set of state names implicit in a given, representative set of strings. The partitions of a given, finite set are partially ordered by the relation of *refinement*. One partition is a *refinement* of another just in case every cell of the former is included in some cell of the latter. If one partition is a refinement of another, then it is *more refined* or *less coarse* than the latter. As it turns out, the set of partitions of a finite set form a lattice under the refinement relation (c.f. chapter seven).

Recall that Pao's procedure considers every pair of partitions of the states in the *ad hoc* automaton. It is known that the number of partitions of a given set is an *exponential* function of the set partitioned. As we have seen, this explosive relationship means that Pao's procedure must consider over one hundred ninety million partitions when the sum of the lengths of the input strings is 14.

Angluin's procedure, on the other hand, starts with the *coarsest* possible partition of states and refines it, successively, until the partition corresponding to the target automaton is found. Since the procedure never backs up, it considers at worst just one path from the top of the partition lattice to the bottom. That is, the number of possible partitions considered by Angluin's procedure is at worst *polynomial* in the sum of the lengths of the strings in the given, representative sample. The accompanying diagram underscores the dramatic degree to which Angluin's procedure ignores hypotheses that are considered by Pao's method.

Notice that Pao's procedure still seems to employ a "generate and test" strategy, while it is tempting to describe Angluin's procedure as *constructing* its hypotheses from the evidence. To a large extent, this intuition reflects the relative abilities of the two procedures to ignore hypotheses on the basis of the input evidence with no loss in inductive generality.

3.4. Horning's System for Inferring Context-Free Grammars

3.4.1. Horning's Context-Free Grammatical Inference Problem

The hypothesis language of this problem is the set of all *stochastic, unambiguous, context-free grammars*. The context-free grammars were defined in the previous chapter. Recall that they can be represented as finite sets of *productions*, each of

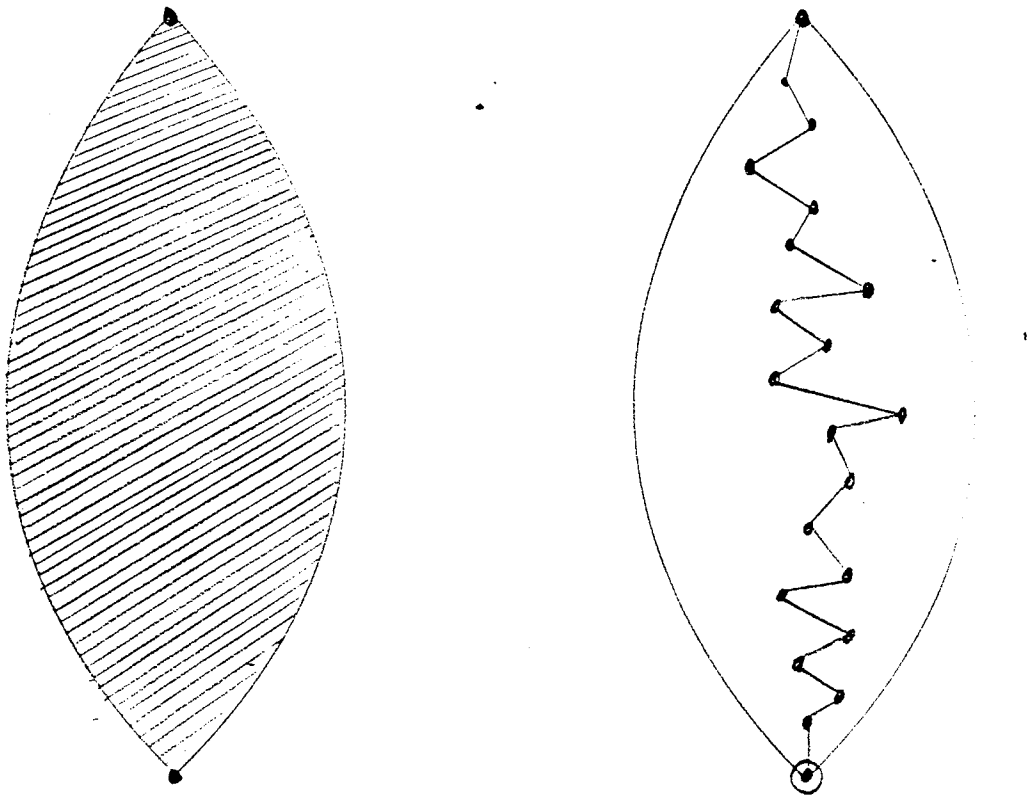


Figure 3-4: Worst Case Searches of the Procedures of Pao and Angluin

which consists of an upper case letter called a *nonterminal symbol* on the left, separated by an arrow from an arbitrary, finite string of nonterminal symbols and *terminal symbols* (i.e. lower-case letters).

Unambiguous, stochastic grammars were not discussed previously. A derivation is *canonical* if at each step in the derivation, the first occurring non-terminal symbol is the one to which a production rule is applied. A context-free grammar is *unambiguous* just in case no string has two distinct, canonical derivations with respect to this grammar. Ambiguity is an undecidable property over context-free grammars [Hopcroft79].

A *stochastic grammar* is an unambiguous, *context-free* grammar whose rules are labeled with rational numbers so that the numbers associated with rules that have the same non-terminal on the left-hand side add up to unity. A stochastic, unambiguous, context-free grammar specifies a unique probability for each string derivable from it in the following manner. Since the grammar is unambiguous, any derivable string has at most one canonical derivation. Recall that in a canonical derivation, we must apply some rule to the left-most non-terminal symbol occurring in the previous line. Hence, we can think of each canonical derivation as a finite

sequence of rule selections, where the number attached to a rule is its probability of being selected from the set of all rules with the appropriate non-terminal symbol on the left-hand side. By replacing each line of the derivation with the probability of selecting the rule applied to produce that line, we obtain a finite sequence of selection probabilities. The probability of the string so derived is defined as the product of the probabilities in this sequence. The probability of a string reflects, therefore, the probability of the sequence of rule selections that results in the construction of its canonical derivation (under the assumption that the probability of selecting a rule at a later stage of the derivation is independent of all previous choices).

Finally, Horning demonstrates that there are effective probability distributions over the context-free grammars, and he assumes that one of these measures represents *a priori* preference among possible hypotheses. This preference may reflect syntactic complexity, for example.

The proposed scope of Horning's problem is the set of all stochastic, unambiguous, context-free, languages. A language is *context-free* just in case there is a context-free grammar from which all and only the strings of the language are derivable. A context-free language is *unambiguous* if it is generated by some unambiguous context-free grammar, and is *ambiguous* otherwise. A natural question is whether ambiguous context-free languages exist. As a matter of fact, they do [Hopcroft79], p. 99, so Horning's restriction to unambiguous grammars is a material one rather than being a mere notational restriction. Finally, a stochastic language comes equipped with a probability distribution that can be specified by a stochastic, unambiguous, context-free grammar.

The evidence true of a language is restricted to positive examples of strings in the language. No negative examples are provided. There is no oracle, so the inductive method cannot perform experiments, but the process that provides the method with evidence is stochastic. In particular, the process can be viewed as an infinite sequence of independent selections of sentences of the language according to the language's probability distribution.

Horning assumes a stochastic criterion of identification. A method *converges* to a grammar just in case for every ϵ there is a δ such that for every $k > \delta$ the probability (with respect to the sampling distribution of the target language) of the set of all sample sequences of length k for which the method does not conjecture G is less than ϵ. That is, the probability of conjecturing G can be made as great as

you please by providing enough evidence. The *best* stochastic, unambiguous, context-free grammar for the target language is just the *a priori* most preferable one that specifies the target language's actual distribution. Finally, a method *identifies* a stochastic, unambiguous, context-free language just in case it converges to the *best* grammar for the language.

Like the problems addressed by Pao and Angluin, Horning's problem demands something like the invention of theoretical concepts for its solution. But now, the learning device cannot afford to assume a finite bound on the number of theoretical predicates it must introduce. This difference reflects the fact that the space of possible worlds facing Horning's procedure is much richer and more interesting than the space of worlds to be distinguished by the procedures of Pao and Angluin. While regular sets find few practical applications, most actual computer programming languages are context-free. The task of inferring the grammar of a programming language from examples of programs is not trivial at all, as an inspection of the sophisticated grammar of ALGOL will attest. It is also interesting that the *a priori* distribution over hypotheses will favor some combinations of theoretical concepts over others. So Horning's setting provides at least an attempt to show how one might systematically select among different theoretical approaches to the data.

3.4.2. Horning's Enumeration Method

It is no accident that Horning's problem is set up perfectly for the application of Bayesian conditionalization. Bayes' theorem states that the probability of an hypothesis on given evidence is just the *a priori* probability of the hypothesis times the likelihood of the evidence on the hypothesis, all divided by the probability of the evidence. The likelihood of a string with respect to a stochastic grammar can be calculated by finding the canonical derivation of the string from the grammar, associating the appropriate numbers with the lines of the distribution, and multiplying the sequence of numbers that results. And we have assumed that the *a priori* distribution is effective, so there is some procedure that computes it. What is not available is the probability of the evidence, but since Horning's technique maximizes posterior probability over fixed evidence at each stage, the denominator of Bayes' formula need not be computed.

It was assumed that the hypothesis language is effectively enumerable in descending order of *a priori* probability, so his method can employ some such enumeration procedure. On a given sample, Horning's procedure looks for the first

grammar in this enumeration for which the likelihood of the current evidence sequence is non-zero. Let this grammar be G_n, the nth grammar in the enumeration. Next, the method calculates the value of $f(G_n,E)=P(G_n)P(E|G_n)$ with respect to the current evidence sequence E. The procedure then runs through its enumeration until it reaches an hypothesis whose *a priori* probability is less than the product just calculated. Let the position of this hypothesis be n'. Finally, it conjectures the grammar occurring between positions n and n' in the enumeration whose value of f is greatest with respect to the current evidence.

The method's conjecture must be the most probable hypothesis with respect to the evidence, for no hypothesis before position n has a posterior probability greater than zero, and every hypothesis occurring after position n' has an *a priori* probability that is less than $f(G_n,E)$, so it cannot have a greater value of f with respect to E than G_n does.[40] Since the denominator of Bayes rule is fixed throughout this comparison, no hypothesis beyond n' in the enumeration can have a higher posterior probability on the evidence than G_n does. Finally, Horning shows that his method identifies every stochastic, unambiguous, context-free grammar in the sense defined [p. 80].

3.4.3. Bayesians who Consider Too Many Hypotheses

I have presented Horning's project in a detailed way because it illustrates that *computational* problems of control and hypothesis consideration do not disappear in a Bayesian setting. This point became vivid to Horning when he implemented his enumeration method on a real computer to see how it would perform in practical terms. It should come as no surprise that it didn't work very well at all.

> The initial portion of the enumerative...procedure selected for implementation was the enumeration itself. Although it was straightforward to write a program which enumerated the grammars in a given form, it soon became apparent that the enumeration process represented a serious problem. The first program quickly consumed the available memory for list structures.... pp 120-121.

Clearly, the considered segment of the enumeration had to be whittled down in light of the evidence if the program was to get anywhere.

> ...[F]ormally, there is no need to augment the inductive procedure with a deductive procedure: all grammars which can be ruled out deductively are automatically rejected by Bayes' theorem. In practice, however, there may be substantial advantage to a procedure that eliminates as many grammars as it can deductively, using the ...[Bayesian] procedure only to discriminate among DA [unrefuted] grammars.

---

[40] The product of two fractions less than one is less than either factor.

> The need for deductive preprocessing arises from the large number of grammars with similar complexities. [Horning takes complexity as the reciprocal of initial, *a priori* probability value.] * * * Before the correct grammar can be guessed, the procedure must at least have considered all other grammars of equal or lower complexity. We can use the number of such grammars as a lower bound on the number considered in the inference. But this number grows exponentially with complexity.
>
> * * *
>
> Only the DA [unrefuted] grammars actually contribute in any way to the solution; the others merely absorb computation, and (ideally) should be rejected as soon as possible. Pp. 86–6.

3.4.4. Paring Down the Hypothesis Enumeration a Priori

Horning distinguishes clearly between techniques that pare down the enumeration *a priori* and those that do so *a posteriori*, or as a function of the current evidence [p. 94]. Hypotheses are ignored *a priori* through the selection of a *normal* hypothesis language (i.e. a subset of the hypothesis language such that every hypothesis is equivalent to some normal hypothesis).

Sometimes, every normal form equivalent to a given hypothesis is more complex than this hypothesis. So the best normal hypothesis for a language may be far worse than the best hypothesis for the language. But the problem is to attain an arbitrarily high probability of conjecturing a *best* hypothesis for the target language. So to examine only normal hypotheses of this sort is to fail to address the problem at hand [p. 90].

But it is possible to define a normal form sublanguage such that every hypothesis has a normal form of equal or smaller complexity. In particular, there are many ways to write down a grammar that make no difference to the language it generates. A grammar's production rules can be listed in various orders, for example Enforcing a lexical order on the productions eliminates only equivalent grammars of the same length. Also, any grammar that has an "erasing" production (i.e. a production with the empty string on the right-hand side that has the effect of "erasing" a nonterminal symbol in the previous derivation line) can be replaced with an equivalent, shorter grammar. Hence, all grammars with erasing productions are excluded.

Now, consider the result of substituting unique non-terminals for unique non-terminals in a given grammar. The resulting grammar must generate the same language and is therefore equivalent to the original one. Call any grammar that

results from another grammar by a 1-1 substitution of non-terminals for non-terminals a *renaming variant* of the given grammar. Renaming variants are all identical in length (by symbol count) and equivalent to one another. Therefore as many should be eliminated from consideration as possible. Horning assumes an enumeration of the the alphabet from which non-terminal symbols for grammars are selected. The first symbol to occur in this enumeration is the *start symbol* S.[41] A normal grammar is then required to mention the start symbol as well as every predecessor of any mentioned non-terminal. This expedient eliminates many, but not all, renaming variants. In chapter six, I present an efficient technique for selecting a *unique* representative of each renaming-equivalence class that can be adapted to Horning's grammatical context.

3.4.5. Eliminating Hypotheses a Posteriori

Next, Horning considers techniques for ignoring hypotheses *a posteriori*, or in light of the evidence received. If G is a grammar, let G[B/A] denote the result of substituting the non-terminal A for each occurrence of B in G. The language generated by G must be a subset of the language generated by G[B/A].[42] G is said to *cover* G' just in case there are non-terminals A,B such that G'[B/A]=G. So if G covers G' and G fails to generate a string in the evidence sample, G must fail to as well. Therefore, there is no point in considering any grammar covered by a grammar that fails to generate a string in the sample.

Assume an enumeration of the non-terminal symbols. A substitution is said to be *canonical* for grammar G just in case it replaces each occurrence of the greatest non-terminal symbol occurring in G with the *start symbol* S. Horning defines the *canonical splits* of a normal grammar G to be the set of all normal grammars that are transformable into G by means of a single canonical substitution. There are at most finitely many canonical splits for any grammar. Every normal grammar with N distinct non-terminals is the canonical split of exactly one normal grammar with N-1 terminals. This organizes the hypothesis space into a finite set of trees in which each daughter of a node substitutes (via a canonical substitution) into the node of which it is a daughter.

---

[41] Recall that the start symbol is the unique non-terminal that constitutes the first line of every derivation.

[42] Let D be a derivation from the grammar G. Let D[B/A] be the derivation D with each string σ' in D replaced by σ'[B/A], and with redundant strings removed. If derivation line σ_i followed from σ_{i-1} in D in one step from a production in G, then line $g(s)_i$[B/A] follows from σ_{i-1}[B/A] in D[B/A] in one step from a production in G[B/A]. So any sentence derivable from G is derivable from G[B/A].

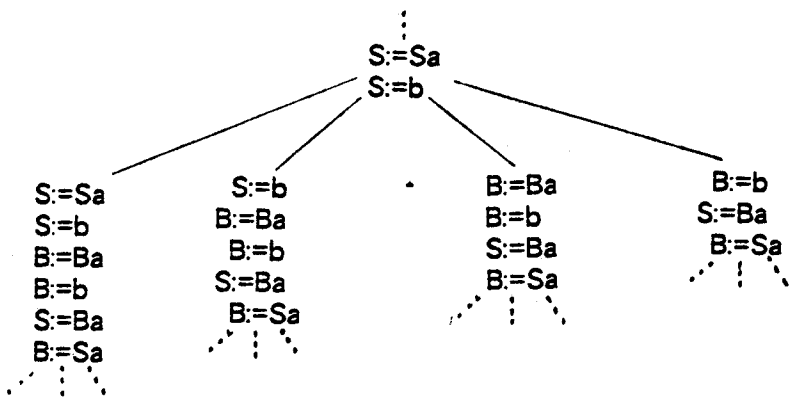


Figure 3-5: The Splitting Tree

Every canonical split of a grammar is covered by that grammar. Therefore, if a grammar fails to generate a string in the evidence, all of its canonical splits must as well. Hence, as soon as a grammar fails to generate a string, we can just refuse to split it any further, thereby ignoring all its inadequate daughters in one, simple act. Notice that this expedient accomplishes more than just withholding these daughters from test. They are not generated, represented, or considered in any serious sense.

But we might like to do better. Notice that not every normal grammar covered by a given grammar is a canonical split of this grammar. For example, consider the following grammars:

G1:
S::=A
A::=Aa
A::=B
B::=Bb
B::=b

G2:
S::=S
S::=Sa
S::=A
A::=Ab
A::=b

G3:
S::=A
A::=Aa
A::=S
S::=Sb
S::=b

All three of these grammars are normal. Grammar G1 is covered by grammars G2 and G3, since G1[A/S][B/A]=G2 and G1[B/S]=G3. But although G1 is a canonical split of G3, it is not a canonical split of G2, for [A/S][B/A] is not a canonical substitution. Now consider the evidence string 'baba'. This string is derivable from

G3, but not from G1 or G2. Ideally, we would like a procedure that would ignore G1 as soon as G2 is tested and refuted. But since G1 is not a canonical split of G2, Horning's procedure will test it in this situation.

In light of this discussion, an obvious design objective is to find a procedure that ignores every normal hypothesis *covered* by a failed hypothesis. The question whether there are such procedures is a crucial issue in the logic of discovery. In chapter seven, I address this question again in the context of a different inductive problem.

3.5. Ehud Shapiro's Model Inference Systems

3.5.1. Model Inference Problems

The problems addressed by Shapiro's implemented model inference systems are all special cases of the following problem. The hypothesis language is the set of all finite sets of *clauses* on some first-order, non-logical vocabulary. A clause is just a universally quantified disjunction of atomic formulae or their negations. Possible worlds are relational structures for the hypothesis language whose individuals are all denoted by closed terms of the hypothesis language (c.f. note 19, Chapter 2 above). The evidence language is the set of all atomic sentences or their negations over the vocabulary of the hypothesis language.

An hypothesis is adequate for a structure just in case it is true in the target structure and it entails each evidence sentence true in this structure. But given Shapiro's special choice of languages and worlds, this criterion of adequacy is equivalent to the requirement that the hypothesis entail all evidence sentences true in the target structure and no evidence sentence false in this structure. Finally, the assumed criterion of identification is EX^*-identification. That is, an inference device identifies a world just in case for every complete evidence presentation in this world, there is an adequate hypothesis that the device conjectures all but finitely many times.

3.5.2. Resolution Theorem Proving

Shapiro's work can be thought of as an application of *resolution* proof theory to Blum's general techniques for the study of inductive inference problems. The following presentation of the resolution method follows [Robinson65] (but employs more standard mathematical notation).

Let L be an arbitrary first-order language and let V be the non-logical vocabulary of L. To computer scientists, a clause is represented as a *set* of atoms or negated atoms of L, but this set is interpreted as tbe universally quantified disjunction of its elements. A *substitution* θ is just a function whose domain is the variables of V and whose range is a subset of the terms constructible from V. The application of substitution θ to a string σ of L is written $\sigma\theta$, and denotes the result of replacing each variable x occurring in σ by the corresponding term $\theta(x)$. Finally, if S is a set of strings, then $S\theta$ denotes $\{\sigma\theta: \sigma \epsilon S\}$.

A substitution θ is a *unifier* for a set S of atoms or their negations just if $S\theta$ is a singléton. A *most general* unifier θ of S is a unifier of S such that for any other unifier λ of S, there is a substitution γ such that $S\lambda = [[S\theta]\gamma]$. That is, θ is a most general unifier of a set of atoms if the unique element of $S\theta$ is as logically general as possible, short of preventing the unification. For example, the set {P(y,x), P(f(g(x,y),z)} is unified by substituting 'f(g(x,y),z)' for 'y' and 'z' for 'x'. Since both substitutions are necessary to unify the set, this substitution is also most general.

If C is a clause, then C^+ is the set of all atoms in C and C^- is the set of all atoms whose negations are elements of C. E is a *resolvent* of the pair L,R just if there is a subset L' of L^- and a subset R' of R^+ such that (1) there is a maximal unifier θ for L' U R' such that $L'\theta=R'\theta=\{a\}$, where a is some atom, and (2) $E=[(L-L') \cup (R-R')]\theta$. The atom a is called the *atom resolved upon*. Resolution is any procedure that produces a resolvent from two clauses.

For example, let L be the clause {-P(x,f(y)),-Q(z)} and let R be the clause {P(a,z),-Q(f(g(y)))}. We choose L' as {P(x,f(y))} and R' as {P(a,z)}. A maximal unifier of L'UR' is the substitution of a for x and of f(y) for z. The atom resolved upon is therefore P(a,f(y)). The resolvent of L and R is is the clause {-Q(f(y)),-Q(f(g(y)))}.

Notice that E may be a resolvent of the pair L,R without being a resolvent of the pair R,L, for the order of the given clauses determines the one from which we select negated as opposed to non-negated atoms in seeking unification. If E is a resolvent of L,R, then L is the *left component* of the resolution and R is the *right component* [Shapiro81]. Notice, also, that negated atoms from the left component are unified with non-negated atoms from the right component.

If S is a set of clauses, $R^0(S)$ is just S, and $R^n(S)$ is the set resulting from all possible applications of resolution to the elements of $R^{n-1}(S)$ along with all the elements of $R^{n-1}(S)$. Resolution provides a sound, complete test of inconsistency for

finite sets of clauses in the following sense: There is an n such that the empty clause is an element of $R^n(S)$ exactly if S is a set of clauses that is not satisfiable [Robinson65].

Any first-order sentence can be converted into a finite set of clauses by (a) putting the sentence into prenex normal form, (b) putting the propositional matrix of the result into conjunctive normal form, (c) skolemizing all the existential quantifiers in the prefix, and (d) taking each conjunct of the result to be a clause. The result of this translation is satisfiable exactly if the original formula was. So if the empty clause is derivable from the translation by resolution, the original formula was not satisfiable. Hence, resolution, with the translation rules just mentioned tacked on the front, is a sound, complete system for first-order validity. Since the translation process is effective, there can be no way to decide whether a given finite set of clauses will lead to the empty clause by resolution, or else first-order validity would be generally decidable.[43]

Intuitively, a single resolution application is a sequence of universal specifications and disjunctive syllogisms, all rolled into one. Universal specification takes care of quantifiers, and disjunctive syllogism takes care of propositional maneuvers. But some sequences of universal specifications could amount to a substitution that is not a maximal unifier with respect to the atoms unified and eliminated through disjunctive syllogism. The completeness of resolution shows that these sequences of applications are not missed. So in a sense, the point of resolution is to avoid considering irrelevant universal specifications in proofs just as intelligent inductive inference systems are supposed to avoid considering refuted hypotheses, or chess players are supposed to ignore pointless moves.

3.5.3. The General Idea

Since the problem for Shapiro is to provide an hypothesis from which *all* and *only* the true evidence is derivable, an hypothesis can be either too weak or too strong for given evidence. That is, it can either entail too many evidence sentences or too few. If it is discovered to be too strong it should be weakened, and if it is discovered to be too weak, it should be strengthened. Shapiro's system begins with an outrageously strong hypothesis (a contradiction) and incrementally reads given evidence strings. As each string is read, the current conjecture is tested for sufficient strength and weakness with respect to the evidence read.

---

[43] Even if the clauses are restricted to at most one negated atom and one non-negated atom each, the satisfiability problem for finite sets of clauses is still undecidable (by a reduction of the Post-correspondence problem) [Reynolds70].

Since the resolution proof system cannot always halt with an answer, an arbitrary, effective resource bound f, which is a function of the evidence sentence to be checked, is imposed on the test. The resource measure is the number of resolution steps required to reach the empty clause. So if no proof of evidence sentence e from hypothesis h can be found in f(e) steps, the effort is abandoned and it is "assumed" that e is not derivable from h. Hence, for any bound f, there is a notion of "too strong" and "too weak" relative to f. Hypothesis h is *too f-strong* if the negation −e of a true evidence sentence e is derivable in f(−e) steps. h is *too f-weak* if some true evidence sentence e is not derivable in f(e) steps.

Relying on f−strength and f−weakness clearly introduces some risk. But there are at least some structures in which no such risk is incurred. An hypothesis is f−easy for a structure just if each evidence sentence true in the structure can be derived in no more than f(e) steps from this hypothesis.[44] A structure is said to be f−easy just in case it has an f−easy hypothesis. Trivially, f−strength and f−weakness are completely reliable indicators of strength and weakness for f−easy structures.

Given f, an hypothesis is strengthened if it is too f−weak, and it is weakened if it is too f−strong. The procedure for strengthening an overly weak hypothesis involves a *refinement operator*, and the procedure for weakening an overly strong hypothesis is called the *contradiction backtracing algorithm*. I review each of these ideas in turn.

3.5.4. The Contradiction Backtracing Algorithm

Assume that a relational structure **M** is under investigation and that a finite set of clauses h is too strong for the given evidence. Then some clause C in h must be false in **M**. The problem is to find out which clause in h is false without working too hard. Since Shapiro employs resolution to test the consistency of h with the evidence, if it is found that h is too strong, then a resolution derivation of the empty clause must have been constructed from h and the evidence. A resolution derivation is a labeled binary tree rooted at the empty clause, such that each vertex that is not a leaf is labeled by a resolvent of the labels of its daughters. By convention, the right component R of the resolvent C labeling vertex v labels the right−hand daughter of v, and the left component L of C labels the left−hand daughter of v. The leaves of the tree are labeled with elements of H and the negations of evidence sentences received as inputs.

---

[44] Shapiro requires such a derivation for "all but finitely many" true evidence sentences, but no generality is lost in the given definition, for all the finitely many exceptions can be added to h and then are derivable in one step.

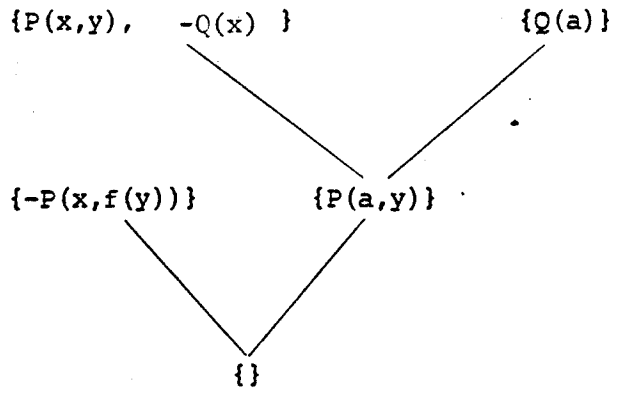


Figure 3-6: A Resolution Proof

Shapiro's backtracing algorithm receives such a resolution proof as input and has access to an oracle over the evidence language that specifies whether any evidence sentence is true in the structure under investigation. The oracle is a "laboratory" in the world under investigation, as it were. The procedure constructs a single path through the tree, beginning at the tree's root. At the kth node visited, the procedure has immediate access to

1. the clause C_k that labels k,
2. the atom P_k resolved upon to obtain C_k from its left and right components L_k and R_k in the tree.
3. the substitution λ_k that is the maximal unifier leading to C_k from L_k and R_k, and
4. a substitution θ_k that it has "built up" along the path already traversed in a manner that will be clear momentarily.

The procedure first finds an arbitrary substitution γ so that $P\theta_k\gamma$ is a closed atom. Then it queries the oracle to see whether or not $P\theta_k\gamma$ is true in the structure under investigation (i.e. it "does an experiment in the lab"). Next, it sets θ_{k+1} to the composition $\theta\theta_k\gamma$. If $P\theta_k\gamma$ is true, then the algorithm moves to the node labeled by L and sets k to k+1. Otherwise, it moves to the node labeled by R and and sets k to k+1. When a leaf is reached, the leaf is output as a false hypothesis.

The algorithm's output is always false, and moreover, the conjunction of the atoms queried by stage k is always a counterinstance to the clause that labels the vertex occupied at k. By convention, an empty set of atoms is a counterinstance of the empty clause, and if a degenerate proof of the empty clause from itself is supplied, the procedure will output the empty clause. Suppose, then, that the atoms tested up to stage k form a counterexample to the label C_k of the vertex visited at stage k. But a counterexample to C_k is "almost" a counterexample to to L [R] in the

sense that for every basic formula (atom or its negation) p in L [R] there is a basic formula p' in C_k entailed by p--- except for the elements of L [R] that are identical to [the negation of] the atom resolved upon to obtain C_k under the maximal unifier applied to obtain C_k. So if we carefully substitute the atom resolved upon consistently with all previous substitutions leading to the given counterexample of C_k and instantiate the result of this process to a closed atom, then adding this atom to the atoms forming the counterexample of C_k must either be a counterexample of L or a counterexample of R, depending on whether this atom is true or not in the structure under study. But this careful collection of prior substitutions is accomplished by the incrementally updated substitution θ_k in Shapiro's algorithm. Shapiro notes that the idea of collecting such substitutions was suggested by a proposal of Green [Green69] for answering questions from a logical data base.

Once again, the metaphor of ignoring alternatives is suggestive. The most obvious procedure to find a false clause in a set of clauses using an oracle for atomic sentences would be to exhaustively instantiate the clauses and compute their boolean valuations for each instantiation. Even given the resolution proof, a less elegant procedure might cross-up its instantiations as it moves down each path so that it would end up backtracking and searching through the resolution tree. By careful attention to the structure of the resolution proof, Shapiro's device cuts a single, decisive path to a false hypothesis, ignoring all mistaken substitutions.

3.5.5. Refinement Operators

Recall the main idea of Shapiro's system: hypotheses found to be too strong are made weaker, and hypotheses found to be too weak are made stronger. The procedure weakens an overly strong hypothesis h by applying the backtracing procedure and eliminating the false clause F found by this procedure from h. But the result h' of this coarse maneuver may well be too weak. After all, F may have carried most of the logical force of h. If, indeed, h' is too weak, Shapiro's idea is to "patch" h' by adding a clause F' properly entailed by F to h' to bolster its strength to something near that of h. F' is called a *refinement* of F, and the procedure for producing it is a *refinement operator*. h' U {F'}, in turn, may be either too weak or too strong. If it is too strong, another false clause is eliminated. If it is too weak, yet another clause F" properly entailed by F is added to h'. Hence, the system's conjecture oscillates between overly-strong, and overly- weak hypotheses of increasing length until an unrefuted "equilibrium" is reached. Then more evidence is read, and the reverberations due to its impact once again dampen to equilibrium until new evidence ceases to have any impact.

Let H be a set of clauses. Let m:clauses--->N, such that no more than finitely many hypotheses are assigned the same number. For a clause c, m(c) is called the *size* of c. For any set of clauses S, S[n] denotes $\{c \in S : m(c) \leq n\}$. For any c,c' in H, c is a *refinement* of c' just if $m(c) \geq m(c')$ and $c' \vdash c$. Function ρ is a *refinement operator* for H exactly if (1) ρ is a function from H to subsets of H, (2) each element of $\rho(c)$ is a refinement of c, and (3) for each k, the set $(\rho(c))[k]$ is computable. For any c,c' in H, $c \leq_\rho c'$ exactly if $c' \in \rho(c)$. Let $\rho^0(c)$ be $\{c\}$ and let $\rho^n(c)$ be $\rho^{n-1}(c) \cup \{c' \in H$: there is a $c'' \in \rho^{n-1}(c)$ such that $c' \in \rho(c'')\}$. Let $\rho^*(c)$ be the closure of $\rho^n(c)$. Refinement operator ρ is *complete* for H just if for every c in H there is a c' in $\rho^*(\#)=H$ such that c is logically equivalent to c' and # denotes the empty clause.[45]

To fix ideas, consider the *refinement graph* $(H, <_r)$. Its "bottom" is #., and as one proceeds from #, clauses become weaker and "bigger". This works because clauses do tend to get weaker as they get longer, for adding function symbols and disjuncts both weaken the clause they are added to. There are finitely many clauses at each level, and there are more clauses at each level as one moves further from #. Finally, there is no bound on the length of clauses, so the graph extends ever upward. A useful image is that of a cone-shaped network extending infinitely upward from the empty clause.[46]

Although the notion of the "size" of a sentence is not specified in the above definitions, Shapiro always employs the same function m such that m(c) = the number of occurrences of nonlogical vocabulary elements symbols in c less the number of distinct variables occurring in c. The latter requirement facilitates the construction of a refinement operator, for formulas must become longer roughly as they become logically weaker. For example, $m(\{P(x,x,f(y))\}) = 5-2 = 3$ but $m(\{Px,x,f(x)\}) = 5-1 = 4$. Note that the number of occurrences of variables in a formula is not the same as the number of distinct variables occurring in the formula.

Shapiro gives the following example of a refinement operator over clauses of cardinality of at most one:

- If p = {} then $\rho_1(p) = \{\{P(x_1,\ldots,x_n)\}$: P is an atom of arity n and $x_1,\ldots,x_n$ are the first n unique variables in an assumed ordering of the variables}

---

[45] Actually, Shapiro requires the stronger condition that $\rho^*(\#) = H$. But if he requires this, some of the functions he claims to be complete refinement operators are not.

[46] I am indebted to Keith Wright who suggested this metaphor in a seminar report on Shapiro's paper at the University of Pittsburgh.

- If p = {a}, where a is an atom, then $\rho_1(p)$ = {a[x/y]: such that x and y both occur in a} U {a[x/f(y_1,....,y_k)]: f is a function symbol, x occurs in a, and y_1,....,y_k are the first k variables not occurring in a}

So an initial segment of a path in the refinement graph of ρ is as follows:

Clause	Size
{}	0
{P(x_1,x_2,x_3}	4-3=1
{P(x_1,x_1,x_3}	4-2=2
{P(x_1,x_1,f(x_2,x_3))}	6-3=3
{P(x_3,x_3,f(x_2,x_3))}	6-2=4
.	.
.	.
.	.

Notice that by the definition of ρ_1, if $q \in \rho_1(p)$ then m(q)=1+m(p) and $p \models q$. Moreover, since the variables to be selected are bounded, $\rho(p)$ is finite and computable. So ρ_1 is a refinement operator. Finally, it is easy to see that a clause of at most unit cardinality that is logically equivalent to any other such clause can be obtained in some sequence of iterated refinements. Hence, ρ_1 is complete over clauses of at most unit cardinality.[47]

An obvious candidate for an operator complete for all clauses on a given vocabulary V would be as follows, where h is any clause.[48]

$\rho(h)$ =

1. {h U {P(x_1,....,x_n}: P is a predicate in V of arity n and x_1,....,x_n are the first n distinct variables in V not occurring in h} U
2. {h[x/f(x_1,....,x_n)]: x occurs in h, f is an n-ary function symbol in V, and x_1,....,x_n are the first n distinct variables not occurring in h} U
3. {h[x/y]: x,y both occur in h and |h| = |h[x/y]|}.

Notice that each element of $\rho(h)$ is exactly one unit longer than h. Case (1) adds an atom with new distinct variables. If the arity of the atom is n, then n+1 symbols are added, but n distinct variables are subtracted from this sum. The same can be said of the case in which a new term is substituted for a variable in case (2). Notice the requirement in case (3) that |h| = |h[x/y]|. This is imposed to prevent the following situation:

---

[47] This is not true for my definition of completeness, not Shapiro's. Shapiro requires that $\rho_1^*(\#)$ be identical to the hypothesis language. But unless the vocabulary has only finitely many variables, infinitely many variable renaming variants would be missing from $\rho_1^*(\#)$. And if They were included, the values of ρ_1 would not be finite sets, so ρ_1 would not be a refinement operator.

[48] Shapiro does not actually propose this simple operator. His own operator raises issues best treated later.

h={P(x,y,y), P(y,x,x)}

h[x/y] = {P(y,y,y)}

m(h) = 8−2 = 6

m(h[x/y]) = 4−1 = 3 < 6

If substitutions are permitted to identify atoms in h, then the size of an element of ρ(h) that is entailed by h is less than that of h, so ρ is not a refinement operator. But given the restriction that the result of a substitution must be at least as long as the clause to which the substitution is applied, each element of (3) is of size 1+m(h). For example, consider the following path segment from the refinement graph of ρ.

Clause	**Size**	
{}	0	
$\{P(x_1,x_2)\}$	1	(1)
$\{P(f(x_3),x_2\}$	4−2=2	(2)
$\{P(f(x_3),x_2,Q(x_1)\}$	6−3=3	(1)
$\{P(f(x_2),x_2,Q(x_1)\}$	6−2=4	(3)
.	.	
.	.	
.	.	

It is evident, then, that if $g \in \rho(h)$ then m(g) = m(h)+1 and h⊢g. So ρ is a refinement operator. It should also be evident that ρ is complete over the clauses on vocabulary V.[49] Shapiro does not adopt this simple, general refinement operator, but it is perfectly adequate for understanding the rest of the presentation of his algorithm.

Now consider how truth and falsity interact with the refinement graph. Let H(**M**) be the set of all clauses in the hypothesis vocabulary that are true in relational structure **M**. If ρ is complete over these clauses, then H(**M**) will be a set that is closed upward in the refinement graph.The remaining area in the cone is occupied by clauses false in **M**. Let I be an upward [downward] closed subset in the refinement graph. Clause c is *maximal* in I if no c' in I has the property that $c' <_\rho c$ [$c' >_\rho c$]. The *source* S(I) of a closed subset I of the refinement graph is the set of all maximal elements of I. The *boundary* B(I) of such a set is just the source of its complement.

So for example, the source in the refinement graph of the H-diagram of a structure for H can be pictured as a "surface" cutting the roughly conical graph. Infinitely many clauses must lie on this surface, for there must be a false clause of

---

[49] In fact, for any clause h, there is a clause h' that is a variable-renaming variant of h in $\rho^*(\#)$.

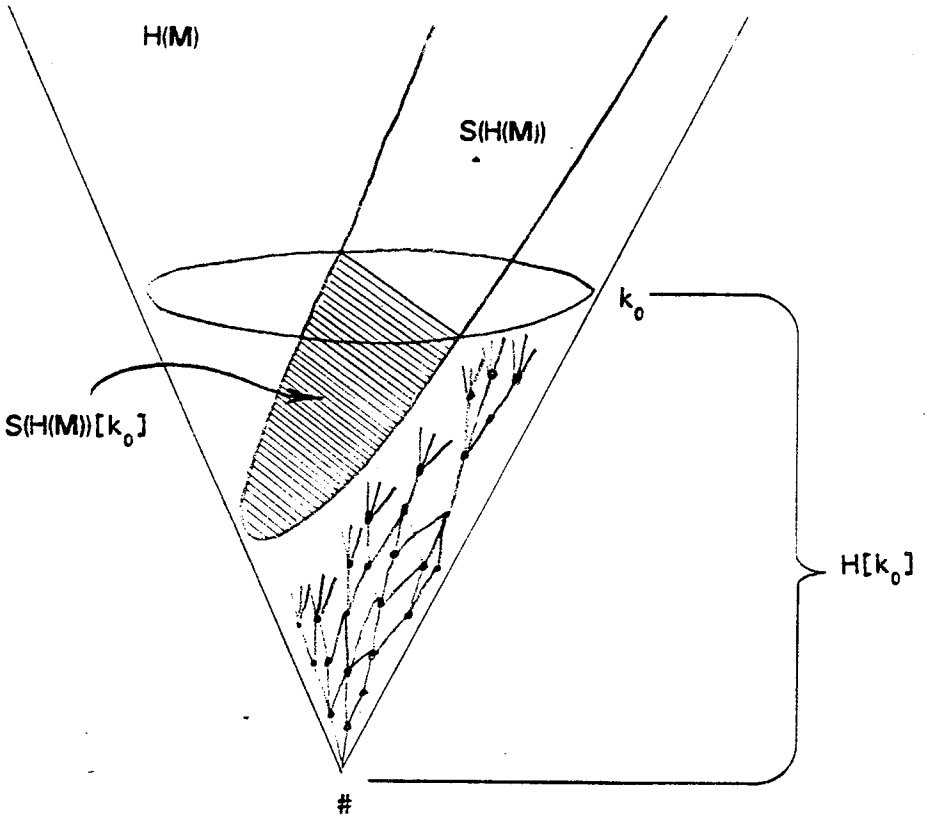


Figure 3-7: The Refinement Graph

each size for any structure. Pictorially, the boundary of the diagram of a structure might be a plane cutting the cone parallel to one of the cone's elements, so that it divides the interior of the cone into two infinite volumes. Clearly, if the refinement operator ρ is complete for H, the source of the H-diagram of any structure **M** for H is an (infinite) axiomatization of the complete H-theory for **M**. But notice that if some finite subset h of H[k] is a true, finitely axiomatizable, E-complete hypothesis h for **M** then the finite set S(H(**M**))[k] is also a true, E-complete hypothesis for **M**. Since h is true, it is above S(H(**M**)) in the refinement graph. Since h is in H[k], it is below the horizontal section of the graph delimiting size k. Hence it is "trapped" in the wedge between the horizontal plane representing size k and the surface representing S(H(**M**)). Since each clause in h must be descended from the empty clause, each clause in h is on a path that passes through a clause in S(H(**M**))[k]. The situation should be intuitive in light of the accompanying figure.

Shapiro requires one more property of refinement operators, but its purpose cannot be explained until the next subsection when his full inductive procedure is developed. Let h,h' be sets of clauses such that for each clause c in h, there is a clause c' in h' such that $c' \leq_\rho c$. Then $h \geq_\rho h'$(h is *above* h' in the refinement graph). Shapiro says that ρ is *conservative* for f just in case for each h, for each h' below

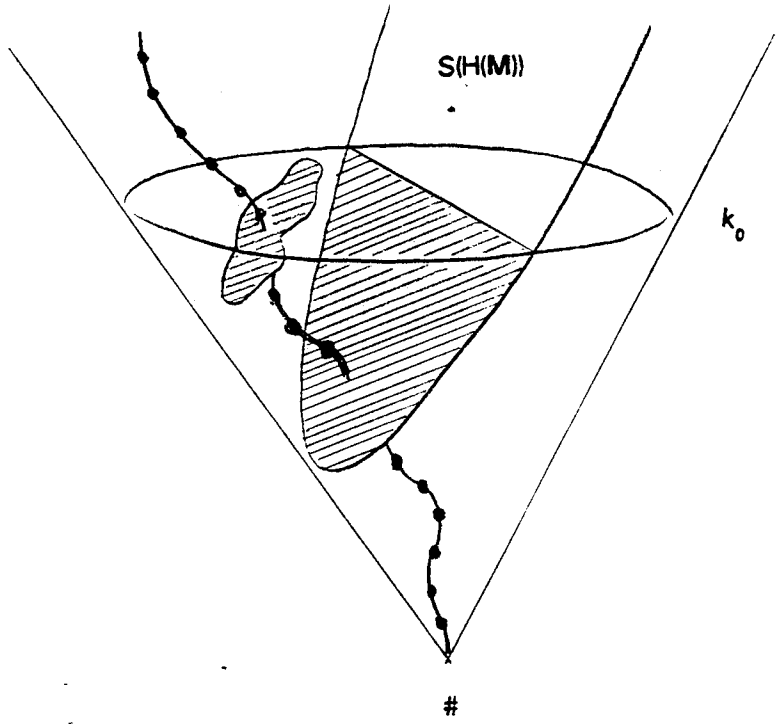


Figure 3-8: If h is a true, E–complete subset of H[k], then so is S(H(M))[k]

h in the refinement graph of ρ, and for each evidence sentence e, if h derives e in f(e) steps, h' must do so as well. That is, f–easiness must be closed downward in the refinement graph. This might seem unlikely, for one could imagine that it would require extra proof steps to "weaken" a strong hypothesis until it is as weak as one below it in the refinement ordering, after which the same proof would go through. But in the case of resolution theorem proving, substitutions of arbitrarily many variables can occur in "one step". So if a clause c *subsumes* another clause c' (i.e. if there is a q such that $c\theta$ is a subset of c) then no resolution proof involving c as premise requires more steps than a proof involving c' as premise, for this substitution can be incorporated into one of the steps of the proof involving c'. As it turns out, all the refinement operators employed by Shapiro as well as the general one are such that the refinement of a clause is subsumed by the clause it is a refinement of.

3.5.6. Shapiro's Algorithm

The operation of the algorithm is complicated, but it can be illuminated in terms of the geometrical cone and surface metaphor introduced earlier. Since Shapiro provides perfectly adequate proofs of his own, there is no need to be fussy about rigor here.

In the following discussion, it is assumed that H is a language of clauses, E is the set of basic sentences of H, ρ is a complete, conservative refinement operator for H, and **M** is a structure for H. The true evidence sentences form a scattered cloud in the H-diagram of **M**, for their terms may be of arbitrary complexity. The plane marked by k_0 is the source of $H[k_0]$ in the refinement graph. k_0 is the least k such that there is an adequate (i.e. E-complete, f-easy) subset of H[k] for **M**. The accompanying figure should assist in sorting out all the pieces.

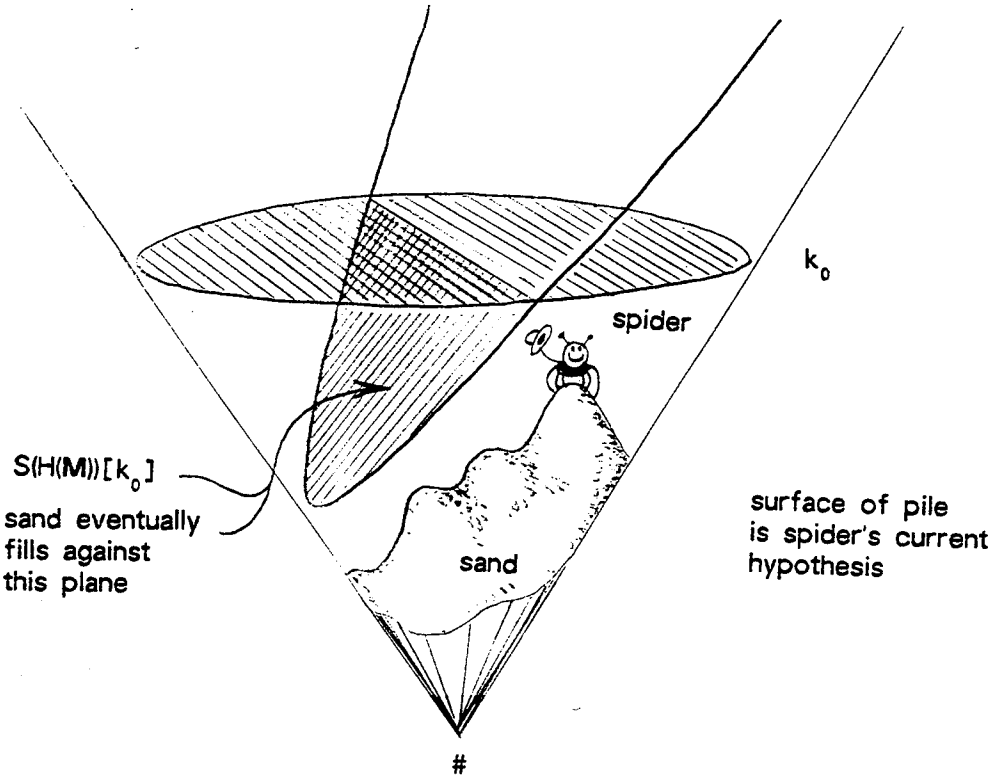


Figure 3-9: refinement graph and **M**

Imagine a blind spider armed with (i) a program that computes ρ, (ii) the resolution proof procedure, (iii) a program for f, and (iv) an arbitrary bound k on the size of hypotheses it should consider. This spider begins at the vertex of the refinement graph (i.e. at the empty clause) and a compatriot shouts evidence at it from the

sidelines. At stage n, the spider tests its current, finite set of clauses h(n) by testing each clause c in the set against the negations of all the basic sentences shouted to it thus far by its compatriot. This test is performed by running its resolution procedure for n steps. (Notice that the resource bound f is not used to test for refutation). If a contradiction arises, it removes the offending clause by means of the backtracing algorithm, places this clause into a set Refuted(n), and adds all the refinements of size less than or equal to k to its next conjecture h(n+1). That is, h(n+1) is always the boundary set of Refuted(n). Notice that once a clause of size k is refuted, part of the boundary of Refuted(n) is no longer in H[k] so the set h(n) will have a "hole" where Refuted(n) intersects the limit k.

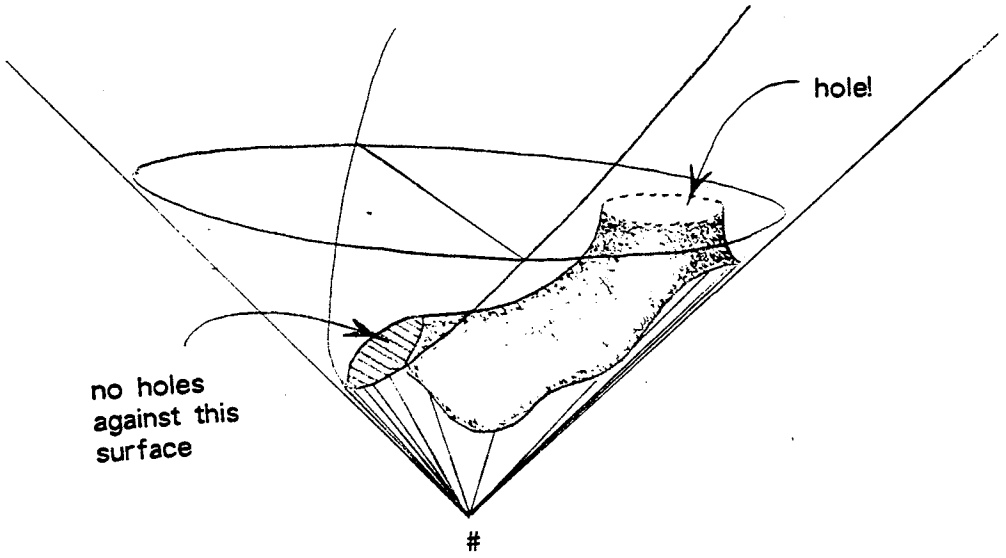


Figure 3-10: A Hole in an Hypothesis

This hole is crucial in the operation of Shapiro's procedure.

If we think of the spider as placing a grain of sand on each hypothesis in Refuted(n), it would appear as though the spider were involved in building a small heap of sand in the bottom of the cone in the figure. This pile can never violate the confines of the two planes cutting the cone, although the spider has no way of ever being sure whether it has reached the plane representing S(H(M)) or not, for it may not have evidence sufficient to refute all the clauses below this plane.

Consider the figure once more. Our spider is bound to be frustrated following the above procedure if his limit k is set lower than k_0, for the boundary of his pile of sand restricted to clauses of size k will never be an adequate hypothesis. He needs a way to raise the bound and patch the hole in his hypothesis (consult figure 3-11). As soon as this hole is patched, the hypothesis is much stronger, and may once again be false. So indiscriminate raising of the ceiling can lead the spider to

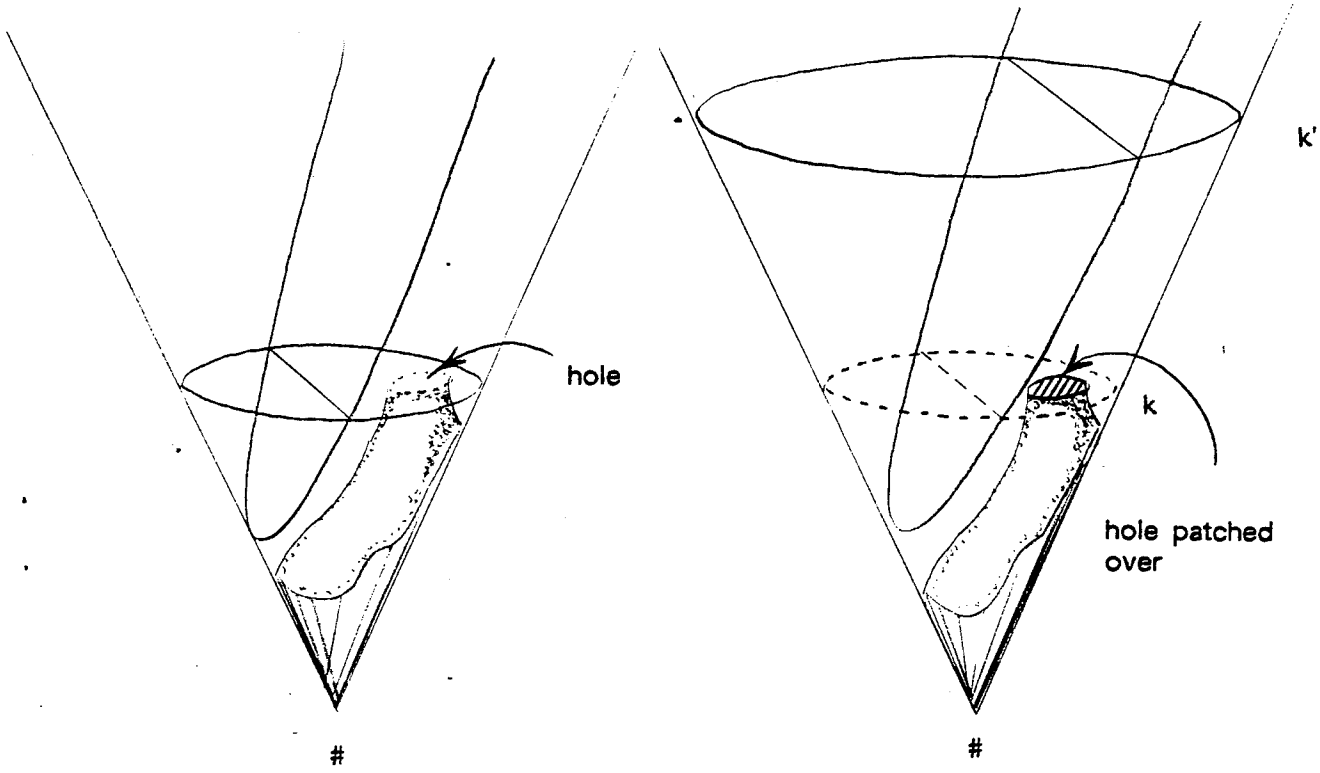


Figure 3-11: Patching Holes in an Hypothesis

conjecture an infinite sequence of adequate hypotheses interspersed with false ones. But we want the spider to *converge* to an adequate hypothesis. Shapiro's solution is to increment k and to patch the hole in h for which k is responsible only when no clause in h can be refuted in a proof of no more than n steps, and when some given evidence sentence e cannot be derived in f(e) steps from h. This is why the conservatism of ρ is crucial for Shapiro's approach. For if ρ is conservative, then any hypothesis h with no holes that is considered before $S(H(M))[k_0]$ must be h-easy if $S(H(M))[k_0]$ is. Hence, the ceiling k_0 will never be raised according to the condition just described. Any lower ceiling k will be raised, however, for the spider will build its sand-pile right up to the ceiling, and his hypothesis will be just S(H(M))[k], which by hypothesis, is not f-easy.

So finally, once the ceiling is bumped to k_0, the spider receives its nth bit of evidence, increments its bound on refutation testing, and adds refuted hypotheses to Refuted(n). Eventually (although the spider cannot tell when) its conjecture is exactly the finite surface S(H(M)), which is adequate. Thereafter, all the spider's time is spent in the futile task of incrementing the bound on the refutation test and reading more evidence, but its adequate conjecture will never change--- assuming that M is indeed f-easy. Otherwise, the spider will increment the ceiling forever and the

procedure will never converge. The property of converging only in worlds one identifies is called *reliability* in the computational literature, and is considered an epistemological asset. Shapiro nicely proves this property for his algorithm in formal detail, but a sequence of diagrams animating the procedure is worth a thousand henscratches.

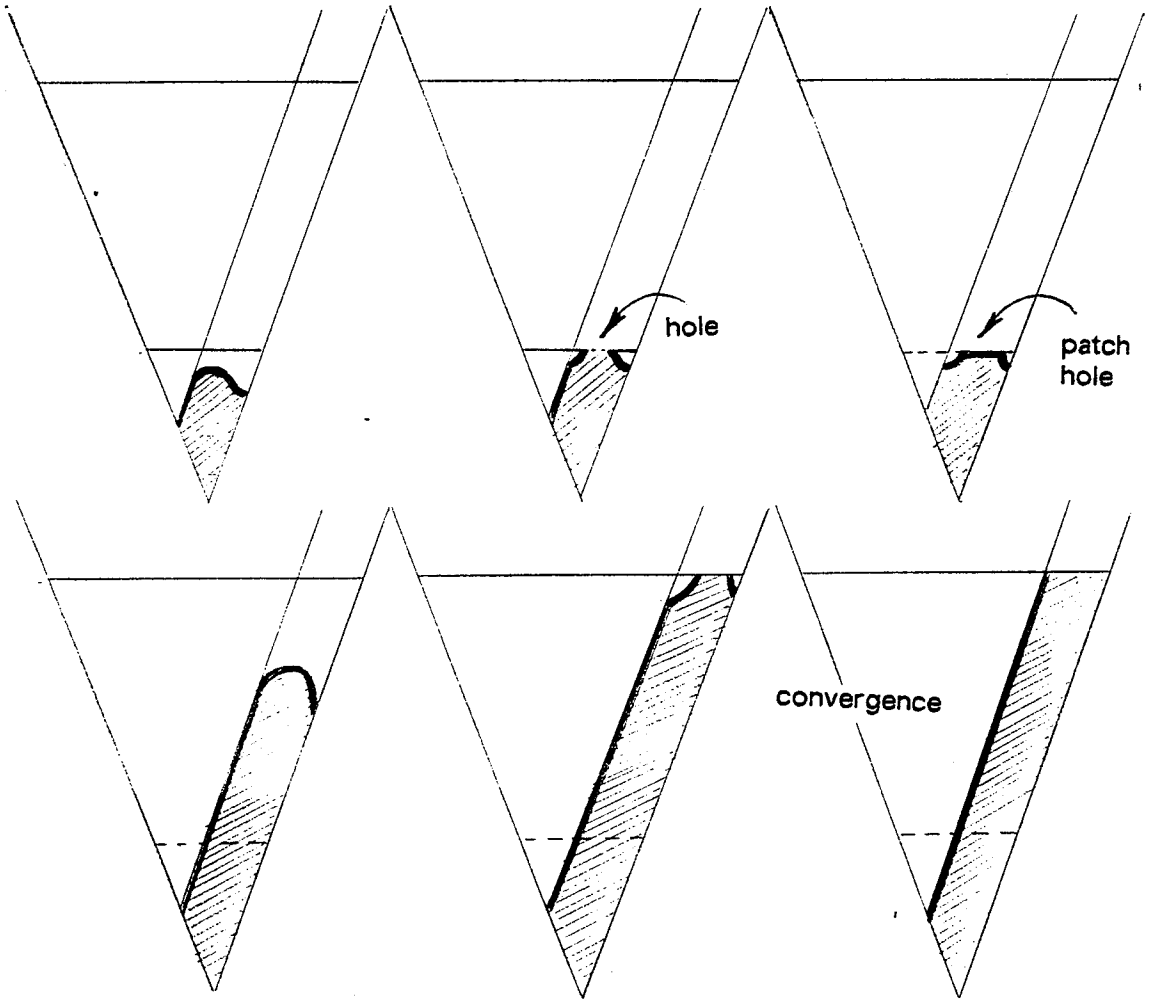


Figure 3-12: How the Model Inference System Converges

More formally, the algorithm is:

```
MIS(f):
SET k=0, Refute={#}, E={}, h=ρ(#);
REPEAT
        SET e=READ NEXT EVIDENCE, E=(E U {e});
        REPEAT
                Comment: Weakening Loop
                WHILE h |− the negation of some element
                        of E in no more than n steps DO
                        SET BACKTRACE(h,E)=culprit;
                        SET h = (h−{culprit}) U
                                        ρ(culprit);
                        SET Refute = Refute U
                                        {culprit};
                Comment: Strengthening Loop
                WHILE NOT h(n) |− some e in E(n)
                        in no more than f(e) steps DO
                        SET h = (h U
                                {g(r)(c): c∈Refute and m(c)=k+1};
                        SET k=k+1
        UNTIL neither WHILE condition is satisfied;
        OUTPUT h(n)
FOREVER
```

In the first WHILE loop, if h is too strong, then the spider removes all the clauses from h that are refuted in n steps on the evidence gathered so far by stage n, puts them onto its "sandpile" and adds all hypotheses below length k that are in the boundary of this pile into h. In the second WHILE loop, if h is too weak, then the spider bumps the bound from k to k+1 and patches the hole in h that was caused by the bound at k. If the current hypothesis is adequate, neither loop is entered, and the system simply reads more and more evidence.

3.5.7. Strengths of Shapiro's Proposal

Shapiro's method is well-presented, elegant, and metaphorically compelling. One can just picture an automated scientist gathering data from an inexhaustible supply of journals in the library, constantly strengthening and refining his current hypothesis as he reads, and querying nature directly in his laboratory when he notices his hypothesis is refuted. Moreover, for any f, H, and E, MIS(f) is a completely general solution to any model inference problem whose proposed scope includes only f-easy structures. Depending on the f chosen, MIS can address structures that border on mathematical interest.

When the structures in the problem addressed are not f-easy, the situation is equally suggestive. If we have limited resources for constructing explanations of given evidence from a given hypothesis, then our inductive failure in worlds that are difficult to explain is determined. Number theorists have begun to take intrinsic

limitations on the formal investigation of mathematical structures very seriously in view of recent lower bound results regarding the lengths of proofs in decidable fragments of number theory [Fischer74]. Shapiro invites us, from a very practical and credible perspective, to share this real concern in the context of simple empirical problems.

Another interesting feature of Shapiro's procedure is that it is best described as searching a space of parts of hypotheses rather than of entire hypotheses, for recall that an hypothesis is a "surface" in a refinement graph whose vertices are single clauses. For example, whenever a clause is rejected as false, every hypothesis containing the clause is ignored without being explicitly considered and rejected as a separate entity. The usefulness of this perspective is evident when one considers an enumeration method whose enumeration includes every possible, consistent subset of clauses in a refinement graph. Consider all such surfaces bounded by k_0 in the refinement graph. Even if Shapiro's system does raise its bound to k_0, it considers only those surfaces it conjectures, and these are a microscopic fraction of the set of all finite subsets of clauses bounded in length by k_0.

3.5.8. Room for Improvement

There are lingering concerns about Shapiro's systems, however. From the prior discussion of the algorithm there is no reason to expect that the clauses in the procedure's conjecture at a given stage are logically independent. As in the case of Horning's "splitting" procedure (which is essentially a complete refinement operator with respect to a grammatical size measure that counts non-terminal symbols and an "entailment" relation defined by inclusion over the language generated), two distinct clauses can share equivalent or logically dependent refinements. For example, consider the simple operator ρ_1 discussed earlier. Its refinement graph would contain the following paths, (and many more) all leading to equivalent results.

$\{\}$	$\{\}$
$\{P(x_1,x_2,x_3)\}$	$\{P(x_1,x_2,x_3)\}$
$\{P(x_1,x_2,f_{x_4})\}$	$\{P(x_2,x_2,x_3)\}$
$\{P(x_2,x_2,f_{x_4})\}$	$\{P(x_2,x_2,f[x_1])\}$

Clearly, there is a model in which all clauses prior to the last one in each column are refuted. And once both of these clauses are represented in the current conjecture, each one is refined to spawn new redundancies, which again spawn new ones, and so forth. If logical dependence in the clauses of a conjectured

hypothesis is not controlled, refinements of redundant clauses can lead to testing and maintenance problems that quickly soak up all available resources. As the refinement operator becomes richer, as with the general operator ρ sketched above, we confront a snowballing of non-independent clauses in the boundary of the set of hypotheses currently refuted.

Shapiro's response to this difficulty is to compromise the inductive generality of the procedure by employing radically *incomplete* refinement operators. Consider the predicate *Plus*(x,y,z) interpreted to hold just when x+y=z. Now consider the standard recursive definition of *Plus* in terms of the successor operation in standard and clausal form:

standard notation:
(x)[Plus(0,x,x)]
(x)(y)(z)[Plus(x,y,z)--->Plus(s(x),y,s(z))]

clausal notation:
{Plus(0,x,x)}
{-Plus(x,y,z), Plus(s(x),y,s(z))}

It would be difficult, in general, to sift such an axiomatization out of an arbitrary clausal language whose vocabulary includes 'Plus', '0' and 's(__)'. But Shapiro's implemented program did not have to do much sifting. First of all, it is provided with exactly the vocabulary 'Plus', '0' and 's(__)'. This situation is a bit unrealistic, for sifting out irrelevant vocabulary is often a crucial aspect of addressing an interesting inductive inference problem. Nonetheless, sifting the adequate axiomatization from among all finite sets of arbitrary clauses on this vocabulary would still be an amazing feat. But the actual implementation did not have to do this either. Rather, it made use of a very incomplete refinement operator defined as follows:

$q \in \rho_2(p)$ just in case

1. $q \in \rho_1(p)$ or

2. $p=\{P(t_1,\ldots,t_n)\}$ and $q = p \cup \{-P(x_1,\ldots,x_n\}$, where x_i *occurs in* t_i.

Notice that ρ_1 is the operator for clauses with at most one atom. Notice also that no refinement of a clause can ever have more than one non-negated atom and one atom. Nor can any clause of unit cardinality have just one negated atom. Nor can there be any complex terms in the negated atom, or even any variables that do not occur in the non-negated atom *and* in the same position. Finally, notice that both atoms share the same predicate. In general, such clauses have the form

$\{-P(x_1,x_2), P(f(f(f(x_1), f(x_2)\}$

Such clauses are called *context-free transformations* by Shapiro.[50] Given this heavily restricted refinement operator, the search for the *Plus* axioms begins to seem much easier.

The refinement operator ρ_2 begins with the only clause of length 0, namely {}. {} is false in arithmetic, so the refinements of {} are generated. There is only one, namely *Plus*(x,y,z). This clause is also false in arithmetic, so its refinements are generated. At level 2 the real search begins, for the refinement operator does not constrain the position or composition depth of the functor s(__) or the position of 0 *a priori*. So the refinements of *Plus*(x,y,z) are exactly the clauses

specialization
{Plus(x,x,z)}
{Plus(x,y,x)}
{Plus(y,y,z)}
{Plus(x,y,y)}*
{Plus(z,y,z)}
{Plus(x,z,z)}

term subst.
{Plus(s(u),y,z)}*
{Plus(x,s(u),z)}
{Plus(x,y,s(z))}*
{Plus(0,y,z)}*
{Plus(x,0,z)}
{Plus(x,y,0)}

tacking on an atom
{Plus(x,y,z), −Plus(x,y,z)}

These clauses are all false, except for the last one which is a tautology, and hence will remain in the convergent conjecture. The divisions in the table reflect the cases under which each clause was generated from *Plus*(x,y,z), and should be self−explanatory.[51]

Each of the starred clauses is on a path to a clause that will be in the hypothesis converged to. The rest will be refined unsuccessfully until all their paths run into the size ceiling that is fixed when the first adequate hypothesis (i.e. the usual definition of *Plus*) is reached. For reasons of space, I trace only the progress of the four starred hypotheses. At level 3, the refinements of {*Plus*(x,y,y)} include {*Plus*(*0*,*y*,*y*)}, which is true in the standard model of arithmetic, so it will remain in

[50] Clauses with at most one negated and one non-negated atom are called transformations by Reynolds [Reynolds70]. Deciding consistency for finite sets of transformations can be shown to be undecidable. Reynold's proof does not determine whether Shapiro's more restricted case is decidable or not.

[51] Note that constants are 0-ary functions, so the constant instantiations are special cases of new term substitution.

the system's conjecture forever. The refinements of {*Plus*(0,y,z)} include {*Plus*(0,y,y)} and {*Plus*(0,z,z)}, both of which are true, and which remain in the system's conjecture forever.

The 13 refinements of {*Plus*(s(u),y,z)} include

{Plus(s(u),y,s(w))}

which leads to a true clause, and similarly, the 13 refinements of {Plus(x,y,s(u))} include

{Plus(s(w),y,s(u)}

which also leads to a true clause. In either case, the paths of the other 12 clauses bump into the final size ceiling and die in the convergent conjecture's "hole". Finally, at level 4, both of these clauses have a true refinement and the procedure's convergence point is reached. Specifically, we have

{Plus(s(u),y,s(w))} refines to
{Plus(s(u),y,s(w)), −Plus(u,y,w)} and

{Plus(s(w),y,s(u)} refines to
{Plus(s(w),y,s(u)), −Plus(w,y,u)}

Notice that the added atoms are *uniquely determined*, for their predicates and their variable patterns are fixed by rule (2) of Shapiro's implemented refinement operator. This strong restriction on refinement makes it very easy to find the desired addition postulates, which happen to be considered in every possible world at the expense of an infinity of alternatives which are considered in no circumstances. But the speedup represents no improvement in *efficiency*, for the problem solved is now a much easier one. The speed advantage is the infamous toaster's speed advantage, for it is won at the expense of a radical decrease in inductive scope.

To finish the story, the conjecture to which the algorithm converges is

(x)(y)(z)[Plus(x,y,z) v −Plus(x,y,z)];
(y)[Plus(0,y,y)];
(y)[Plus(0,y,y)];
(z)[Plus(0,z,z)];
(u)(y)(w)[Plus(u,y,w) ---> Plus(s(u),y,s(w))];
(w)(y)(u)[Plus(w,y,u) ---> Plus(s(w),y,s(u))]

Notice that even with the great *a priori* limitations on the refinement operator employed, redundancies are still considered in the inductive process. They do not

seem serious in this case, for the target hypothesis was tailored to the incomplete refinement operator employed and very little search was involved. But it is easy to imagine the impact of such redundant clauses if the search were not arbitrarily constrained. So the implementation just described not only robs its own inductive scope for speed; it fails to exploit opportunities to ignore hypotheses that might have been ignored without compromising inductive scope.[52] There is nothing wrong with studying solutions to easy inductive problems. In Chapter Two, I argued that efficient solutions to easy problems are no less important than efficient solutions to difficult ones. But we must be very careful not to confuse sensible solutions to easy inductive problems with spectacular solutions to difficult ones.

Shapiro's system is perhaps the most sophisticated and practical procedure for effective, logical inductive inference that has yet been proposed. But it would be even better than it is if it could *ignore* these redundant clauses without the sort of radical compromise in inductive scope we have noticed in Shapiro's implemented systems. The problem of ignoring some of these clauses is addressed in chapters six and seven below.

3.6. Conclusion

The purpose of this chapter was to illustrate in a positive way what the logic of discovery, as I have characterized it, might look like. Of particular interest is the great attention to detail that is apparent in the work of computer scientists in this area. This attention is not gratuitous, for efficiency is at stake, and sloppy descriptions of procedures can easily obscure their intractability and wastefulness.

The systems examined in this chapter suggest several obvious strategies to improve the performance of a general solution to a given inductive problem without affecting its inductive generality. For example, one can alter the procedure to withhold hypotheses from unnecessary empirical tests. Or one can design a method that ignores some hypotheses altogether. The set of hypotheses withheld from test

---

[52] Shapiro does address the problem of redundant clauses in conjectures when he considers practical implementations of his algorithm employing more general refinement operators.

> Optimizing the number of hypotheses in the conjecture is another problem to be solved. The model inference algorithm is guaranteed to minimize the maximal size of the hypotheses in the conjecture. The conjecture may contain many superfluous hypotheses that do not increase its logical power. This behavior may also influence the efficiency of the algorithm, since the complexity of the tests in the condition of the *while* loops grows with the number of hypotheses [clauses] in the conjecture [hypothesis]. [Shapiro81], p. 35.

or ignored may be a function of the evidence at hand, or may be fixed over all possible worlds. That is, hypotheses may be ignored *a posteriori* or *a priori*, respectively.

The usual technique for ignoring hypotheses *a priori* is to define a normal form for hypotheses such that the set of all normal hypotheses is simple to enumerate. But if hypothesis adequacy has anything to do with syntactic complexity, the normal language must be sure to catch at least some minimally complex element of every equivalence class of hypotheses. Otherwise, inductive scope is compromised, as we saw in the discussion of Horning's procedure.

The standard technique for avoiding hypothesis tests *a posteriori* is to arrange the hypothesis language in a tree, and to cut off sub-trees when their respective roots are refuted. This technique was employed, for example, by Pao and Horning. As we have seen in the systems of Horning and Shapiro, however, this method fails to ignore equivalent formulations of an hypothesis that occur in distinct subtrees whose roots are unrefuted.

In the next chapter I introduce a logical generalization problem of a sort not yet addressed in the computational literature. In developing some solutions for it, I return to the techniques for improving efficiency that were reviewed in this chapter. As it turns out, there are some fundamental theoretical limitations the applicability of these approaches.

Chapter 4
The Induction of Universal Theories: Problem and Solutions

Until now, my intention has been to characterize the study of hypothesis generators at a very general level. But at some point, talking about the study must give way to the study itself. In the balance of the thesis, I introduce a logical generalization problem and propose a variety of solutions to it. In the subsequent chapter I discuss what it would mean for a solution to such a problem to be efficient. Chapters six and seven present ways to make the discovery methods of this chapter more efficient, as well as some strong reasons to expect that some plausible approaches to improving their efficiency will never succeed.

4.1. A Logical Generalization Problem

4.1.1. Shapiro's Model Inference Problems Revisited

Since Shapiro has already studied a logical generalization problem, let us review some of its features. Recall (from chapter two) that in the inductive inference problems addressed by Shapiro's actual methods, the hypothesis language is some subset of the purely universal fragment of an arbitrary first order language. The evidence language consists of the atomic sentences of the hypothesis language or their negations. The possible worlds considered by Shapiro are just those in which each element is denoted by some closed term of the hypothesis language.

For Shapiro, an hypothesis h is adequate for possible world w just in case h is true in w and h entails every evidence sentence true in w. But recall that for the problem just described, this condition is equivalent to requiring an adequate hypothesis to entail all and only the evidence sentences true in the target structure. To contrast this adequacy criterion with other proposals later in the chapter, let us call it *evidence-complete adequacy* or or *C-adequacy* for short.

The goal of finding C-adequate hypotheses makes perfect sense in view of

Shapiro's goal to develop an automated computer programmer, for such an hypothesis is essentially a PROLOG program for computing the relations that are mentioned in the evidence. But as a general, intuitive criterion of hypothesis adequacy, C-adequacy is both too weak and too strong.

It is too strong, for none of our most highly revered empirical theories measure up to it. These theories require particular "initial conditions" to be specified before any atomic predictions can be derived. But according to C-adequacy, no such initial conditions should be required to get predictions out of an adequate theory. In terms of state spaces, an C-adequate theory of a system must select a unique state trajectory for the system, rather than merely restrict the possible trajectories of the system. For example, it is not enough to know that if an apple is dropped at time t it will be on the ground at time t'. A C-adequate theory must *entail* that the apple is dropped at time t and that it is on the ground at time t', if it actually is. Physics would look much different than it does if this norm were taken seriously. For example, relativity theory would be compelled (on pain of *inadequacy*) to entail a particular value for the stress-energy tensor at every space-time point causally connected to our own. But there is no evidence of any desire for such a revision of the theory among theoretical physicists.

At a simpler level, imagine a world in which there are black ravens, non-black non-ravens, and black non-ravens, but no non-black ravens. Moreover, imagine that we are interested in truths about "blackness" and "ravenhood". The generalization "all ravens are black" seems perfectly adequate for this world, given our interest in blackness and ravenhood. But since this hypothesis does not specify the blackness and ravenhood status of each object describable in the evidence language, it is not C-adequate even if it is true.

But if C-adequacy seems too strict in its rejection of our favorite examples of empirical theories, it is also too weak. For example,

$(x)[0+x=x]$

$(x)(y)(z)[-(x+y=z) \vee s(x)+y=s(z)]$

is an C-adequate hypothesis for the structure $\langle N,+\rangle$.[53] In fact we saw in Chapter 3 that Shapiro's program, equipped with a rather *ad hoc* refinement operator, quickly converges to exactly this hypothesis (with some redundant clauses thrown in) when supplied with basic facts about addition. But this hypothesis does not imply:

1. $(x)(y)[x+y=y+x]$

---

[53] i.e., the natural numbers with addition.

2. (x)(y)(z)[(x+y)+z=x+(y+z)]

(i.e. the associativity and commutativity of addition [Boolos80], p. 168.) The problem is that there are extensions of the structure <N,+> possessing domain elements not denoted by any closed term of the hypothesis language that satisfy each evidence sentence true in <N,+> but that do not satisfy either (1) or (2). But commutativity and associativity are such obvious, general truths about addition on the natural numbers, that every schoolboy can recite them. So obvious and general are these truths that a theory of addition that does not entail them is in some sense inadequate.

C-adequacy is concerned with the usefulness of a theory as a predictive machine. My objection to this criterion is not that one never wants to find such a theory. In the task of automatic computer programming, one obviously does. But there is another, distinct aim for inquiry that is not captured by the criterion of C-adequacy. This aim is to find general truths about one's world, whether or not they specify a perfect predictive machine. Inquiry can aim to be edifying and informative as well as useful, and these aims are not always the same.

4.1.2. Logical Inference as the Construction of a Universal Theory

It is almost a cliche that an aim of empirical science is to discover true generalizations. To what extent can this cliche be formally addressed in the logic of discovery? The rest of this chapter is devoted to a study of proposed solutions to logical inductive inference problems involving the following adequacy criterion for purely universal theories:

> A purely universal theory is universally adequate (U-adequate) for a structure just in case it is true in the structure and it entails every purely universal sentence true in the structure.

This definition alters the notion of an inductive inference problem slightly in that it treats *theories* rather than individual sentences as the proper objects of adequacy. Hence, infinite sets of sentences that are not finitely axiomatizable (or even axiomatizable) are potential candidates for adequacy. This alteration generalizes our previous framework, for if a finitely axiomatizable theory is adequate, any single sentence that axiomatizes this theory may also be said to be adequate.

It is evident that U-adequacy addresses what was taken as the undue leniency of C-adequacy, for the commutativity and associativity of addition must be entailed by any U-adequate theory of the structure of addition on the natural numbers. On the

other hand, for languages and worlds of the sort entertained by Shapiro, each U-adequate theory must be C-adequate as well, for a basic sentence is purely universal in a trivial sense.

The additional stringency of U-adequacy can be diminished by dropping Shapiro's requirement that the hypothesis language include the evidence language, and by considering an hypothesis language H such that not every domain element is named by a closed term of H. For example, an ornithologist interested in bird color may denote the fifth raven to fly past by the expression 'r(r(r(r(r(0)))))'. But his background knowledge may strongly suggest that the color of birds has nothing to do with the order in which he sees them. Therefore, he would not consider intricate hypotheses involving the function r, such as

B(0) & R(0)

(x)(y)(z)[(B(x) & R(x)) ---> (B(r(r(x))) & R(r(r(x))))],

which says that every even numbered thing is a black raven. But this function symbol would occur in each evidence report as a notational device. A student of addition could similarly view the successor function symbol in the language of arithmetic as a mere notational device for naming numbers and concern himself only with universal truths expressed in the non-logical vocabulary {0,+}. In this case, an adequate theory would be required to entail the commutativity and associativity of addition along with the fact that 0 is the additive identity element, without being required to entail each basic statement true in <N,+>. This seems a reasonable requirement for an adequate hypothesis about the structure of addition.

It might be objected that U-adequacy is too weak because it fails to hold the inductive agent responsible for laws of more general logical forms. On the one hand, this objection is indisputable. The restriction to universal sentences without function symbols results in a radical limitation on the expressive power of the hypothesis language. On the other hand, the *study* of an inductive problem is more interesting if there is at least some hope of developing a practical solution for it. The undecidability of first-order relational logic with arbitrary quantifier structure promises a very rough road for the logic of discovery in any problem defined over such a broad, expressive language. Purely universal theories, on the other hand, have a simpler structure that can be exploited computationally. Other decidable fragments of first-order logic involving existential quantification are also decidable, but the terrain of universal languages is more familiar from a computational perspective.[54]

---

[54] This familiarity arises from the extensive literature on "resolution theorem proving".

It is also important to remember that "expert systems" like MYCIN and DENDRAL, which rival human experts in some domains, generally rely on purely universal theories in their operation. Hence, the restriction to universal theories should not be assumed to be practically trivial, even if it is computationally and logically trivial compared to the unlimited, first-order problem.

4.1.3. Inductive Generality and Convergence to Theories

Now that a criterion of theory adequacy has been settled upon, it remains to characterize inductive generality for methods with finite outputs when the objects over which adequacy is defined may be infinite theories. Figuratively, a very large buffalo must be pulled out of a very small hat. Recall that inductive generality is defined in terms of inductive scope, inductive scope is defined in terms of structure identifiability in the limit, and identifiability in the limit is defined in terms of convergence of a device to an hypothesis on an infinite sequence of inputs. So the basic problem is to define a theoretically interesting sense in which a device whose conjectures are all of finite length can be said to converge to an infinite theory.

If a theory is axiomatizable, it can be specified by a program for deciding some set of axioms whose deductive closure is the theory. So axiomatizable theories can be conjectured "all at once" just by conjecturing a procedure for deciding some axiom set for the theory. Then it is easy to define convergence to such a procedure, convergence to an axiom set (so that convergence is consistent with infinite variation in the conjectured procedures) or convergence to a theory (so that convergence is consistent with infinite variation in the sets decided by the conjectured procedures). These criteria are all of the form that after *some* finite time each conjecture entails *all* and only the sentences in the theory converged to. Therefore, these notions will be referred to collectively as EA-convergence criteria for theories, where EA reflects the order of the quantifiers over times and sentences, respectively.

Identification of a structure can be defined as EA-convergence (in one of these senses) to an axiomatization of a theory adequate for the structure on any presentation of the total evidence for the structure. Depending on the adequacy criterion chosen, we may speak of EA-C-convergence or EA-U-convergence. In dreaming up names for identification criteria, I shall always put the abbreviation for the convergence criterion first, and the abbreviation for the adequacy criterion second. By convention, either abbreviation can be deleted when it is specified by context or irrelevant.

But there is also a more liberal approach, considered by Feldman ([Feldman72], p. 34) and proposed independently by Glymour [Glymour84], that is general enough to permit convergence to unaxiomatizable theories:

> Method D *AE-converges* to theory T on infinite sequence e of evidence just in case for each t in T there is a stage after which the conjectures of D all entail t, and for each t not in T there is a stage after which the conjectures of D do not entail t.

Unlike the EA-convergence criteria just discussed, AE-convergence does not require that there be *some* point after which each conjecture entails *all* and only the right sentences. Rather, for *each* sentence, there is *some* time after which it is eternally settled in the right way by subsequent conjectures. In fact, D can AE-converge to T without ever conjecturing T. Hence, the appropriateness of the label 'AE-convergence', for this criterion essentially permutes two of the quantifiers in the prefix of the definition of EA-convergence. D can AE-identify structure M just in case D AE-converges to an adequate theory of M on each complete sequence of the evidence true in M, and AE-scope and AE-generality are both defined as usual in terms of AE-identification.

AE-convergence is, admittedly, a lenient convergence criterion, but it does have an attractive property not shared by some criteria that have been proposed in the literature.[55]

Fact:

If M AE-converges to some T on e, T is unique.[56]

Hence, AE-convergence can be thought of as an alternative output convention for Turing machines by which a machine can take an infinite object as input (an infinite evidence sequence) and can output an infinite object (a theory), just as in the case of H-convergence. So AE-convergence associates each Turing machine with a unique function from infinite evidence sequences to theories, just as the standard convention associates each Turing machine with a partial recursive function.

It is clear (even by the names of the criteria in question) that an ability to EA-identify a structure implies an ability to AE-identify it, so the class of AE-identifiable sets of structures includes the class of strictly identifiable structures.

---

[55] A device is said to hyper-converge to an output just in case it conjectures this output infinitely often [Kugel77]. It is clear that a procedure can converge in this sense to infinitely many distinct outputs on the same infinite input sequence.

[56] Let T' be a theory distinct from T. Then there is an h that is not entailed by the intersection of T,T' but that is entailed by T or by T'. But then there is some point after which no conjecture entails h and there is also some point after which every conjecture entails h, which is absurd.

Also, assuming C-adequacy, a basic-statement evidence language, and an hypothesis language including the evidence language, *every* structure is AE-C-identifiable by a simple device that conjectures the conjunction of the evidence read so far, for such a device must AE-converge to exactly the basic statements true in the given structure, and this theory must be true regardless of Shapiro's admissibility requirement on structures and languages. But such a trivial, "skeptical" method cannot *AE-U*-identify every structure, for no finite set of such evidence entails a universal sentence. Therefore, AE-U-identification still requires a method to formulate conjectures that are not entailed by the available evidence.

Of course, it would be strange to get something of value for nothing just by weakening input-output conventions. It should therefore be mentioned that possessing a device that can AE-U-identify the actual world need not be as useful as having a device that can EA-U-identify the actual world. Consider a finite, immortal system, like the Greek goddess Athena. Athena can ask only finitely many questions in a finite amount of time. Therefore, if she relies on the conjectures of a device that can EA-U-identify the actual world, she can receive at most finitely many incorrect answers in her infinite career of inquiry. On the other hand, if she relies on a device that AE-U-identifies the actual world, she may receive infinitely many incorrect answers in her infinite career, for each conjecture of such a device may be false. Also, if Athena relies on such a device, then there may be no time at which she "possesses" the adequate theory the device converges to, even if there is a finitely axiomatizable theory that is adequate for her world.

On the other hand, if Athena is particularly interested in an arbitrary, finite set Γ of hypotheses, AE-U-identification is as useful as EA-U-identification, for in either case there comes a time after which each element h of Γ is entailed by each subsequent conjecture just in case h is true.

4.2. A Discovery Method Based on Hempel's Confirmation Relation

4.2.1. Hempelian Confirmation and Theory Identification

Hempel's familiar confirmation relation for an hypothesis language H without function symbols or identity amounts roughly to a sort of "strengen implikation" from the evidence to the confirmed hypothesis, in which the class of possible worlds is a function of the evidence. That is, e confirms h just in case for every structure **M** in class O(e), if **M** $\models$ e then **M** $\models$ h. Structure **M** for H is in O(e) just in case for every element d of **M**'s domain, there is a constant c occurring

nonvacuously in e that denotes d. The number of constants occurring nonvacuously in e provides an upper bound on the cardinality of the structures in O(e), so the relation is decidable by enumerating these structures and performing the recursive satisfaction test for each. Finally, e *disconfirms* h on Hempel's proposal just in case e confirms −h, and in keeping with the motivation in Hempel's article [Hempel43], we can say that e *is irrelevant to h* just in case e neither confirms nor disconfirms h.

Intuitively, e confirms h just in case the truth of e implies the truth of h under the assumption that the objects mentioned in the evidence are the only objects in the structure under investigation. This is stronger than the requirement that there be no more objects in the world than there are objects described in the evidence, for O(e) could then include structures for H with no more than this number of objects, but which still leave individuals un−named by evidential constants.

Hempel's relation may also be conceived of syntactically. Let T[e] be a first−order sentence that says every object is named in the evidence. That is, if occ(e) = the set of all constants occurring non−vacuously in e, and occ(e) = $(c_1,\ldots,c_n)$, then

$$T[e] = '(x)[x=c_1 \vee x=c_2 \vee\ldots\vee x=c_n]'$$

Evidently, e confirms h just in case $T[e],e \models h$.[57] Let C denote Hempel's confirmation relation, and let D denote his disconfirmation relation. Notice that since C(e,h) is equivalent to $T[e],e \models h$, the properties of C should be quite similar to the general properties of entailment in a first−order theory. Indeed, all the properties mentioned by Hempel in his well−known article "A purely Syntactical Definition of Confirmation" [Hempel43]. (as well as a few others) are properties of just this kind.

[57] Note that there are no occurrences of the identity relation in e or in h.

Hempel's "entailment condition".

If e|=h then C(e,h)
If e|=−h then D(e,h)

Hempel's "consequence condition".

If Γ is a set of hypotheses,
Γ|=h', and for each h in Γ, C(e,h),
then C(e,h')

If Γ is a set of hypotheses,
Γ|=−h', and for each h in Γ, D(e,h),
then D(e,h')

Hempel's "consistency condition".

If e is consistent, C(e,h) and C(e,h')
then h, h' and e are mutually consistent.

If e is consistent, D(e,h) and D(e,h')
then −h, −h' and e are mutually consistent.

confirmation of a negation

If C(e,−A) then not C(e,A), but not conversely.
If D(e,−A) then not D(e,A), but not conversely.

confirmation of a conjunction

C(e,A&B) if and only if C(e,A) and C(e,B).
D(e,AvB) if and only if D(e,A) and D(e,B).

confirmation of a disjunction

If C(e,A) or C(e,B) then C(e,AvB) but not conversely.
If D(e,A) or D(e,B) then D(e,A&B) but not conversely.

Hempel's confirmation theory suffers from some devastating objections as an intuitive norm of evidential relevance. The most common of these is the "raven paradox", which arises because −Ra confirms (x)[Rx−−−>Bx] according to Hempel's theory, but not intuitively. Hempel's reply is that −Ra is positively relevant to (x)[Rx−−−>Bx], but perhaps not as strongly as Ra and Ba would be. Since a qualitative explication of confirmation is intended to capture positive evidential relevance and not its degree, there is no problem.

Another well-known difficulty with Hempel's account is that any hypothesis that implies that there are k objects is *disconfirmed* when fewer than k constants occur non-vacuously in the evidence, for such an hypothesis contradicts T[e], and so T[e],e|=−h. For example, Peano's infinity postulates

(x)(Ey)(P(x,y))
(x)–Pxx
(x)(y)(z)(Pxy & Pyz ---> Pxz)

cannot be confirmed by any finite set of evidence. Under a very charitable reading one could consider this result to be a way in which Hempel's confirmation theory enforces the observance of "Ockham's Razor": if one hasn't seen more than n objects, then one ought not to believe an hypothesis entailing the existence of more than n objects. But since the entire point of confirmation theory is to provide an account of evidence for claims that apply to unobserved as well as to observed events, this stringency accords poorly with Hempel's insistence that his relation of confirmation holds whenever evidence is even minutely supportive of a given hypothesis.

Deductive logic is monotonic in the following sense: one never loses a theorem by adding a premise. Hempel's confirmation theory is highly non–monotonic in this sense, for any epistemic relation (confirmation, disconfirmation or irrelevance) can be the result of adding a single evidence sentence to evidence currently bearing any given epistemic relation to a given hypothesis. Any reasonable confirmation theory must be non–monotonic, but the non–monotonicity of Hempel's criterion is hopelessly unreasonable. There can be a purely universal, function free hypothesis h confirmed by an arbitrarily large body E of evidence such that there is a single evidence sentence e consistent with E that by itself confirms h, but such that E U {e} is irrelevant to h. For example, the platitude "everything is related to everything else" is confirmed by the single sentence "Frank is related to himself" but not by nine hundred–ninety–nine sentences asserting the relation of all but one pair of names from a set of one hundred names. If the relation is to explicate the notion of positive evidential relevance, it is difficult to imagine how nine hundred, ninety–nine instances could be irrelevant to an hypothesis when each of these instances is relevant by itself.[58] And as the number of names in the evidence increases, this absurdity increases exponentially.

Hempel criticized Nicod for taking confirmation to be a semantic relation between an hypothesis and observed objects. But ironically, if we think of adding new evidence as adding everything there is to know about a newly observed individual, the counterexample just described disappears, for

> *Fact:* If e is the diagram of some finite structure for a purely universal hypothesis language H, then for any h in H, C(e,h) just in case M|=h.

---

[58] I do not intend to suggest that such a consequence is always undesirable. In a sophisticated confirmation theory, the evidence sentence "the previous evidence is mistaken" could cancel the positive relevance of all prior evidence. But the anomaly under discussion arises from no such consideration and is merely a technical artifact.

Since a purely universal, function-free hypothesis is true in a structure M just in case it is true in each substructure M' of M, the set of such confirmed hypotheses is non-increasing as the evidence increases.

A dual result clearly holds for purely existential hypotheses. That is, the set of confirmed, purely existential, function-free hypotheses is non-decreasing as the evidence increases. Each of these properties is fairly natural, for as evidence increases, one finds both more counterexamples to universal claims and more objects that vindicate existential claims.

If constants are admitted into the hypothesis language H, then not every restriction of a realization M of H to some subset of the domain of H is a substructure of M; for any such restriction whose domain excludes a distinguished element of M is not a substructure of M. The analogous result for hypothesis languages with constants is that *once all the hypothesis language constants occur essentially in the evidence* the set of hypotheses confirmed on this evidence is non-increasing as the evidence increases. Until all the constants occur in the evidence, the set of confirmed hypotheses may increase.[59] So if there are only finitely many constants in the hypothesis language, there is only a finite initial segment of exceptions to the above result. If there are infinitely many constants, however, there is no finite initial segment of evidence after which the set of confirmed hypotheses is nonincreasing in the evidence.

Despite the fatal intuitive difficulties facing Hempel's confirmation theory, the idea of instance or satisfaction based confirmation relations is common in A.I. applications. DENDRAL, for example, rates hypotheses by counting their positive instances in the data. And all the various "concept learning" systems (e.g. Hunt's CLS [Hunt66]) can be viewed as generating universal, monadic biconditionals confirmed on Hempel's criterion. Hence, there is some independent motivation for the study of the usefulness of Hempel's theory as a cog the AE-U-identification problem posed in the previous section. Happily, the fact that the set of confirmed, purely universal hypotheses is non-increasing as more *individuals* are "inspected" provides an idea for a Hempelian logic of discovery that solves this identification problem quite generally. Such a system is presented in the next section.

---

[59] For example, Pa does not confirm Pa&Pb, but Pa,Pb does.

4.2.2. A Hempelian Procedure

Let L be an arbitrary, first-order language without function symbols of arity 1 or more that has no more than finitely many constant symbols. Let the hypothesis language H be the set of all elements of L in prenex normal form in which no existential quantifiers occur. Let L' be the augmentation of L with a *countable infinity* of constants, and let the evidence language E be the atomic sentences of L' or their negations. Let the possible worlds be structures for L', each of whose domain elements is named by some constant of L'.[60] The evidence presented to an inductive method is assumed to be an enumeration of the set of all evidence sentences true in the target world. The operative identification criterion is AE-U-identifiction.

The question is whether the generalization problem just defined can be solved by a method that relies on Hempel's confirmation relation in the short run. In fact, it can be, as the following procedure shows.

- Choose an arbitrary, effective ordering of H.[61]
- Let Hyps(i) be a subroutine that can generate, for any i, the set of all hypotheses occurring no earlier than i in the assumed enumeration of H.
- Let e be a finite set of evidence and let h be a finite set of hypotheses. Let Sort(e,h) return the set of all hypotheses in h that are confirmed by e in light of Hempel's confirmation relation.
- Let subroutine Subev(e) output for any finite set e of evidence sentences some maximal subset of this evidence that is the diagram of a finite structure for L.

```
PROCEDURE HEMP:
BEGIN
        Ev:={};
        Stage:=0;
        REPEAT forever
        BEGIN
                Ev:= Ev U READ(Stage)
                Conjecture(Sort[Subev(Ev),Hyps(Stage)])
                Stage:=Stage+1;
        END
END.
```

At stage n, this procedure adds the nth evidence sentence to its stored evidence (Ev) and calls Subev to solve for the greatest subset of this evidence that is the diagram of some finite realization of L. For example, if the nonlogical vocabulary of L is {P,Q}, and the given set is

[60] This requirement has the same force as Shapiro's admissibility requirement.

[61] Of course, there is one for the prenex normal sentences.

P(a,b)
P(b,a)
P(b,b)
P(a,a)
P(a,c)
Q(a)
Q(b)
Q(d)

then the output of Subev is

P(a,b)
P(b,a)
P(b,b)
P(a,a)
Q(a)
Q(b)

Next, the procedure uses Hyp(n) to find the first n hypotheses in H. Then Sort returns those hypotheses in this set that are confirmed on the reduced evidence, and the output of Sort is conjectured. Finally, the level is incremented, and all the preceding steps are repeated for the next evidence sentence.

It is easy to see that HEMP is a general solution to the inductive inference problem in question. That is,

Fact: HEMP can AE–U–identify any structure in O.

Proof: Let M be in O, and let $h \epsilon H$ be true in M. Then there is some stage n' at which h is in Hyp(n'). Since the evidence presentation provided to HEMP in the limit is complete, there is a stage n after which which Subev(e) is always the diagram of a finite restriction M' of M to a domain that includes each of the finitely many distinguished elements of M. Since h is true in M and h is purely universal, h is true in any substructure of M. But any restriction of M to a subdomain that includes all (finitely many) distinguished elements of M is a substructure of M. Hence h is true in M'. So by the previous fact, h is confirmed by Subev(e). Then h is in the conjecture of HEMP at every stage after stage MAX{n,n'}, so the conjecture at each of these stages trivially entails h.

Now let $h \epsilon H$ be false in M. It must be shown that after some finite stage n, the conjecture of HEMP never entails h. Since h is in H and hence is purely universal with no function symbols, if there are k distinct variables occurring in h, and k constants in the language H, then h is false in some substructure M' of M of cardinality k+j. Since the evidence presentation is complete, there is a stage n after which Subev(e) is always the diagram of a structure M'' of which M' is a substructure. Since h is false in M' and hence in M'', h is disconfirmed by Subev(e). Each conjecture of HEMP is a finite set of confirmed hypotheses, and the conjunction of these hypotheses must be confirmed, for a conjunction is confirmed if and only if each conjunct is. But every logical consequence of a confirmed hypothesis is confirmed, by the fact that C satisfies Hempel's' "consequence condition". Since h is not confirmed, h is therefore not entailed by any conjecture of HEMP after stage n.

Notice that there is no reason to expect HEMP to EA–U–identify O, for the incrementation of Hyp(n) can always lead to yet another false conjecture before the evidence refuting the added hypothesis is encountered. Hence, even if HEMP conjectures an adequate theory at some point, there is nothing to prevent it from strengthening this conjecture and re–weakening it infinitely many times. On the other hand, if HEMP does not explore conjectures stronger than its current one, it will necessarily fail to identify the structures for which only stronger hypotheses are adequate. It is natural to wonder whether this epistemological dilemma results from a design flaw in HEMP or from an intrinsic feature of the inductive problem addressed. For example, one might reasonably suspect that allowing HEMP to conjecture infinite axiomatizations would permit it to achieve stronger identification properties, for it is trivial that a device that cannot conjecture an infinite axiomatization cannot EA–U–identify a structure whose H–truths are not finitely axiomatizable.

It is also tempting to expect that the class of structures for H with finitely axiomatizable adequate theories is EA–U–identifiable, for entailment is decidable over H. One might expect to EA–U–identify any structures with an adequate, finitely axiomatizable theory by successively testing the sentences in an effective enumeration of H that is everywhere non–increasing with respect to entailment.[62] Given such an enumeration, one could EA–U–identify O by waiting until Subev(e) is not empty and then by conjecturing the first hypothesis in the order that is confirmed on this evidence. There must be one, so this search would take finite time. Moreover, any false hypothesis would be rejected after reading some finite amount of evidence, so the system could not converge to a false hypothesis. Finally, the device would converge to the first true hypothesis, and by our assumed ordering, this hypothesis would entail all the true hypotheses in the structure (if any hypothesis does) for no true hypothesis could be properly stronger than it is.

But there can be no such ordering, effective or not; for the proposed identification task is impossible for *any* method that is a function of the current evidence, be it effective, intellectual, or even magical.

> *Theorem:* Let V be a non–logical vocabulary including at least one predicate of arbitrary arity and equality, but no function symbols and no more than finitely many constants. Let H be the purely universal sentences on vocabulary V. Let C be an enumerable collection of constants, and let E be the atomic sentences on the vocabulary V U C. Finally, let O be the set of all structures for H whose objects are all named by constants in C.

---

[62] I.e., no sentence occurs in the enumeration before any sentence that entails it.

Then no method can EA-U-identify even that subset F of O whose H-theories are finitely axiomatizable by sentences in H.

Corollary: No method can EA-U-identify O.

The idea is quite simple, for consider the following ordered set of sentences:

(.)[Pxy & (x=y)]
(.)[Pxy & (x=y v x=z v y=z)]
(.)[Pxy & (x=y v x=z v x=w v y=z v y=w)]
.
.
.
(.)[Pxy]

The nth element of this order says that everything is P-related to everything else and there are at most n things. The minimal element of the set drops the cardinality restriction and merely says that everything is P-related to everything else. Notice that if h is "higher" in the order than h', then h entails h' *properly*. Moreover, each sentence in this set is an adequate hypothesis for *some* structure.[63] Hence, there is no complete sequence of universal complete (i.e. possibly U-adequate) hypotheses that is that is non-increasing with respect to proper entailment.

The point is that we can easily provide evidence about the structure for which the terminus of the order is adequate that would force *any* proposed method into the following, epistemic dilemma: either the method must skip some hypothesis in the order on arbitrarily much evidence consistent with it (and hence fail to identify its associated structure) or the method attempts to traverse the entire infinity of hypotheses between the endpoints of the order, and hence never "arrives at" the terminal hypothesis in the order, so it fails to identify the structure for which this terminal hypothesis is adequate. More precisely,

Proof: Let $P_1,...,P_n$ be all the predicates of H. Let U be the sentence that says every object is related by each of these relations to any other object. Let C[n] be the purely universal sentence that says there are at most n things, as in the sequence just illustrated.

Let D be a method that can identify F. Let σ be a complete sequence of the atoms of E. We begin by repeatedly feeding σ to D. Since D can identify F, it can identify any structure for which U & C[1] is adequate, so after some finite presentation of non-negated atoms, D must conjecture U & C[1]. Once D conjectures U & C[1], we casually continue feeding evidence to D as before until some atom a=b turns up

[63] The nth element h_n of the set is k-categorical for each $k \leq n$. Let M be a model of h_n of cardinality n. Let h be true in M. Since h_n and h are both purely universal, they are satisfied in each substructure of M. But h_n is satisfied only in structures isomorphic to substructures of M. Hence $h_n \not\models h$.

> in σ such that either a or b does not occur in any identity statement fed to D previously. Since σ is complete, this must happen in some finite amount of time. Once a=b occurs in σ, we feed −(a=b) to D. We continue once again to feed σ to D, except that any identity whose negation is entailed by −(a=b) and identities fed to D is negated before being fed to D. This evidence, if continued indefinitely, would be the complete presentation of the diagram of a structure for which U & C[2] is adequate. Since D can EA−U−identify F, it conjectures U & C[2] after some finite time. Then we once again wait for an identity with a new constant, and continue as before, and so forth.
>
> Since D can can identify F, it eventually changes its mind after receiving each negated identity in which a new constant occurs. Hence, infinitely many such negated identities will be read, so D will change its mind infinitely many times. Hence, the total evidence fed to D in the limit will be the diagram of an enumerable structure M that satisfies U. U is aleph−null categorical, for the only countable structures for H that satisfy U are those whose relations are all universal, and each of these structures is isomorphic to any other. Hence U is adequate for any such structure. But D does not converge to M on a complete presentation of the diagram of M. Hence, D does not identify F, contrary to assumption.

Therefore, HEMP is not to be despised for failing to EA−U− identify the structures for which a finitely axiomatizable theory is adequate, for this inductive scope is unattainable by any method. So it is unattainable *a fortiori* by any effective method.

4.3. A Discovery Method Based on Consistency with the Evidence

Although Hempel's confirmation criterion suffers from severe objections as a theory of confirmation, it has been shown to function as a useful cog in a general solution to a particular kind of inductive inference problem. This fact constitutes some reason for interest in Hempel's relation as a heuristic tool rather than as a confirmation relation. But just as the interest of an hypothesis as an explanation of a given phenomenon depends upon the availability of other equally good explanations, the interest of a confirmation relation as a general heuristic tool depends upon the performance other available methods−−− and methods that rely on very different "confirmation relations" can have the same inductive scope. For example, HEMP's inductive scope would remain unaltered even if the evidence were ignored for an arbitrary finite amount of time before being fed to Subev for the purposes of confirmation testing. And a procedure that conjectures several million hypotheses that are obviously refuted by the given evidence before it invokes HEMP as a subroutine would also have the same inductive scope as HEMP.

Consistency with the evidence is an outlandish proposal for a general relation of hypothesis suitability, for no one believes that an arbitrary sentence confirms every sentence that is logically independent of it. Nonetheless, it will be shown presently

that an algorithm relying merely on consistency with the evidence can also AE–U–identify O. Moreover, this method can begin to conjecture an hypothesis in the adequate theory of a structure M arbitrarily more quickly than HEMP can.

As an epistemic relation, consistency with the evidence does satisfy Hempel's equivalence condition, but it is very different in most respects. For example, consistency does not satisfy Hempel's entailment, consequence, and consistency conditions, even for a purely universal hypothesis language H that excludes constants and function symbols.[64] Gone also are the irregular non–monotonicities of Hempel's criterion. The class of hypotheses consistent with some evidence is non–increasing as the evidence increases.

On the other hand, when we also assume that the evidence is the diagram of a finite realization of H, then if h is an element of H, and the constants occurring non–vacuously in h occur non–vacuously in evidence e, e confirms h just in case e is consistent with h.[65] The requirement that H be purely universal is not vacuous, for the hypothesis that there is a white raven is consistent with any number of observations of black ravens, but may nonetheless be false in the world from which these observations are drawn.

This simple fact has an important consequence for HEMP. Since Hempel's confirmation criterion amounts to mere consistency with the evidence when the evidence is the diagram of some finite structure, and since HEMP applies the test only to such evidence, HEMP may as well simply conjecture all the elements of Hyps(n) that are consistent with the *total* evidence read by stage n. Hence, the bothersome period during which HEMP waits to find a finite structure diagram in the evidence before making a conjecture is eliminated. To solidify this suggestion, consider the procedure CONSIST, and its proof of adequacy with respect to AE–U–identification. The subroutine Hyps is just as described earlier. Sort2 is like Sort, except it produces the hypotheses in a given set that are consistent with the given evidence rather than those that are confirmed by Hempel's criterion. Finally, the subroutine Subev, which HEMP employs in order to stall until the presented evidence comprises the diagram of a finite structure, is appropriately missing.

---

[64] When consistency is being discussed as a "confirmation relation", it will be denoted C', and when inconsistency is viewed as a "disconfirmation relation" it is denoted D'. (1) If e is inconsistent then e entails h but h is not consistent with e, so not C'(e,h). (2) Let e be 'Pa&Ba', let h1 be 'Px--->Qx', let h2 be 'Qx--->-Bx' and let h3 be h1 & h2. Then C'(e,h1) and C'(e,h2) and h1,h2 ¦= h3, but not C'(e,h3). (3) Let e, h1, h2 be as before. C'(e,h1) and C'(e,h2) but h1,h2 are inconsistent with e.

[65] ===> is just Hempel's consistency condition, which C satisfies in general. <=== Let e be the diagram of M, and let h∈H. Since h is purely universal, e is consistent with h just in case M¦=h. Hence, by the above fact, C(e,h).

```
PROCEDURE CONSIST:
BEGIN
        Ev:={};
        Stage:=0;
        REPEAT
        BEGIN
                Ev:= Ev U READ(Stage)
                Conjecture(Sort2[Ev,Hyps(Stage)])
                Stage:=Stage+1;
        END
        FOREVER;
END.[66]
```

CONSIST has an advantage over HEMP, in that it is capable of eliminating refuted hypotheses immediately, without waiting for a substructure diagram to appear in the evidence. This "speed-up" is not bounded by any finite quantity, for the time consumed in waiting for the first finite structure diagram to appear in the evidence is unbounded.[67]

4.4. A Logic of Discovery Based on a Modification of Nicod's Criterion

Although HEMP and CONSIST are equally general solutions to the universal inductive inference problem, the performance of these methods in the short run will be quite different. HEMP can spend an arbitrary amount of time conjecturing a refuted hypothesis, while CONSIST is capable of conjecturing an hypothesis that has no vocabulary in common with the evidence and that seems therefore to have nothing to do with the evidence provided. These performance characteristics can be traced directly to the suitability criteria upon which the respective methods rely. Hempel's confirmation relation enforces some appearance of relevance between a confirmed hypothesis and its evidence, but is highly non-monotonic in the face of increasing evidence. Consistency with the evidence, on the other hand, is monotonic in the face of increasing evidence, but it does not ensure any reasonable degree of relevance between a confirmed hypothesis and its evidence.

---

[66] To see that for any constant, function-free H, CONSIST can AE-U-identify O(H), consider an arbitrary structure M in O(H). Now consider an arbitrary h in H that is true in O(H). There is a stage n after which h is always in Hyps(n). Moreover, since h is true in M, h never contradicts the evidence, and hence remains in each subsequent conjecture. Now consider an h that is false in M. There is a finite substructure M' of M in which h is false, for h is purely universal, and H has no function symbols. After some stage n, the diagram of M' is included in e. Each element of each conjecture h' after stage n is consistent with e and hence is true in M'. Therefore, the entire conjecture h' is true in M'. Then any logical consequence of h' is true in M' after stage n. Since h is false in M', h is not a consequence of any conjecture of CONSIST after stage n.

[67] Imagine that the evidence begins with '-Pab', and withhold the instance +'Paa' until position n in the evidence presentation, where n is arbitrary. HEMP would idly wait until stage n before rejecting the hypothesis '(x)(y)[Pxy]', even though this hypothesis is obviously refuted by the first evidence sentence encountered.

It would be desirable, therefore, to formulate a suitability relation that leads to the same inductive generality in the long run, but that steers an even course in the short run between the radical non-monotonicity of Hempel's criterion and the evidential leniency of consistency. Consider the following suitability relation,[68] which may be called the *modified Nicod's criterion*, or N(e,h) for short.

> *Definition:* N(e,h) just in case e entails some instance of h and h is consistent with e,

where an instance of an hypothesis h is some result of uniformly substituting constants for variables in h and eliminating all the quantifiers occurring in h. So for example, an instance of (x)(y)[Pax v (Ez)Qbz] would be [Pac v Qbd]. Notice that if e is consistent, then e confirms h on the modified Nicod's criterion just if e confirms the result of replacing each occurrence of a universal quantifier with an existential quantifier according to Hempel's criterion. So the modified Nicod's criterion is more stringent than the consistency criterion in that it requires the evidence to entail *some* instance of h, and it is more lenient than Hempel's criterion in not requiring the evidence to entail *every* instance of h when h is purely universal. Hence we have that C'(e,h) ===> N(e,h) ===> C(e,h), but not conversely.

Like consistency with the evidence, the modified Nicod's criterion does not satisfy the entailment, consequence, or consistency conditions.[69] But unlike consistency with the evidence, the modified Nicod's condition does not even satisfy the equivalence condition.[70]

Another difference between mere consistency with the evidence and the modified Nicod's condition is that the class of hypotheses consistent with the evidence is non-increasing as the evidence increases, but the class of Nicod-confirmed sentences does not always decrease as the evidence increases.[71] On the other hand, if the evidence presentation is complete, then for any hypothesis h, there is some point in the presentation after which the evidence entails an instance of h, and once this instance appears, h remains confirmed until it is refuted by further evidence. So while increasing evidence can alternately confirm and not confirm the

---

[68] This relation was suggested to me by Glymour.

[69] (1) if e is inconsistent, e entails h, but h is inconsistent with e so not N(e,h). (2) Let A=(x)(Px--->Qx) and let B=(x)(Qx--->Rx). Let C=A&B. Let e={-Pa,-Qa,Pb,-Rb}. Then N(e,A) and N(e,B) and A,B|=C but not N(e,C). (3) The previous example is also a counterexample to the consistency condition, for N(e,A) and N(e,B) but {A,B} is inconsistent with e.

[70] Let e={Pab}. Let h=(x)(Pxx), and let h'=(x)(y)[Pxx v Pxy]. Evidence e is consistent with both h and h', and e entails an instance Paa v Pab of h'. But e does not entail an instance of h. So N(e,h') but not N(e,h).

[71] For an hypothesis is not confirmed on the modified Nicod's criterion until some instance of it is entailed by the evidence.

same hypothesis infinitely often on Hempel's criterion, the modified Nicod's criterion permits only two alternations of status for any given hypothesis as the total evidence is received. Every hypothesis h begins its career non-confirmed on null evidence. Then it is possible that the evidence at some point entails some instance of h, but does not contradict h, in which case h is confirmed. Finally, a counterexample may arise in the evidence, in which case h remains eternally disconfirmed and hence non-confirmed thereafter. The possible alternations of status of a given hypothesis with respect to increasing evidence are summarized in the accompanying table.

Consistency Criterion:
confirmed forever.

confirmed, then disconfirmed and disconfirmed thereafter.

Modified Nicod's Criterion:
non-confirmed, then confirmed,
and confirmed thereafter.

non-confirmed, then disconfirmed,
and disconfirmed thereafter.

non-confirmed, then confirmed, then disconfirmed,
and disconfirmed thereafter.

Hempel's Criterion:
non-confirmed, confirmed, non-confirmed, confirmed,...etc...,then disconfirmed and disconfirmed thereafter.

non-confirmed, confirmed, non-confirmed,...etc..., in unending alternations.

Figure 4-1: The course of inquiry under different suitability relations

Let NICOD be a procedure just like CONSIST, except that NICOD employs the modified Nicod's criterion just where CONSIST applies a consistency test. NICOD can evidently AE-U-identify the set of all structures for H' whose individuals are all named by constants of H'.[72] On the other hand, NICOD will not conjecture hypotheses that share no vocabulary with the evidence, which is a an intuitive improvement over the short-term performance of CONSIST. Moreover, NICOD will never conjecture an hypothesis that is inconsistent with its current evidence, which is a rudimentary but nonetheless significant improvement over the short-term performance of HEMP, as well.

---

[72] For if h is true, then some evidence entailing an instance of h will appear in the evidence sooner or later and h will be conjectured forever thereafter. In case h is false, things work just as in the proof of adequacy of CONSIST.

Hempel, along with most other confirmation theorists, has insisted that any adequate concept of confirmation must satisfy the equivalence condition. The intuition is that evidence bears on what a sentence expresses rather than on a sentence itself. Sentences express propositions, and any two logically equivalent sentences express the same proposition. So evidence bears on one expression of the proposition just in case it bears on the other. It is interesting, then, that the performance of NICOD seems none the worse for relying on a criterion of suitability that violates this requirement. In fact, it seems to yield more sensible results than either HEMP or CONSIST. This fact illustrates how issues that seem crucial in conventional confirmation theory are diminished in importance when confirmation relations are viewed pragmatically as cogs in an overall discovery procedure.

From the discussion of the limiting performance of NICOD, it is evident that *any* syntactic relation R can be required in conjunction with consistency with the evidence without depleting the inductive scope of CONSIST, just so long as for each true hypothesis h, there is a finite set of true evidence e such that h bears R to each true set of evidence including e. That e entails an instance of h is, of course, only one such property. We could require, for example, that e entail an arbitrary, fixed number n of distinct instances of h. Or we could require that e entail some fixed number of *maximally general* instances of h, where a maximally general instance is one in which distinct constants are substituted for distinct variables occurring in h. This latter requirement is desirable, for it places a reasonable bound on the short-term optimism of the associated discovery method without damaging its inductive scope in the long run. Moreover, as we tune n higher, the apparent "tightness of fit" of the hypothesis to the available evidence can be adjusted to suit the user's short-term temperament. We cannot, on the other hand, require some number of instances that is an increasing function of the input evidence, for this sort of requirement, like Hempel's confirmation relation, opens the possibility of eternal vacillation on a true hypothesis as the evidence increases.

4.5. Some Obvious Questions about Enriched Hypothesis Languages

If the hypothesis language H is purely universal and has at most finitely many constants then HEMP, CONSIST and the various NICODs can AE-U-identify the class of O all structures, each domain element of which is named by one of countably many constants of the evidence language. Moreover, *no* method can EA-U-identify O, or even that subset of O whose elements all have H-theories finitely axiomatizable in H. But it remains to be seen what the situation is when H is enriched with function symbols.

It is easy to see that HEMP fails to identify some element of O if even one unary function symbol is added to H. For consider the structure $M = \langle N,<,s\rangle \in O$, where N is the set of all natural numbers, and $<$ is the usual, strict order on N. Consider the following, universal hypotheses true in M:

1. $(x)[x<s(x)]$

2. $(x)(y)[x<y \text{ ---> } -y<x]$

3. $(x)(y)(z)[x<y \ \& \ y<z \text{ ---> } x<z]$

Finally, let e be any finite set of basic statements true in the structure. Let M' be any structure for the hypothesis language that satisfies e and whose domain elements are all named by constants in occ(e). The function s of M' that interprets s must be total and closed on the domain of M', by the definition of a relational structure. Since the domain is finite, the finite, directed graph of s must have a "circuit". That is, for some $n \leq |occ(e)|$ and for some object a in the domain of M, applying s for n times to a brings us back to a again. The circuit, itself, is the sequence $\langle a,b_1,\ldots,b_{n-1},a\rangle$, where b_i is the value of i applications of s to a. To satisfy hypothesis (1), the structure must be such that $a<b_1<\ldots<b_{n-1}<a$. So if the structure satisfies the third hypothesis (transitivity of $<$), then $a<a$. But then the structure does not satisfy hypothesis 2, for $a<a$ is a counterinstance. So for any e true in M, e fails to confirm these three truths about M. So there is no point at which they will be included in HEMP's conjectures. Hence, HEMP cannot AE–U–identify M, which is an element of O.

While HEMP can fail to eventually conjecture some universal truth, CONSIST and NICOD can fail to eventually reject, once for all, some universal falsehood. When the hypothesis language has function symbols, the relation of consistency is undecidable.[73] CONSIST and NICOD must always try to strengthen their current conjectures by adding sentences to Hyps(n), on pain of not eventually sticking with some general truth. But each time CONSIST or NICOD adds a sentence to Hyps(n), it runs the risk of having added a sentence refuted by the available evidence, for CONSIST and NICOD cannot wait forever to make another conjecture. But CONSIST and NICOD cannot even *enumerate* the sentences consistent with the evidence in an effective manner.[74] Moreover, any bound imposed on this test will lead to the possibility of conjecturing a falsehood. Since there are infinitely many hypotheses equivalent to any given hypothesis, the possibility of adding a sentence equivalent to

---

[73] For if consistency were decidable, then by skolemization, general first order validity would be decidable, which contradicts Church's theorem.

[74] If there were such an effective enumeration, it could be used in along with a complete, consistent proof theory to decide consistency, which is absurd.

a sentence already in h can arise infinitely often, and if the refutation proofs for these equivalent formulations exceed the resource bound infinitely often, CONSIST and NICOD will both fail to eventually reject a false h.[75]

But the knowledge that three procedures do not solve a problem (however suggestive it may be) falls infinitely short of knowledge that no procedure can solve the problem--- just as two fallacious proofs do not establish the negation of the intended result. So it would be interesting to discover whether it is possible to AE-U-identify the set of structures whose individuals are named by evidential constants when the hypothesis language includes function symbols. I leave this question open, although I expect a negative result.

[75] Of course, this scenario is not the only possible one. If we try to weed out equivalent formulations, there is still the possibility that adding a sentence logically independent to the false h to the rest of the previous conjecture will result in a conjecture that entails h.

Chapter 5
The Complexity of Discovery

In the previous chapter, the procedures HEMP, CONSIST, and NICOD were all shown to be general solutions to the problem of AE-converging to a theory that is complete over the purely universal, function-free sentences true in a a structure. But recall from chapter two that generality is only one criterion by which to judge a discovery method. Another concern is computational efficiency. And it is clear that the simple-minded "enumerate-and-test" procedures of the previous chapter are in some sense very inefficient.

But while the inefficiency of the previous chapter's proposed algorithms seems fairly obvious, the design of more efficient methods demands that we be more explicit about what efficiency is. Currently, there is no formal theory of the efficiency of general solutions to the AE-inference problem posed in the previous chapter. There is, however, a powerful and successful theory of the computational complexity of standard (terminating) computations. Robert Daley and Carl Smith have also developed a precise theory of the complexity of EA-convergent computations. In light of these proposals, I attempt to develop a new theory of complexity for the more general case of AE-convergent computation. I do not entirely succeed, but it is interesting to see how some plausible attempts fail. These failures counsel caution in speaking about the efficiency of convergent discovery procedures.

5.1. Standard (Short-Run) Complexity Theory

The point of standard complexity theory is to mathematically characterize the computational difficulty of effectively soluble problems. But the difficulty of a problem is a matter of the cost of computing its solution. An easy problem is one that is solved by a program that consumes few resources. A difficult problem has no easy solution. Program efficiency is a matter both of the amount of resources consumed and the difficulty of the problem solved. That is, a program is efficient if the problem it solves admits of no solution that is far less costly to compute.

Formally, let $\{\phi_i: i\epsilon N\}$ be an acceptable numbering of the Turing computable functions from input strings to output strings.[76] In standard complexity theory, a *problem* is a partial recursive function f and a *solution* to problem f is any index j such that $f=\phi_j$. Indices can be thought of as Goedel numbers for computer programs. Hence, a solution to a problem is a program that computes it. An *instance* of a problem is any finite string of symbols over the fixed input vocabulary.

The computation of program P_i on a given problem instance σ expends some resources. For example, a Turing machine that halts on a given input must execute some number k of state transitions. This measure is called *Turing time*. *Turing Space* is the number of distinct tape squares visited at least once during the computation.

Consider Turing time. It can be computed for a given Turing machine T_i on input σ as follows: "Simulate T_i on input σ, and add one to a counter c for each simulated step. When $T_i(\sigma)$ halts, return c". Notice that this procedure returns a value on n just in case T_i does. A similar construction can be given for Turing space. It is also decidable, for any i,n and k, whether the time or space consumed by machine T_i on input σ is no greater than k. Just modify the above construction to check at each stage whether c=k+1. If so, return 'no' immediately. If the halting state is reached before c=k+1, return 'yes'.

Every proposed resource measure for computer languages can be computed by simulating the computation of the measured program in the manner just described. Accordingly, Manuel Blum takes this relationship to be axiomatic of the very notion of a complexity measure. That is, let $\{\phi_i: i\epsilon N\}$ be an acceptable programming system. Let $\{\phi_i: i\epsilon N\}$ be a recursive enumeration of partial recursive functions. $\{\phi_i: i\epsilon N\}$ is a *complexity measure* for $\{\phi_i: i\epsilon N\}$ just in case

$\Phi_i(\sigma)$ is defined if and only if $\phi_i(\sigma)$ is

$\Phi_i(\sigma)\leq k$ is a computable predicate over all i,σ,k.

No one really believes that these postulates are sufficient conditions for an intuitively adequate complexity measure.[77] But under the more plausible assumption that they are necessary conditions, they provide a convenient way to prove surprising facts about all possible complexity measures at once.[78]

---

[76] i.e., there is an index u for the universal machine and the numbering satisfies the s-m-n theorem. C.f. [Machtey78] for details.

[77] For obvious counterexamples, see [Machtey78], p. 143.

[78] C.f. [Machtey78] for proofs of the "gap", "compression", and "speedup" theorems.

Standard complexity theory also assumes a natural valued *size* function $|\sigma|$ over all possible input strings σ. In applications, the size of a string is always taken to be the number of symbols occurring in it. The *worst-case* complexity measure is a function defined on the natural numbers such that

$$W_i(n) = \mathrm{MAX}\{\Phi_i(\sigma): |(\sigma)|=n\}$$

That is, the worst case complexity of a program for a given input size is just the maximum complexity over all input strings of that size. By taking the mean rather than maximizing, one obtains what is called the *expected* complexity of a program with respect to Φ and μ.

Program P_i is *as easy to compute as* program P_j just if for all but finitely many x, $\Phi_i(x) \leq \Phi_j(x)$. One might therefore expect the complexity of a *problem* ϕ to be definable as the Φ_i such that P_i is the easiest solution to ϕ. But Manuel Blum has shown that for any measure satisfying his two axioms there is a problem such that for any solution to this problem there is an easier solution according to the measure.[79] That is, for each measure there is a problem with *no maximally easy solutions*. So the complexity of a problem is not definable in terms of maximally quick solutions.

Even if there is no easiest solution for a problem, upper and lower asymptotic *bounds* on the complexities of its solutions can be sought. This fact has led to interest in complexity *classes* of problems. A problem ϕ is said to be in complexity class C_g just in case the worst case complexity of some solution to ϕ is bounded almost everywhere by g. It is typical to study unions of such classes rather than the classes themselves. For example, a problem is said to be linear (polynomial, exponential, etc.) if its complexity is bounded almost everywhere by some linear (polynomial, exponential, etc.) function. By establishing asymptotic lower bounds, we can discover that a given problem is *not* in one of these classes.

5.1.1. P, NP-completeness, and Tractability

The current consensus among computer scientists is that polynomial problems are tractable, while non-polynomial problems are intractable. Therefore most practical applications of the theory center on the question of membership in the class P of polynomial problems. Hence, it is of interest to find non-polynomial lower bounds, which imply non-membership in the class, or polynomial upper bounds which imply membership in the class.

---

[79] [Machtey78], p. 155.

In the case of many problems of importance to industry, combinatorics, and number theory, it is difficult to show whether the problem is polynomial or not. Many such problems seem to require "intractable search". That is, every proposed algorithm is forced, for infinitely many inputs of different sizes, to apply a fast (polynomial) test to each element of an search space whose size increases exponentially as the size of the input increases.

This intuitive picture of "easy test, hard search" is captured elegantly in the computational formalism of nondeterministic computation. Such a problem can be solved in polynomial time by a device that can non-deterministically "guess" the correct element of the space and then verify its guess in polynomial time with the easily computed test. So "search problems" of this sort are all in the class NP of problems that are soluble in polynomial time by a non-deterministic device.

Roughly, a problem is complete for a class if its easy solution would imply the easy solution of every problem in the class. Technically, a problem f reduces polynomially to a problem f' just in case there is a polynomially computable function g such that for each input σ, $f'(g(\sigma)) = f(\sigma)$. That is, given a fast (polynomial) program G[n] for g and a fast (polynomial) program F'[n] for f', the cost of program G[F'[n]] is the composition of two polynomially bounded functions and hence is itself polynomial (i.e. "fast").

A problem is said to be NP-complete just in case every problem in NP is polynomially reducible to it. So intuitively, an NP-complete problem is as "hard" as any problem in NP. Hence, if any problem that *seems* to require intractable search really does *require* it, all NP-complete problems do. Unfortunately, one of the major open questions in the foundations of mathematics and computation theory is whether any problem that seems to require intractable search in its solution actually does require it (i.e. whether P is not identical to NP.) But so many difficult and counterintuitive propositions have been shown to follow from the supposition that P=NP that many computation theorists suspect its falsity.

Regardless of the truth of this proposition, to show that a problem is NP-complete is to show in a very robust sense that no *known* technique will solve it in polynomial time, for as soon as any NP complete problem falls to such a technique, they all do at once. And this has not yet happened, although many hundreds of familiar and practical problems have been shown to be NP-complete [Garey79].]

5.2. Limiting Complexity

Our primary interest is in the complexity of AE-convergent computations. AE-convergent computational devices needn't be any different than standard ones. AE-convergence is, rather, a novel input-output convention for the same sorts of devices. On the standard view, a Turing machine's input is a finite string written to the left of its read-write head on an otherwise blank tape, and its output is whatever is in this position on an otherwise blank tape when the device is in its halting state. According to the convention of AE-convergence, on the other hand, the output of a device can be an infinite, non-R.E. set, and its input can be an infinite, non-R.E. sequence. So while the standard convention associates each Turing machine with a (finitary) function, AE-convergence associates each such machine with a map from infinite sequences to infinite theories.

Unfortunately, standard complexity theory makes sense only with respect to the standard input-output convention, under which a device can expend only finitely many resources before producing an output. Limiting computations, on the other hand, are always infinite in duration. Since the adequacy of HEMP, NICOD, and CONSIST is assessed in the limit according to the convention of AE-convergence, their elegance cannot be assessed directly within the standard "*short-run* theory of complexity.

What is required, then, is a theory of the complexity of AE-convergent computation that is sufficiently "like" the standard theory to be of interest. Unfortunately, I am aware of no attempt in the literature to formulate such a theory. But the issue of characterizing the complexity of EA-convergent computation has already been addressed by computer scientists interested in inductive inference.

Feldman seems to have been among the first computer scientists to be concerned with complexity in inductive inference, but his notion of complexity is restricted to the syntactic complexity of conjectures. He seems not to have addressed the question of the computational cost of arriving at a conjecture. Much less does he raise the issue of characterizing the complexity of EA-convergence itself. But his student, J.J. Horning, raised the latter question quite explicitly.

> A more realistic theory of inference should include computational cost [in addition to the complexity of the grammar converged to] in its definition of optimality, reflecting the fact that in most applications there are trade-offs among the cost of computation, the cost of further sampling, and the cost of guessing incorrectly.
>
> Short of developing a general theory of the cost of inference, one

> might test various heuristics which lead to nearly optimal solutions at substantially lower cost.[80] [1969, p. 152].

While Horning simply raised the question, other computer scientists have attempted to address it. In this section, I discuss some of this literature on the complexity of inductive inference.

5.2.1. Short-Run vs. Long-Run Complexity

In standard computation theory, the first order of business is to distinguish a problem (i.e. a function) from its solutions (i.e. the programs that compute it). That there is a real distinction to be made is evident from the fact that uncountably many problems have no solutions, while all soluble problems have infinitely many distinct solutions.[81] In the case of limiting computation, there are at least three sorts of entities to keep straight. First, there are identification problems, as described in chapter two. Second, there are conjecturing behaviors (functions from finite evidence sequences to hypotheses) that solve these problems. Third, there are the methods (programs) that compute these conjecturing functions. There may be no conjecturing behavior that solves a given identification problem. But if there is one, there are infinitely many distinct ones.[82] Some conjecturing behaviors are uncomputable, and the computable ones are computable by infinitely many distinct programs.

One way to bring complexity theory to bear on the study of an EA-identification problem is to investigate the complexities of particular conjecturing behaviors that solve the problem. Since a conjecturing behavior is just a computable function, we can investigate its complexity in light of the standard theory of complexity that was reviewed in the preceding section. For example, Gold [gold78] and Angluin [angluin78] study the minimal automaton inference problem in just this way. Angluin also assesses the complexity of a particular conjecturing behavior that solves the problem of inferring "pattern languages" in the limit [Angluin80].

Such results are of interest in their own right. It is not a trivial matter to discover that a popular or obvious approach to the solution of an inductive problem is NP-

---

[80] The reader may well wonder what Horning means when he says that a heuristic could "lead to nearly optimal solutions with substantially lower cost" in the absence of a general theory of the cost of EA-convergent computations. This question is particularly acute given that the heuristic admittedly has a different limiting and hence short-run complexity from the exhaustive procedures to which it is compared.

[81] As usual, I am assuming that the programs belong to an acceptable programming system.

[82] Notice that any finite variant of a behavior that solves an EA or AE-convergent inference problem also solves the problem.

complete. This knowledge can save much wasted effort that might have been expended in applying ordinary techniques to find a worst-case polynomial implementation of the approach.

But on the face of it, such results are about the short-run complexities of particular *approaches* to solving the EA-identification problem in the limit. It is part of common wisdom that one can always take a wrong-headed or inefficient *approach* to a task that is intrinsically easy. So despite the power and interest of complexity results concerning particular conjecturing behaviors, there remains a further question: what do they tell us about the *intrinsic* difficulty of the limiting identification problem whose solution is our ultimate objective?

Perhaps the most straightforward answer to this question is that an EA-identification problem is tractable if and only if there is a tractable conjecturing behavior that solves it. If we formalize the tractability of conjecturing behaviors in the standard way as polynomial computability, we have a clear definition of tractability for EA-identification problems. Under this definition, a proof that a limiting problem has a polynomially computable solution provides an upper bound on the intrinsic complexity of the limiting problem. But a lower-bound proof for a particular conjecturing behavior is not a proof of a lower bound on the intrinsic complexity of the limiting inference problem the behavior solves--- any more than a proof that my favorite sorting algorithm is slow is a proof that sorting is an intractable problem.

But this simple and natural definition is subject to a serious formal difficulty. For according to it, *all* soluble EA-identification problems are tractable. This fact can be seen by means of what is often called a "padding" argument. Let M be a method that solves EA-identification problem P in time bounded by an exponential, computable function f, but by no polynomial function. Now define the *f-Fabianization* of M as the following program. Given evidence e, set n := the size of e. Set k := the least natural number greater than or equal to $f^{-1}(n)$. Next, set e':= the initial evidence segment of length k. Finally, simulate M on input e'. We must assume that the length of the evidence can be found in time polynomial in the size of the evidence, but it is hard to imagine a sensible input size measure that does not have this property. Then k can be computed in polynomial time using a log table and some arithmetic. Finally, when M is simulated on an input of size $f^{-1}(n)$ it uses resources only polynomial in n, for M uses only f(n) resources on an input of size n. The f-Fabianization of M is a polynomial procedure that solves any EA-identification problem solved by M. Since M is arbitrary, the same construction

works for any exponential function, so every exponential EA-identification problem is indeed polynomial.

It is easy to see that all problems are polynomial in this sense. All that is required is to find, for each bound f, an upper bound on f whose inverse is computable in polynomial time. So we have the result that all soluble EA-identification problems are tractable. Since the major point of a complexity theory is to assess the inherent difficulties of distinct problems, this result is fatal to our definition of tractability for EA-identification problems.

The Fabianization of M is named after Fabius Cunctator, who conquered Hannibal by delaying. The name is appropriate, for the more the Fabian methods delay, the more efficient they are deemed in light of the complexity concept under discussion. The difficulty is easy to spot. An arbitrary delay in the point of convergence is not penalized if we measure only the resources consumed in generating each conjecture. But these conjectures can be made arbitrarily easy to produce by a sufficient delay in the point of convergence. So all EA-identification problems are easy.

One way to forestall this trivialization is to impose a bound on the amount of evidence that may be seen before converging to an hypothesis on each presentation of each world.[83] Since Fabian strategies must see much more evidence before converging than more sensible strategies, any bound on the amount of evidence that may be seen before converging must bound the extent to which a Fabian strategy may delay its convergence point, and hence precludes the trivial, arbitrary speedup attainable by appeal to such methods. According to this account, problems with tighter bounds are intrinsically more difficult than those with lenient bounds, which seems sensible. On the other hand, it is not easy to see how to go about selecting such bounds to study.

A related approach is to augment each EA-identification problem with a suitability relation. A method solves such a problem if and only if it can identify every world in the problem's proposed scope *and* each of its conjectures is suitable for the input evidence for which the method generates it.[84] For example, we may think of the minimal automaton inference problem as the problem of inferring a minimal state acceptor for an arbitrary regular set with the added, short-run constraint that each conjecture must be a minimal state acceptor consistent with the current sample. No

---

[83] There is no reason to assume that the same natural number must bound the amount of evidence seen in each world or in each ordering of the evidence from a given world. Rather, the bounding function g may be a function from infinite evidence presentations to natural numbers.

[84] This proposal was essentially suggested to me by R. P. Daley in a private conversation.

solution to this problem can afford to ignore evidence in the short run, so the Fabian behaviors are excluded from the complexity assessment. This limiting problem may therefore be said to be NP-complete in light of the results of Gold and Angluin alluded to above.

Notice that the limiting part of such a composite, long-run/short-run problem need not be a silent partner to the suitability relation it is paired with. There is no reason to assume that every conjecturing behavior that abides by the assumed suitability constraint must solve the identification problem in question. Problems such that every conjecturing behavior that accords with the suitability relation solves the identification problem in the limit are simply special cases.[85]

I began this section with the opinion that it is a confusion to define the complexity of an identification problem as the complexity of a conjecturing behavior or class of conjecturing behaviors that solves it. This opinion was based on the observation that there might have been an easier way to solve the problem than the one chosen. But the Fabian methods show that there is something trivial about this observation: one can *always* find an easy solution to such problems, in the straightforward sense of "easy" that we have been considering.

Our two responses to this trivialization amount to a proposal to consider only a limited range of conjecturing behaviors that solve an identification problem in the assessment of its complexity. One way to restrict the class of solutions to a limiting problem is to place a bound on the amount of evidence that may be read in each world on each presentation of that world. But we observed that it is not obvious which such bounds to study. Another way to restrict the range of behaviors to be considered is to propose a suitability relation and to require that each conjecture be suitable. Notice that this is just what Gold and Angluin do when they prove that finding a minimal-state acceptor consistent with a given sample is an NP-complete problem. I will present similar results myself, in chapters six and seven. Perhaps such results are the best that can be done in the absence of more reliable intuitions about what limiting complexity is.

---

[85] For example one can fail to EX_*-identify a minimal acceptor for a regular set even if one conjectures a minimal acceptor consistent with the sample at each stage. The conjectured acceptors may all be renaming variants of one another so that EX-convergence is never achieved. On the other hand, any procedure that produces such hypotheses is guaranteed to BC_* identify a minimal acceptor for any regular set.

5.2.2. The Daley/Smith Approach

Recall the difficulty of the previous section: if an EA-identification problem is in a complexity class if and only if it has a conjecturing behavior that is in that class (in the standard, short-run sense), then every EA-identification problem is tractable. The difficulty stems from the fact that the effort required by an arbitrary device to converge to a correct hypothesis can be smeared out to an arbitrarily low growth rate as the evidence size increases. An obvious observation at this point is that although the conjectures of a Fabian method are easier to produce at each stage, its overall convergence is much delayed. So if we measure the *total* effort spent before converging rather than just the effort spent in producing each conjecture, the Fabian gambit is defeated. This is because the Fabian machine must do at least as much as the method it simulates would have done to achieve convergence. In fact, it does far more.[86]

Daley and Smith [Daley84] study measures of this kind and call them *area under the curve* (AUC) measures. More precisely, let $\{\Phi_i : i \in N\}$ be a Blum complexity measure, and the let the problem be to EX_*-identify each function in some subset F of the partial recursive functions. Let $f|n$ denote the restriction of function f to the natural numbers from one to n. Define the *modulus* $\mu(M,f)$ of M on f to be the length of the graph of f seen by M at the time M converges to an hypothesis. We assume that the graph of the target function is presented sequentially, in the order of its domain. Finally, we we can define the area under the curve (AUC) measure of M on function f to be the sum of $\Phi_M(f|n)$ for n=1 to $\mu(M,f)$. The figure illustrates why "area under the curve" is an apt name for measures of this sort.

Does this sort of measure provide an adequate, general assessment of AE-convergent complexity? Unfortunately, the very feature that defeats the Fabian gambit also leads to an intuitive objection. In the standard theory of complexity, the effort spent before the halting stage is reached is measured. Analogously, the AUC measure counts only the effort expended before the convergence point is reached. This attention to the point of convergence ensures that the Fabian methods are billed for their foot-dragging. But in the standard theory, there is no objection to ignoring effort expended after the halting state is reached because, intuitively, there *is no such effort to ignore.* When the halting state is reached, the machine is viewed as having stopped. But in EA-convergent computation, there is *infinitely much* effort expended after the convergence point is reached, rather than none at

---

[86] i.e., it must simulate M and repeat M's actions by a "large" factor of f.

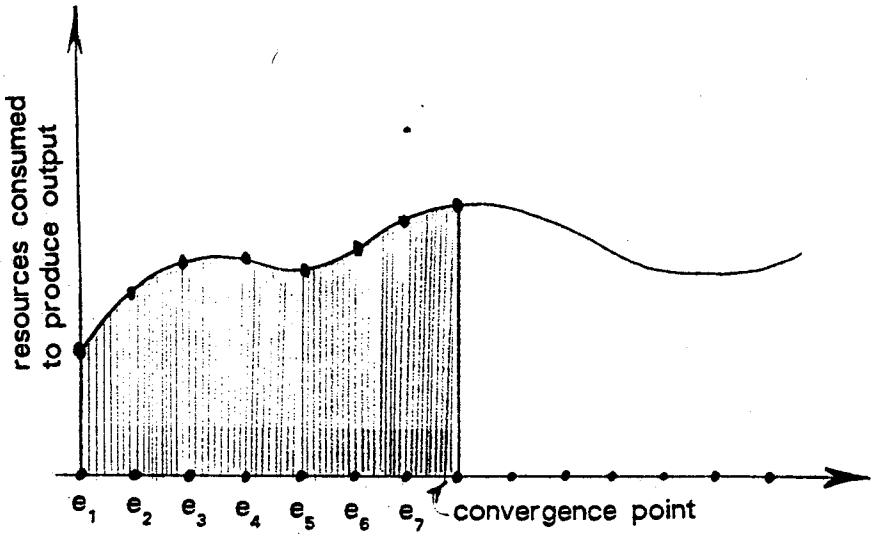


Figure 5-1: Area under the Curve Measures

all. There would be no intuitive difficulty if the user of an IIM could, intuitively, tell when it has converged so as to turn it off. But there is no effective manner for doing so, or limiting computation would be no more powerful than ordinary computation. We can imagine that one IIM could expend less effort than another before converging, but much more at each stage thereafter. If the user cannot tell when to turn off the device, then there seems some justification in his choice of a machine that uses fewer resources *almost everywhere* rather than one that uses fewer resources only finitely often. This intuition is underscored by the fact that recursion theorists typically focus on properties that hold almost everywhere rather than on those that hold only finitely often.

It should be mentioned that Daley and Smith do not require that *every* measure of EA-convergent complexity ignore the effort expended after the convergence point. They require only that such a measure converge to a value in a world if and only if the measured method does, and that it be decidable in the limit whether the measure takes on a given value on a given function. These axioms permit the measure to converge later than the measured method, and therefore to change its mind some finite time after the measured method converges. So the measure's assessment can take into account some of what the method does after the point of convergence.

Recall that the total effort expended to converge to a conjecture is relative to the order in which the evidence is presented. It is a matter of common sense that one can fool a method arbitrarily long by withholding crucial information. And insofar as a method's convergence is delayed, the AUC measure of the method's resource consumption increases. This raises a philosophical objection to such measures when

possible worlds are sets and the evidence consists of finite samples of these sets (as in many language acquisition problems).[87] A complexity measure should guide us in our choice of one method over another at the outset of inquiry. But AUC measures are essentially relative to a canonical presentation order for the evidence. Hence, we cannot compare the relative efficiencies of methods unless we can discover the order in which we would have encountered the total evidence in each possible world--- a hopeless inductive problem in its own right. But if we do know the canonical presentation order for the evidence antecedently, then we have tacit negative evidence at our disposal after all (i.e. if an expected string does not come up in its appointed position, it is not in the language). As Gold has shown, such problems are much easier to solve. So either we are choosing a method for a problem much harder than the one we actually face, or we can't choose the applicable AUC measure to evaluate candidate methods until we solve a harder inductive problem than the one we are choosing a method to solve. Either horn of the dilemma raises questions about the motivation of AUC measures in the case of language acquisition from positive evidence.

Finally, there is a serious philosophical question about how to define *tractability* in terms of Daley/Smith complexity measures. In standard complexity theory, a problem is tractable only if its worst case complexity is bounded by some polynomial function. In this theory, a problem instance is a finite string that has a finite size, and only finitely many distinct problem instances have the same size. Since the sizes are natural numbers, we can speak of the greatest resource consumption of a program over all inputs of this size, and then we can speak of a bounding function over this worst-case resource consumption for all but finitely many problem instance sizes. This polynomial growth rate in resource consumption is essential to the standard explication of computational tractability.

Now consider the limiting case. Daley and Smith name their complexity classes with functionals rather than functions, so that an identification problem C (i.e. a class of functions) is in complexity class ψ just in case there is a method M such that for all but finitely many functions f in C, the Daley/Smith complexity of M on function f is no greater than $\psi(f)$. In this case, problem instances are infinite functions on an enumerable domain. But it makes no sense to speak of ψ as "polynomial" unless each function can be associated with a natural number representing its "size", as in the standard theory. Intuitively, there ought to be some reason to expect a "bigger" function to require more effort (according to the AUC measure) to identify. It is not obvious how such a measure might be defined.

---

[87] I am indebted to R. P. Daley for what I understood as the following argument.

Indeed, Daley and Smith have shown that if there is such a notion of function size, it is radically ineffective. That is, there is no effective device that converges to a size value for every partial recursive function such that only finitely many functions are assigned the same size in the limit (p. 25). This result shows that if function sizes behave anything like the usual problem instance sizes familiar in the theory of NP-completeness, the assignment of a size to a function must be hopeless by mechanical means, even in the limit. But there remains the question whether there is some other way to obtain a definition of tractability in terms of Daley/Smith measures. The next section discusses four possible approaches.

5.2.3. Four Approaches to Defining Tractability

First Approach

One way to obtain an analogue to the usual complexity class hierarchy in terms of a Daley/Smith measure is to define complexity classes independently of any size ordering on problem instances. Let S be the range of a functional that names a Daley/Smith complexity class. So S is a set of natural numbers. Say that S is linear (polynomial, exponential, etc.) just in case S is the range of a linear (polynomial, exponential, etc.) bijection on the natural numbers. The resulting hierarchy is not trivial. For example, consider the class of polynomial sets. Evidently, the set of natural numbers is polynomial, for the identity function is a bijection with the natural numbers as range that is bounded by a polynomial function. Now let S be the range of the function $\lambda n(2^n)$. No bijection with S as range can be bounded almost everywhere by any polynomial function.

Although the resulting hierarchy is not trivial, it does promise to be very coarse. For example, if an enumerably infinite set S is linear (polynomial, etc.) then for any non-linear (non-polynomial, etc.) set S', SUS' is linear (polynomial, etc.). This is in sharp contrast to the standard, short-run approach, in which any problem that "interleaves" two problems in different complexity classes has the complexity of the more complex constituent.[88]

Second Approach

Another obvious approach is to define sizes for functions, but to abandon any hope of computing the size of a given function, even in the limit. Given a fixed

---

[88] That is, let $f(n)=n$ if n is even and 2^n otherwise. This problem is non-polynomial, but the problem whose domain is restricted to even numbers is linear.

programming system, each such function has a program of least size, where the size of a program is arrived at by counting its symbols. So we could let the size of a function be the size of the smallest program that computes it.

This proposal agrees nicely with the apparent aim of Daley and Smith to make the theory of EA-convergent complexity as analogous to the standard theory as possible. As in the standard theory, for example, there are only finitely many distinct functions of any given size. Hence, the worst-case complexity of a machine with respect to problem size n is defined exactly when the machine converges on each problem instance of this size.[89] But this fact also tells us (in light of the result of Daley and Smith cited earlier) that the size of a function cannot be assigned by any effective procedure, even in the limit.

Another difficulty with this approach is that there is no obvious connection between the minimum program size of a function and the work required to identify it. In the standard theory, a larger instance of a graph search problem means more arcs and nodes, and exponentially more paths to search. Adding larger numbers means comparing and carrying more digits. Checking larger numbers for the property of being prime implies more tests of possible factors. Now consider the obvious method for identifying the primitive recursive functions in the limit. It employs an enumeration of primitive recursive indices and always conjectures the least index consistent with the given evidence. The Daley/Smith complexity of identifying a function of size n will depend, then, primarily on the position in which its first program appears in the enumeration and on the computational resources expended in testing each hypothesis up to this point. There seems little reason to suspect that either of these quantities should depend upon program size.

A possible response runs as follows. Let M be the usual enumeration method employing some, complete non-redundant enumeration σ of the primitive recursive functions. Regardless of the order of σ, we know that for each n there is an n'>n such that $\sigma_n < \sigma_{n'}$. It is plausible to take the effort expended by stage n to be the number of hypotheses considered by stage n. Hence, any asymptotic bound on the worst-case complexity of an arbitrary enumeration method must rise with function size after all.

But it is still the case that the size of a function has nothing to do with anything like the size of the "input" an inductive procedure receives when attempting to

---

[89] Daley and Smith assume canonical function presentations, so I shall not distinguish a function from its canonical presentation.

identify the function. In the standard theory, the size of an instance is usually taken to be the number of symbols occurring in the machine's input. The intuitive connection between increased computation time and increased input size is that it takes longer to "chomp on" a bigger input because there is more input to "chomp on". In the limiting complexity theory just proposed, there is no such intuition to favor one "size" function over another. Therefore, if complexity classes are not preserved under the arbitrary bijective transformation of function sizes, the theory just proposed would be too arbitrary to explicate intuitions of inductive efficiency.

Third Approach

In the last approach, the assumed size measure on functions seemed arbitrary because it did not reflect the input received by a program attempting to learn a given function. An obvious candidate for the size of the input to an inductive device with respect to an evidence sequence is the length of the initial segment of the sequence the device reads before converging to a conjecture. So a natural measure of the size $|f,M|$ of a function f (with respect to an IIM M) is the amount of evidence M reads when faced with a canonical enumeration of the evidence for f.[90] Next, let $\{\phi_i: i\in N\}$ be a Daley/Smith measure for IIM's. Let S be a set of functions, and let M identify S in the limit. Then define the worst-case complexity of M on S at stage n as

$$W_M(S,n) = MAX\{\Phi_i(f): |f,M|=n\}.$$

Define complexity classes as follows: $S\in C_g$ just in case there is an M that identifies S such that for all but finitely many $n>0$, $W_M(S,n)\leq g(n)$.

This theory exploits the intuition that the resources required to identify a function that requires more evidence to identify will increase asymptotically. Hence, one might expect a complexity hierarchy as in the standard theory.

But unfortunately, the hierarchy is threatened with trivial collapse in light of the "Fabian strategy" discussed earlier. Recall that the scope of the Fabianization of a machine is identical to that of the machine Fabianized. Moreover, we have seen that the modulus of g with respect to the f-Fabianization of M is greater than g(Mod(f,M)). That is, the f-Fabianization of M "inflates" problem size by more than a factor of f, without doing significantly more work than M, which it simulates on a cut-down evidence set in order to produce its conjecture. But an inflation of input size by a factor g(x) is the same as a deflation of complexity by the same factor.

---

[90] This basic idea was suggested to me by Clark Glymour in a telephone conversation.

So for any problem S, it seems that there is a radically Fabianized strategy that is "easy" to compute according to this theory.

Fourth Approach

The fatal defect in the previous theory is that each machine is permitted to assess the size of the task it faces--- and like people, some machines are prone to exaggerate the difficulty of their undertakings. A response to this defect is to put all the machines on a common standard by defining $|f|$ to be $MIN\{|M,f|: M \text{ can identify } S\}$, for each $f \in S$.

But this cure is worse than the disease. For example, let S be the set of all primitive recursive functions. Then for each f in S, $|f|=1$. For recall that S can be identified by an "enumeration method" that tests each primitive recursive index against the data until the index fails and the method conjectures the next index in the enumeration. To obtain a device that identifies S for which the modulus of f is unity, just construct an enumeration method whose first conjecture is an index for f. Clearly, a size measure under which all problem instances are of size one will not support an interesting hierarchy of asymptotic complexity classes.

5.2.4. Section Summary

We would like a theory of the complexity of EA-convergent complexity that permits us to define tractability non-trivially, that is invariant under the order of presentation of the evidence, that permits us to compare the intrinsic difficulties of all sorts of inductive problems, that can be applied *a priori* to help us evaluate inductive methods before using them, and that does not seem entirely arbitrary or conventional in providing these assessments.

So far, no single proposal has even remotely delivered all of this. The obvious concept of tractability with which we began was trivialized by Fabian methods. Bounds can be placed on the amount of evidence a method may read before converging to a correct hypothesis, but it is difficult to motivate the study of any particular such bound. The AUC measure seems like a way to counter this trivialization, but at the expense of being sensitive to the order in which the evidence is presented and providing no intuitive concept of tractability.

We began by pointing out that to prove that a conjecturing behavior that solves a problem is difficult is not to show that the problem *itself* is difficult. But given the difficulties that arise from our intuitions about what the intrinsic complexity of a

limiting inference problem is, the best course for the present may be to investigate the relatively clearer short-run complexity of particular conjecturing behaviors that solve the problem.

5.3. The Complexity of AE-convergent Computation

In seeking a theory of AE-convergent complexity, it is natural to ask first whether any one of the theories of EA-convergent complexity considered above applies to the more general case of AE-convergent computation. First, recall that the AUC measures are defined in terms of a unique conjecture at which point the device may be said to have converged, once for all, to an hypothesis. But a device can AE-identify a theory without doing so at a unique point in its conjecture sequence. Indeed, there can be a distinct "convergence point" for each equivalence class of sentences in the hypothesis language. Therefore, even if these measures were entirely unobjectionable on other grounds, none of them is applicable to AE-convergent computation.[91]

The simpler approach discussed at the beginning of the previous section would indeed apply to the case of AE-convergent computation without alteration. But, alas, it is trivialized by the Fabian methods. We can skirt this trivialization by studying beefed-up problems, but this is really just a way of changing the subject. The account of tractability is still trivial regarding the problems we wanted to study originally.

Since these approaches are representative of the literature on the question, AE-convergent computation still lacks anything like an adequate complexity theory. The purpose of this section is to examine some proposals for such a theory. We can always fall back on the study of augmented problems, just as in the EA-convergent case. But we ought to take a stab at a more general perspective, if only to see what sorts of new issues arise in characterizing tractability in the novel context of AE-convergent computation.

5.3.1. Computational Model

To begin with, we must settle the computational model for which the desired complexity measure is to be defined. A *theory identification machine* (TIM) is an oracle Turing machine that queries for evidence sentences and that conjectures finite sets of purely universal, function free sentences in a given hypothesis language.

---

[91] Not all Daley/Smith measures are defined in terms of the modulus of convergence.

HEMP, NICOD and CONSIST were presented as functions that take a finite set of evidence as input and that output some finite set of universal sentences. There is no conflict here, for any such procedure M can be converted into a TIM by tacking on a querying "front-end" and a buffer that saves all the evidence read so far and passes it as an input to M.

5.3.2. The "Milestone" Theory of AE-convergent Complexity

The Daley/Smith theory does not apply to AE-convergent computation because the modulus function is undefined in this more general setting. But a TIM must, in some sense, forever "approach" the theory (deductively closed set) to which it converges. This observation suggests that asymptotic complexity bounds should be sought over the resources consumed in achieving "milestones" of *approximation* to the theory converged to. Then instead of seeking a "size" for infinite inputs from which we can inherit asymptotic resource bounds on AE-convergent computations, we can instead base the asymptotes on the effort spent in achieving milestones of approximation of the theory converged to. Since a better degree of approximation to a given target theory is intuitively more difficult to achieve, such bounds would be quite naturally motivated.

To explicate the notion of "achieving milestones", a formalization of the notion of "achievement" is required. More precisely, say that

> TIM M *achieves* theory T at stage n on evidence presentation σ just in case for each n'>n, the conjecture of M on reading $\sigma_{n'}$ is consistent and entails T.

Notice that if M's conjectures are always consistent, M achieves every tautologous theory at every stage. Also, achievement by M at stage n is closed under logical consequence. If T is achieved by M at stage n and $T \models T'$, then T' is achieved by M at stage n. Now we can define an obvious analog to the concept of the modulus of EA-convergence.

> The *achievement modulus* of M with respect to T and σ is n just in case n is the least n' such that M achieves T at stage n' on presentation σ.

The general idea is to measure the resources consumed in achieving each of infinitely many degrees of approximation to the theory AE-converged to, such that each method that AE-converges to T achieves, *eo ipso*, each degree of approximation in the sequence. But to characterize this notion precisely, a concept

of degrees of approximation of a theory is required so that there is an infinite sequence of ever better approximations to *any* given theory. That is, we need a function m such that for each pair of theories T,T' in the hypothesis language H, m(T,T') is called the *degree of approximation* of T' to T. To be technically useful m should satisfy the following axioms:

1. If T=T' then $m(T,T')=\omega$
2. Otherwise, $m(T,T') \in \omega$
3. If $M[\sigma]$ AE-converges to consistent theory T then for each $n \in \omega$ there is a k such that the achievement modulus of T' by M on σ is k and m(T,T')=n.

where ω is the set of all natural numbers. First, each theory approximates itself "infinitely well". Second, any theory is approximated by a distinct theory to some finite (possibly zero) degree. Finally, if M AE-converges to T then each degree of approximation to T is achieved by M after some finite number of conjectures. The first two requirements ensure that there will be a totally ordered, countable infinity of milestones so that the total effort required to achieve each one can support an asymptotic bounding function of the sort familiar in standard complexity theory. The third postulate ensures that these milestones are not vacuous. That is, no device can AE-converge to T without achieving each finite milestone at some finite time. It would clearly be undesirable for a device to be capable of AE-converging to T while remaining stuck forever at some finite "approximation" of T.

Given that we have a notion of achieving a theory and a way to characterize the degree to which one theory approximates another, it remains to measure the resources expended in achieving some theory that approximates the one converged to by a given degree. Accordingly, let $\Phi_M(T)$ be like a Daley/Smith measure, only with the the modulus of M on σ replaced by the *achievement modulus* of M with respect to T and σ. Call such a measure an *achievement measure* for M. Intuitively, Φ measures the "total effort" expended by M in coming to achieve T on input σ, just as a Daley/Smith measure reflects the total effort expended by M in EA-converging to an hypothesis on σ. Like the Daley/Smith measure, this one is order dependent with respect to σ.

Given these pieces, the complexity of TIM M at milestone n on presentation σ is defined as follows:

$$C_M(n,\sigma)= \begin{cases} \text{MIN}\{\Phi_M(T'):m(T,T')=n\} \text{ if } M[\sigma] \text{ AE-converges to } T. \\ \text{undefined otherwise.} \end{cases}$$

That is, the complexity of M on sequence σ at stage n is the amount of resources consumed in achieving the nth milestone of approximation to the theory AE-converged to by M on sequence σ, if there is one.

An *inductive inference problem* is a set S of enumerable structures for H. For each structure R in S, let T(R) be the complete H-theory of R. Assume that the evidence presentation for each structure R is some canonical sequence ρ(R). This is analogous to the Daley/Smith assumption that function presentations are in canonical order.

Finally, we can introduce a hierarchy of complexity classes as follows:

> Problem S is linear (polynomial, exponential, etc.) just in case there is a linear (polynomial, exponential, etc.) function p and a TIM M that AE-identifies S such that for all but finitely many n, MAX$\{C_M(n,\rho(R)):R \epsilon S\}$ is defined and no greater than p(n).

Notice that if any C_M value is undefined at n, the MAX expression is undefined as well. But the MAX can also be undefined even if C_M is defined for each $R \epsilon S$. This happens when for each problem instance R in S whose nth milestone takes k steps, there is an R' in S whose nth milestone takes k'>k steps.

This theory is complicated, but its motivation is straightforward and it has some advantages. As in standard complexity theory, the complexity of a device is defined independently of whether the device *solves* the problem. And unlike the Daley/Smith account, this theory requires no "natural" concept of the "size" of a problem instance. Asymptotic bounds are based, rather, on the difficulty of achieving better approximations to the theory converged to. Notice that this feature also thwarts the Fabian machines that collapse the Gold/Angluin approach. Milestone size depends only on the structure of a conjecture and on the theory converged to, and Fabian tactics affect neither of these factors. Finally, unlike the second proposed revision of the Daley/Smith theory presented earlier, this account holds the potential of non-trivial complexity for finite problems (sets of structures).

5.3.3. Approximation and Verisimilitude

So far it has been assumed that there is an approximation function m such that $m(T,T')=\omega$ if $T=T'$ and $m(T,T')\epsilon\omega$ otherwise. The intuitive interest of the milestone theory of complexity rides heavily on how naturally this function reflects the "degree to which T' approximates T."

Notice that in assessing the complexity of *problems*, we are interested only in values of m(T,T') for which T is the H-complete theory of some structure R for H. So we may speak interchangeably of m(T,T') and m(R,T') when T=T(R). The restriction of m to complete theories in its first argument can therefore be thought of as a "measure" of the "truth-likeness" or *verisimilitude* of T' with respect to R, for it is intended to reflect the degree to which T' "fits" world R.

The notion of verisimilitude was introduced by Karl Popper in one of his many quarrels with what he took to be the methodological dictates of probability theory [1965, pp. 233-4]. Unfortunately, Popper's theory and its successors all take degrees of verisimilitude to be rational numbers in the unit interval. Hence, such theories do not satisfy the three axioms on approximation degrees laid down in the previous section. But following definition of m[92] does satisfy these axioms.[93]

$$m(T,T')=\begin{cases}\text{MAX}\{k:\ k\epsilon\omega \text{ and } T'\restriction k = T\restriction k\} \\ \quad\text{if a natural maximum exists;} \\ \omega \text{ otherwise;}\end{cases}$$

Intuitively, the nth milestone faced by a TIM that converges to T on this theory is to get each subsequent conjecture to be consistent and to entail all the consequences of T of length n. Or semantically speaking, the nth milestone is to get each subsequent conjecture to be consistent and to entail the set of all sentences of length n that are true in structure R.

This measure places a premium on accounting for short, informative hypotheses as

---

[92] This definition was suggested by Glymour in a private conversation.

[93] (1) Assume $T=T'$. Then $T\restriction k=T'\restriction k$ for each k. Hence no maximum such natural number k exists. Therefore $m(T,T')=\omega$. (2) Assume T is not identical to T'. Then there is some sentence s of minimal length that is in S but not S' or vice versa. Let the length of s be k. Then $m(T,T')=k-1\epsilon\omega$. (3) Finally, suppose M AE-converges to consistent theory T on σ. Then for each s in T, there is a j such that s is entailed by all conjectures after the kth one, and for each s' not in T, there is a j' such that s' is not entailed by any conjecture after the kth one. Since T is consistent, there must also be some least stage j" after which each of M's conjectures is consistent (or else j' does not exist). M achieves $T\restriction n$ at stage n just in case all subsequent conjectures are consistent with $T\restriction n$ and entail $T\restriction n$. Since $T\restriction n$ is finite, there is always some stage j_n by which each conjecture of M entails each sentence s in $T\restriction n$ (namely, $j_n=\text{MAX}\{j:\ s\epsilon T\restriction n$ and j is the first stage such that for all subsequent stages, the conjecture of M entails s$\}$.) Hence, $\text{MAX}(j'',j_n)$ is the finite achievement modulus of $T\restriction n$.

soon as possible. Also, to figure out one's eternal commitment to longer hypotheses seems intuitively to require more work than determining one's eternal commitment to short, syntactically simple hypotheses, especially considering that the number of hypotheses of size n rises exponentially in n.

There is also an important objection. No amount of improvement in the agreement of T's long consequences with the facts about R can possibly raise m(R,T) until each shorter sentence is accounted for. But intuitively, enough such improvements ought to increase the verisimilitude of a theory at least a little, even if some small consequence is yet unaccounted for. So this definition of m seems to unfairly penalize methods that prefer to examine big sentences long before considering some short one.

If the hypothesis language is closed under conjunction, then the last criticism is somewhat ameliorated by the fact that if a short sentence is not accounted for, all the longer conjunctions in which this sentence occurs as a conjunct fail to be accounted for. So not achieving earlier milestones can prevent the achievement of infinitely many later ones. But this consideration does not always preclude the previous objection, for if milestone n is not achieved, there may not be conjunctions of missing sentences of size n that are of every size greater than n.

A natural way to attempt to avoid counter-intuitive reliance on "artificial" syntactic features of hypotheses in the assessment of the complexity of inductive inference is to define m in terms of semantic concepts only. So what is desired is a definition of m(D,T') that satisfies axioms similar in spirit to those presented earlier, but that is defined in terms of a verisimilitude relation (whose first argument is a structure) rather than an approximation relation (whose first argument is a theory). The revised axioms are as follows:

1. If $T(D)=T'$ then $m(D,T')=\omega$
2. Otherwise, $m(D,T') \in \omega$
3. If $M[\sigma]$ AE-converges to T(D) then for each $n \in \omega$ there is a k such that the achievement modulus of T' by M on σ is k and $m(D,T')=n$.

It is readily seen that these axioms enforce the spirit of the previous ones when the first relatum is a structure rather than a theory.

But to find an intuitive definition of verisimilitude that satisfies these axioms is not so simple. For example, consider the following measure. Assume H has no

function symbols. Then every substructure of a structure for H is a structure for H. For any enumerable structure R for H, let Sub(R,n) be the set of all restrictions of R to structures of cardinality no greater than n. Then we can define

$$m(R,T)=\begin{cases} \text{MAX}\{n\text{: for all } R' \text{ in } Sub(R,n),\ R' \models T\} \text{ if it exists;} \\ 0 \text{ if there is no } n \text{ such that for all } R' \text{ in } Sub(R,n),\ R' \models T; \\ \omega \text{ otherwise.} \end{cases}$$

Roughly speaking, the nth milestone facing an inference device is to settle on a theory that is true in each substructure of R whose cardinality does not exceed n.

Unfortunately, this definition does not satisfy the third axiom if the function-free hypothesis language H can express existential quantification and has some non-logical predicate. Let D be a finitely axiomatizable structure in which there is but one domain element with property P. Let A be a finite axiomatization of T(D). A device that always conjectures A EA-converges and hence AE-converges to T(D). But each conjecture must entail the sentence (Ex)(Px), which is true in D. But for each k, (Ex)(Px) is false in some substructure of D of size k (e.g. in those substructures from which the unique object with property P is eliminated.) Therefore, M AE-converges to T(D) without achieving any milestones of "approximation" to T(D) on this account. This is exactly the vacuous situation the third axiom is intended to prevent.

The difficulty is apparently associated with existential quantification. But HEMP, NICOD, and CONSIST do not conjecture existential hypotheses anyway. So there is still a strong motivation to see how naturally this account works for purely universal, function-free hypotheses. But unfortunately, the third axiom still fails. Intuitively, the difficulty is this: AE-convergence to the purely universal theory for a structure D does not imply that for each n there is some finite point after which each conjecture fails to entail any sentence false in a substructure of D of size n. And if there are infinitely many points at which such sentences are rejected by M, the nth milestone is never achieved, even though M AE-converges to T(D).

5.4. Revised Ambitions

The milestone theory of AE-convergent complexity (based on the syntactic concept of theory approximation) may be formally nontrivial, but it is not free from objections.

First of all, the milestone theory, like the Daley/Smith approach, is relative to a canonical order for the evidence. So it suffers from equally serious objections when the problem is to infer a set from positive examples only.

Second, it is intuitively biased against AE-convergent devices that consider large hypotheses first. The semantic approach would alleviate the sense of syntactic bias, but I have discovered no formally adequate example of a semantically defined verisimilitude function. In short, the question is open whether the "milestone" approach to a hierarchy of asymptotic complexity classes is viable. The failure of two obvious attempts should not be taken as persuasive evidence that it is not.

Even if no plausible approximation measure can be found for the milestone theory just presented, the considerations that motivated this theory must be addressed by any adequate alternative. The first consideration is that AE-convergent complexity must somehow reflect the resources consumed in producing suitable conjectures in the short run. The second is that the notion of complexity must avoid trivialization by Fabian strategies that are easy to compute in the short run, but that take much longer to achieve any degree of approximation of the theory AE-converged to in the long run. The milestone theory is perhaps the most obvious way to try to balance these considerations, but it need not be the only way.

In the meanwhile, life must be faced without an explicit, universal theory of AE-convergent complexity. And it can be, so long as our theoretical aims are not too ambitious. Recall that in standard, short run complexity theory, complexity classes of problems (computable functions) are defined in terms of the resource consumption of programs that compute them. There is no interesting standpoint that is less general than that of the problem and yet more general than that of particular programs. But in each of the limiting complexity theories reviewed in this chapter, the situation is different. Between inductive problems and their solutions are *conjecturing behaviors*. More precisely, a *conjecturing behavior* is a total function from finite evidence sequences to finite sets of clauses. In standard complexity theory many programs solve the same problem. In the limiting theory, many programs compute the same conjecturing behavior, and many distinct conjecturing behaviors lead to a solution of the same inference problem. So in the

study of limiting complexity, we can take a natural, general perspective (of conjecturing behaviors) without taking the *fully* general perspective (of problems).

In the special case of TIMs, there is yet another natural, intermediate perspective more general than that of individual conjecturing behaviors but less general than that of problems. For any two conjecturing behaviors f,g, f is *equivalent* to g just if for any evidence sequence σ, $f(\sigma)$ is logically equivalent to $g(\sigma)$. Clearly, any device whose conjecturing behavior is equivalent to f has the same inductive scope as any device that has behavior f, but not conversely.[94] Now define a *short-run strategy* to be an equivalence class of conjecturing behaviors. Many conjecturing behaviors realize the same strategy, and many strategies solve the same inductive problem. These relationships are sketched in the following table:

Standard complexity theory:

problems
(functions)

solutions
(programs)

AE-convergent complexity theory:

problems
(sets of structures)

short-run strategies
(equivalence classes of equivalent conjecturing functions)

conjecturing behaviors
(functions)

solutions
(programs)

For example, recall the procedure CONSIST. It is defined with respect to some enumeration {Hyps(n)} of nested subsets of the hypothesis language. At each stage n, it conjectures the set of all clauses in Hyps(n) that are consistent with the first n evidence sentences in the given evidence presentation. So its conjecturing behavior is

$f(\sigma)$= {$s \in$ Hyps(n):s is consistent with the set of all evidence sentences appearing in σ}

Hence, the *strategy* of CONSIST is:

---

[94] Recall the Fabian conjecturing behaviors.

> For each finite evidence sequence σ, to conjecture some subset of Hyps(n) that is equivalent to the set of all s in Hyps(n) such that s is consistent with the set of all evidence sentences appearing in σ.

So although we are in no position to compare the efficiencies of all possible programs that solve a given AE-identification problem, we can at least compare the relative efficiencies of programs that pursue the same *strategy* in solving such a problem. That is,

> M is as efficient an implementation of strategy [f] as M' is just in case $\phi_{M'}$, $\phi_{M'}$ are equivalent to f, and for all but finitely many finite input sequences σ, $\Phi_M(\sigma) \leq \Phi_{M'}(\sigma)$.[95]

Moreover, one can say that

> a *strategy* is linear (polynomial, etc.) just in case some conjecturing behavior that accords with this strategy is linear (polynomial, etc.)

This definition permits us to speak of the NP-completeness of a strategy, for example. As we saw earlier, such a result implies nothing about the intractability of the overall inference problem addressed by this strategy. But it *does* tell us something very general and useful about the practical project of designing an elegant algorithm that implements the strategy under study: any known approach will involve an ugly search in infinitely many cases. Such a result can serve as ample warning that it will be hard to find an algorithm for the strategy that avoids the ugly search.

But it is also important to remember what this sort of analysis cannot do. We are not entitled on this approach to compare the relative efficiencies of programs that pursue distinct strategies (i.e. that compute inequivalent conjecturing behaviors). And the programs HEMP, NICOD, and CONSIST all pursue distinct strategies. Moreover, devices that rely on distinct suitability relations in formulating their conjectures will typically pursue distinct strategies, and hence be incomparable. Therefore, we have no general, theoretically sound basis for saying that one suitability relation is "better than" another for a particular theory inference problem.

---

[95] Where Φ is an assumed Blum complexity measure.

5.5. Complexity and Efficient Generation

It is reasonable to expect to find a device far more efficient than CONSIST in accordance with the sufficient condition just discussed. That is, some device with a conjecturing behavior equivalent to that of CONSIST must surely use fewer resources on each input. At stage n, CONSIST tests *every* sentence in $Hyps_n$ against the evidence. But as we shall see in the next chapter (c.f. section 6.7.1), this consistency test is likely to be intractable in the size of the hypothesis tested. Hypothesis size increases linearly with the size of the input (for a TIM reads one evidence instance at each stage, and hence has read n instances when it tests hypotheses of length n). Therefore, testing hypotheses gratuitously is extremely wasteful. And nothing prevents CONSIST from testing logically equivalent clauses. Hence, a device that does not test all logical variants of the same hypothesis could be expected to be more efficient, provided that its management of the equivalence check does not offset the effort saved in empirical testing. It would be better still if all sentences in $Hyps_n$ logically equivalent to a given sentence in $Hyps_n$ could be efficiently excluded from *consideration a priori* (e.g. without considering the evidence). Also, any h that entails a refuted hypothesis or that is entailed by an hypothesis that passes the relevant test may be ignored safely *a posteriori* (i.e. as a function of the current evidence). If these sentences need not be tested against the evidence, it is also clear that they need not be considered for any other reason.

From these considerations, it is clear that improvements of HEMP, NICOD, and CONSIST will have much to do with finding more elegant ways to *generate* certain restricted subsets of $Hyps_n$ in the *short run*. But standard complexity theory has evolved with particular emphasis on the tractability of *decision* problems rather than on the tractability of *generation* problems [Garey79]. The philosophical preoccupation with tests (as opposed to generators) that was encountered in Chapter One has to this extent infected computation theory as well. But does the computational complexity of the decision problem tell us anything of interest about the corresponding generation problem? This question has a general computational significance that transcends its particular application to the logic of discovery. Or rather, it would have such significance if it were a well-posed question. To pose it mathematically demands a formal characterization of finite set generation problems as well as a non-trivial, intuitively informative account of the resource consumption of their solutions.

5.5.1. The Problem of Generating Finite Sets

In formulating a theory of generational complexity, the first step is to characterize generation problems mathematically. It would not do to conceive of a generation problem as a single, finite set which we would like to generate, for this sort of problem can be solved with trivial ease by a device that "memorizes" the set in question and then prints it out on demand. But a less trivial situation is actually encountered in applications: given some arbitrary natural number k (and perhaps some other inputs of different types), the problem is to generate a some finite subset P_k of some infinite, R.E. universe U. So a generation problem $\{P_x\}$ is just a function from natural numbers to finite sets. Since this sort of "parameterized" generation problem is infinite, no trivial "lookup table" solution is generally available. Such problems arise naturally in the design of efficient AE-identification procedures. For example, P_k might be a result of removing all logically redundant sentences from $Hyps_k$ in the procedure CONSIST. Or it might be one of Shapiro's "refinement operators", where $P_{k,\sigma}$ is the set of all sentences at level k of the refinement graph that are refinements of sentence σ. Or finally, it might be a subroutine that generates a subset of $Hyps_k$ that is free of of logical redundancies.

Next, explicit conditions must be provided under which an arbitrary Turing machine is taken to have solved a generation problem. Assume that the machines in question have separate, write-only output tapes and a finite output alphabet Σ. Also assume a coding scheme that associates each element of the universe U with some finite string of characters in Σ. A machine is taken to have generated subset P of U just when it is in its halting state, and the write-only head is immediately to the right of a finite sequence of codings for elements of U, separated by single blanks. So a set is represented by a list (possibly redundant) of code strings for its elements. Such a machine is given a number n as input just in case it is placed in its initial state with its read-write head immediately to the left of a binary numeral for n. A machine solves $\{P_x\}$ just in case it outputs P_k in finitely many steps after receiving k as input, for any natural number k.

The major point buried in these technical conventions is that a machine has not made an output until it is in its halting state. Therefore, it must "know" that it has written down every element of P_k in finite time. It does not count, for example, if a machine goes on an infinite search and eventually writes down each element of the target set without terminating its search explicitly by entering the halting state. The motivation for this requirement is simple: a general logic of discovery must generate conjectures in finite time. Hence, any generator serving as a subroutine to

the overall conjecturing process must complete its computation in finite time and explicitly signal the main routine that it is finished.

5.5.2. The Complexity of Generating Finite Sets

Since a generation problem is merely a function with finite sets as values, one approach to a formal theory of generational complexity is to take the standard, worst-case Turing time measure over programs that compute the function.

Recall that the standard, worst case measure is based on the "size" of the input. A methodological convention in complexity theory is that input encodings should not "pad" input size so as to make a problem seem easier than it "really" is [Garey79] p. 10. Hence, it is considered unfair to take the size of an input natural number to be the number itself, for each number n can be coded in a binary numeral system by a string whose length is on the order of $\log_2(n)$. Let f(x) be any function on the natural numbers that represents a value based resource consumption measure. To switch to a numeral-length based measure is just to adopt a measure bounded below by the functional composition $r(x)=f(\log_2 x)$. Since the log function has an inverse, this is equivalent to

$$f(x) = r(\log^{-1}_2 x))$$

$$= r(2^x).$$

The composition of any monotonic, increasing polynomial function with an exponential function is non-polynomial. So basing the complexity measure on input length has drastic consequences for complexity theory. Indeed, problems that are NP complete on the numeral-based measure would be polynomial on the value-based measure.[96]

Each generation problem $\{P_x\}$ is associated naturally with a decision problem $\Delta(x,s)$, where $\Delta(x,s)$ holds of natural number x and element s of U just in case s is an element of P_x. The problem is to decide for any given x and s whether $\Delta(x,s)$. A solution to such a problem is a machine that computes its characteristic function. The same worst-case complexity measure applies to such solutions.

With one account of generational complexity in hand, what does the easiness of a generation problem say about the easiness of the corresponding decision problem (and *vice versa*?) To begin with, it is at not clear that there is a general procedure to convert a fast test procedure into a generation procedure that solves the

---

[96] E.g. the PARTITION problem. [Garey79], p. 91.

corresponding generation problem. An obvious attempt, for example, would be to enumerate the universe U effectively and to successively test each item in the enumeration. But this procedure lacks an effective criterion by which it can terminate its search of U without failing in some case to consider some element of the set to be generated. And it is not obvious how such a criterion might be obtained from an arbitrary test procedure and an enumerator for U.

It is certain, however, that the existence of an easy test does not guarantee the existence of an *easy* generator. For there exists a difficult generation problem whose corresponding test problem is easy. Let P_k be the set of strings of length k on some finite alphabet Σ. Clearly, the decision problem $\Delta(x,y)$ corresponding to P_x can be solved in time linear in the length of the inputs.[97] But any machine that solves the generation problem must print the answer. Hence, its resource consumption must be at least super-exponential in the input length.

But the converse is true: - if a generation problem is easy, then its corresponding test problem is as well. A generator cannot operate in polynomial time unless $|P_k|$ grows polynomially in log(k). And if a given program generates P_k in polynomial time, it requires only polynomially more time to check the membership of x in P_k; for in the worst case, only polynomially many comparisons need be made.

Each of these results is a shallow and relatively uninteresting reflection of the fact that if $|P_x|$ rises exponentially in $\log_2 x$, then any solution to the problem uses more than polynomial time just to print its answer. There is a legitimate sense in which any procedure that generates intractably large sets is intractable. But there is also a sense in which such a procedure can seem efficient; and the standard approach to the theory of complexity tends to obscure this fact. It is one thing to be required to generate a large set. It is quite another to dawdle in generating some of the elements of that set. And it is the degree of unnecessary dawdling that is relevant to assessments of elegance and efficiency. Therefore, it would be desirable to have an account of generator complexity that does not penalize a generator for the sheer size of the set to be generated.

An obvious response to these concerns is to "factor out" the expense of writing down the answer from our assessment of generational complexity. That is, we can let the *generational complexity* of a finite set generator be its standard complexity divided by the size of the output set as a function of the input size.

---

[97] Just count the string to check its length, and then check to ensure that each symbol occurring in the string is an element of Σ.

> Generation problem $\{P_x\}$ is in *generational complexity class p(x)* just in case there is a program P that solves $\{P_x\}$ such that for each natural number n, the greatest resource consumption of P on an input of size n is less than or equal to $p(n)|P(n)|$,

where $|P(n)|$ is the size of P(n).[98] Notice that the size of the set generated, rather than the size of the machine's output, serves as the denominator in the quotient. Hence, a machine is penalized for listing its output set in a highly redundant manner.

An obvious candidate for output size is the cardinality of the set generated. But the matter is flexible, and in some applications (e.g. when the *elements* of the set to be generated are intractably large with respect to the input size) it might be of interest to measure the size of the output set as the sum of the sizes of its elements. The size of an element may be any natural, syntactic measure such as string length. So for example, the output string

<112 212 345 445 213 123 123 123 123>

would have size 5 on the cardinality measure but size 15 on the summed string length measure.

Consider two generators that conjecture at stage i some subset of Hyps(i) that bears relation S to the given evidence e_i at stage i. Assume also that the first device generates an elegant, independent axiomatization at each stage, while the latter operates like HEMP, NICOD, and CONSIST (i.e. by "sifting" Hyps(i) by means of some (expensive) test procedure for S). *Ceteris Paribus*, a procedure P that generates an elegant, independent set seems more efficient than a procedure P', whose conjecture is highly redundant. But assume that the size of a generated set of sentences is just the sum of the lengths of the sentences in the set. Then the inelegance of the "sloppy" procedure P' does not show up in its generational complexity. This anomaly results from the division of the resource consumption of P' by by the size of the gratuitously large set it generates. Such a measure confuses bureaucratic "red tape"[99] with productivity.

But the difficulty is easily corrected. Let the *logical* size of a set Γ of sentences be the size (in the previous sense) of the smallest subset K of Γ that is logically

---

[98] There is an alternative approach in the literature that takes the complexity of a generation problem to be the complexity of the problem of deciding the graph of the generation function. That is, given a pair <n, S>, where n is a number and S is a finite set, the machine must decide whether $S=P_n$. I have no quarrel with this approach, but since mine applies directly to generation procedures without converting them into decision procedures, I find it more convenient. I thank Ken Manders for pointing out the alternative theory to me.

[99] This has nothing to do with Turing tape.

equivalent to Γ.[100] Assuming this size measure, devices with heavily redundant conjectures are indeed penalized by generational complexity. Now, a device that tests too many hypotheses cannot "hide" behind the bloated size of the redundant set it conjectures.

The importance of this proposed approach to generational complexity theory is to provide a formal explication for pre-theoretic intuitions whether a given generator is elegant. Since almost every inductive inference system in A.I. involves some sort of generate-and-test architecture,[101] there is indeed a healthy source for such intuitions.[102] By way of illustration, consider the following two examples. The first example illustrates how an intuitively elegant generator has a low generational complexity, and how an intuitively inelegant generator has a high generational complexity. First, consider the easy problem.

$$P_x = \Sigma^x$$

$$\Sigma = \{1,0\}.$$

$|P_x|$ grows as 2^x and x grows at least as fast as 2^n, where n is numeral length. So $|P_x|$ grows at least as fast as the elementary function

$$2^{2^n}$$

Nevertheless, it seems as though each set P_x can be generated without thinking very hard about how to build any individual element of the set. This intuition is indeed confirmed by the proposed complexity measure. For consider the following, recursive description of the function GEN(x) = P_x:

GEN(n) = $\{\Lambda\}$ if n=0
= $U\{\sigma * k: \sigma \in GEN(n-1)$ and k=0 or k=1$\}$ otherwise.

The set brackets and union signs are suspicious, but they will only be employed in unproblematic situations in which the sets to be joined are disjoint, so that union amounts to list concatenation. The above function can be thought of as fed to an "interpreter" that computes it as though it specified the following, more explicit procedure in an ALGOL-like language:

---

[100] This measure may or may not be effective, depending on the expressive power of the hypothesis language.

[101] Two notable exceptions are Langley and Simon's BACON, and the procedures for inferring regular sets that have been presented by Chomsky and Feldman.

[102] Recall, for example, the discussions of Horning and Pao regarding efficient hypothesis generation.

```
Procedure GEN(n):
begin
        set out=<>;
        if n=0 then output Λ
        else
        begin
                set Γ=GEN(n-1);
                for each x in Γ do
                begin
                        set out=out U {1*x};
                        set out=out U {0*x}
                end
        output out
        end
end.
```

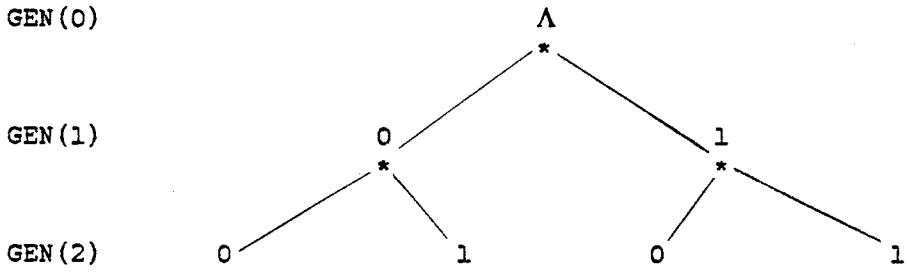
For example, consider the case of computing this function for the case of n=2.

GEN(0) = {()}

GEN(1) = {()*0, ()*1}
= {(0),(1)}

GEN(2) = {(0)*0, (0)*1, (1)*0, (1)*1}
= {(0,0), (0,1), (1,0), (1,1)}

The concatenations performed in the computation of GEN can be thought of as forming the following tree:



where each path in the tree is an element of the generated set.

Let the cost of computing GEN on input *number* x be defined as the number of times the variable 'out' is reset on input x. In light of the tree, it is obvious that

$c(x) = 0+(2X1)+(2X2)+(2X4)+...+(2X2^x)$

$=2[\Sigma_{i=0 \text{ to } x}(2^i)]$.

But the output cardinality is 2^x. Dividing the resource consumption by this quantity yields

$2[\Sigma_{i=0 \text{ to } x}(1/2^i)]$.

Switching from input quantities to the lengths of input numerals requires that the quantity 2^n be substituted uniformly for x.

$$2[\Sigma_{i=0 \text{ to } 2^n}(1/2^i)]$$

The sequence of values of this function is

$$2(1+3/4),\ 2(1+15/16),\ldots,\ 2(1+[(2^{2^n}-1)/2^{2^n}])$$

which is bounded above everywhere by the constant function f(x)=4, so the generational complexity of procedure Gen is *constant* even though the cardinalities of the generated sets expand super-exponentially in input length. The new measure captures the intuition that although the set to be generated is ungainly in size, it can be generated completely with very little effort expended on search or in constructing a description for any particular element.

Not every generation problem can be shown to have a small generational complexity so easily. For example, consider the generation problem $\{P_x\}$, such that P_n is the set of all first order formulas in the language of addition that are true in the standard model. It is known that deciding truth over the unrestricted language of addition requires at least super-exponential time [Fischer74].

The obvious generation procedure is to enumerate the set of all wffs of length k in the alphabet of addition, and to apply the super-exponential test to each wff. The total complexity of this procedure will be the test complexity times the number of wffs over the cardinality of the generated set. The denominator is less than the number of wffs of length k, so the generational complexity of this procedure is at least as great as the test complexity. So the high cost of the test drops straight through the calculation.

If the number of truths of length n of the language of addition grows slowly in n, the generational complexity of the problem cannot be less than super-exponential. For if it were less, the problem of deciding whether an arbitrary first-order formula of addition is true could be carried out in less than super- exponential time using such a procedure. Given an arbitrary sentence, one could calculate its length quickly, pass this length to the generator, and then check whether the given sentence is in the (small) generated set.

We have seen a problem whose generational complexity is low, and one whose generational complexity is not obviously low. How does generational complexity compare to the complexity of the corresponding test problem on this account? Our previous proof that a fast test does not yield a fast generator now falls through, for it depended on the fact that on the standard theory, it is always hard to

generate a big set. But it is still not obvious how a fast test guarantees a fast generator on the new account of generational complexity. But now the converse is not obvious either. Any test procedure that works by generating P_k and performing a membership test will have to inspect all of P_k in the worst case. And if P_k is not polynomial in the size of the given object, this test will also fail to be polynomial in input size in the worst case.

Although this discussion is inconclusive, it does suggest that the quotient measure pushes aside some glib trivialities in the relationship between the difficulty of test and the difficulty of generation. For example, a finite set generator must "discover" in finite time that its output set is *complete* (i.e. not missing any elements). As we have seen, a solution to the corresponding test problem need not embody such an ability. So perhaps there is a generation problem in which an intractable search of some large subset of U is necessary to ensure that no elements of the target set P_k have been skipped. Such a generation problem might be intractable and yet correspond to a tractable decision problem. So it may be that there are special problems in generator design that do not arise in the analysis of the corresponding decision problem. But it would be far better to convert these speculations into a mathematical determination of the question. Unfortunately, it is not settled in this thesis.

Chapter 6
Equivalent Hypotheses and Inductive Efficiency

In the previous chapter, several theories of AE-convergent complexity were reviewed. None of them was found to be intuitively adequate. But even without such a theory, it was observed that the inductive efficiency of different procedures employing the same *strategy* can be compared. So it makes sense to speak of "improvements" in our crude (but very general) strategies HEMP, CONSIST and NICOD, as long as the alleged "improvement" pursues the same strategy.

Recall that HEMP, CONSIST, and NICOD are defined in terms of an arbitrary, effective enumeration of nested, finite subsets of the hypothesis language {Hyps(n)}. At each stage n, each of these procedures conjectures the set of all sentences in Hyps(n) that are suitable with respect to the evidence read so far. Therefore, the strategy pursued by each of these procedures depends upon which enumeration is assumed. Since our investigation of inductive efficiency will be strategy-relative, the choice of an enumeration merits some special attention. This choice will be made in the next section.

With the inductive strategies of HEMP, NICOD, and CONSIST fixed, the investigation of improving their efficiencies can be addressed. The main purpose of this chapter is to investigate the extent to which performance can be improved (without compromising inductive generality) by ignoring equivalent hypotheses *a priori*. The more difficult question of ignoring further hypotheses in light of the evidence given is undertaken in the next chapter.

It should be pointed out, however, that the techniques of this chapter may be of help to any inductive procedure that infers universal hypotheses. Such procedures include Shapiro's model inference system and the Meta-DENDRAL program that seeks universal conditionals linking mass spectra and molecule bond breakages. In either case, the repeated test of equivalent hypotheses is costly dead-weight that ought to be cut out of the system if possible.

6.1. The Hypothesis Enumeration

The two properties of $\{Hyps_n\}$ upon which the methods HEMP, NICOD, and CONSIST rely are:

P1: $Hyps_k$ is a finite subset of $Hyps_{k+1}$ and

P2: for every h in H there is a k such that h is in $Hyps_k$.

The second property is unnecessarily strong. In fact, each of the enumeration procedures proposed in chapter four retains its inductive scope even if the second property is replaced by

P2': for every h in H there is a k such that some finite subset G of $Hyps_k$ is logically equivalent to h.

Since any sentence in a purely universal hypothesis language H is logically equivalent to a finite set of clauses, P2' is satisfied even if $U\{Hyps_x\}$ is just the set CL(H) of clauses on the vocabulary of H. So the discussion can be simplified by letting H be the set of clauses on a function-free vocabulary. Admittedly, this maneuver to eliminate many truth-functional variants of the same proposition from consideration has its dark side. Some sentences in which n disjoined conjunctions of length m occur are expressible in clausal form only by a set of clauses whose cardinality is m^n. But clauses are structures familiar to computer scientists, and we can draw upon and add to this familiarity without any loss in the inductive scope of our methods by choosing them as our hypothesis language.

We have now decided that U{Hyps(n)}=CL(H). But it remains to define each finite set Hyps(n) in a manner consistent with this requirement. This choice will not affect inductive scope in any way, but it will affect the respective inductive strategies pursued by HEMP, NICOD and CONSIST dramatically.

There is some merit in considering syntactically simple, logically strong clauses first, saving long, weak, detailed ones for later. Then if we are lucky, we receive a jackpot of strong, simple truths early in our inquiry, saving the weak, complicated, special cases for later. But in the end, this choice of an hypothesis enumeration is one among many, and I make it only to proceed to more interesting computational issues that demand that some choice be made.

Since adding disjuncts to a given clause can only weaken it, logical strength seems happily wedded to syntactic shortness for clauses. But the marriage is not quite so

simple, for longer clauses can entail shorter ones, as the following, elementary example shows.

$$(.)[Px_1x_2 \vee Px_2x_3 \vee Px_3x_1]$$

$$(.)[Px_1x_1] \ .$$

So there is no sequence of clauses non-decreasing in length that is non-increasing in logical strength. We might still hope to be able to order such a simple class of sentences by logical strength, even if some long sentences precede shorter ones. Unfortunately, this is also impossible, for consider the following schema:

$$C(n)='(.)[Px_1x_2 \vee Px_2x_3 \vee \ldots \vee Px_nx_1]'$$

$$C='(.)[Px_1x_1]'$$

Clearly, for any n, C[n] properly entails C. So infinitely many distinct clauses properly entail C. Hence, no enumeration {Hyps(n)} of finite, nested subsets of CL(H) can be defined such if that if n>n' then no element of Hyps(n) entails any element of Hyps(n').[103] One might attempt to remedy this difficulty by requiring only one representative of each logical equivalence class of clauses to be in U{Hyps(i)}. But notice that none of the C(i) are logically equivalent to one another. So the desired non-increasing class {Hyps(n)} does not exist in this case either.

Since logic does not cooperate with the aim of finding a clausal enumeration that is non-increasing with respect to entailment, it makes sense to focus on syntactic complexity instead. An obvious (but certainly not the only) measure of syntactic complexity of a clause is the number of literals (atoms or negated atoms) occurring in it. Call this quantity the *length* of the clause.

The obvious step would be to define Hyps(n) as the set of all clauses in H of length n or less. But unfortunately, this set is infinite, because H has infinitely many variables, and any substitution of variables for variables in a clause of length n yields a clause of length n. But variable-renaming variants are logically equivalent. So the respective strategies of HEMP, NICOD and CONSIST are not altered if all but finitely many variable-renaming variants are eliminated from each set Hyps(n). One computationally trivial way to do so is as follows: Let a be the arity of the predicate of greatest arity in the vocabulary of H. Let Var(n) be the set $\{x_1,\ldots,x_{cn}\}$ of the first cn variables in the vocabulary of H. Then

---

[103] For any n, $\Gamma(n) = U\{Hyps(i): 0 \leq 0 \leq n\}$ is finite. Also, there is an m such that $C \in Hyps(m)$. But $\{C(n): n \in \omega\}$ is infinite, and hence cannot be a subset of $\Gamma(m)$.

Hyps(n):= {c: cϵCL(H), |c|$\leq$n and if variable x occurs in c then xϵVar(n)}.

Now {Hyps(i)} satisfies P1 and P'2, but each Hyps(n) is finite. This is the enumeration that will be assumed in the balance of the thesis.

6.2. Equivalent Clauses, Complexity, and Efficient Generation

The clauses in Hyps(n) can still be logically equivalent to one another. In this section, a simple syntactic condition for clause-to-clause entailment is given. It follows from this condition that there is no known method that could easily decide such entailments in general (i.e. that the limited entailment problem in question is NP-hard). But there are enough easy, special cases to significantly improve the performance of the naive enumeration techniques of chapter four.

6.2.1. Clause-to-Clause Entailment

Let C,C' be function-free clauses.

Definition: C *collapses into* C' if and only if there is a substitution θ of variables for variables such that each literal L occurring in Cθ occurs in C'.

Fact C1: |=C if and only if for some atom A, both A and -A occur in C.[104]

Fact C2: If C collapses into C' then C |= C'.[105]

Fact C3: If no predicate P and its negation both occur in C', then C|=C' if and only if C collapses into C'.

Proof; Fact C3: The side <=== of C3 is a trivial consequence of C2. So consider the side ===> of C3. Let C,C' be two clauses without function symbols such that not |=C'. Also assume that there is no θ such that each literal occurring in Cθ occurs in C'. We must show that C does not entail C'. Let dq(C) be the result of removing all the occurrences of quantifiers from C. If some predicate occurs in C but not in C' then there is some structure M for the language of C' and an interpretation i such that M,i do not satisfy dq(C'). And since some P

---

[104] Let C be a clause such that no atom and its negation both occur in C. Let V be the set of variables occurring in C. For each n-ary predicate Q occurring in C, construct the relation $R_Q = V^n - \{\sigma \epsilon V^n : Q\sigma \text{ occurs in } C\}$. By assumption, for no Q, σ is it the case that $Q\sigma$ and $\neg Q\sigma$ both occur in C. Hence $\langle V, R_{Q1}, \ldots, R_{Qn} \rangle$, id do not satisfy the open matrix of C, where id:AE--->V is the identity function. So C is not a tautology.

[105] This fact is trivial.

occurs in C but not in C', we can extend M with the universal relation to interpret P if P occurs in C non-negated, or the empty relation if P occurs in C negated. Therefore, C does not entail C', and we are done. So we need only consider the case in which each predicate that occurs in C occurs in C' as well. Then by the fact hypothesis, either (I) some predicate Q occurring in C occurs with the opposite sign in C' and only with the opposite sign in C', or (II) every predicate occurring in C occurs with the same sign in C' and only with the same sign in C'. Assume case (I). Let VAR(C') be the set of all variables that occur in C'. Choose some Q that occurs in C and (only) with the opposite sign in C'. If Q occurs non-negated in C, let R(Q) be $VAR(C')^n$. If Q occurs negated in C, let R(Q) be the empty set. For each predicate P of arity n occurring in C' that is not identical to Q, let the relation $R(P)=VAR(C')^n-\{\sigma$: $P\sigma$ occurs in C' non-negated$\}$. Let <M,f> be the structure whose domain is VAR(C') and such that f(P)=R(P), for each predicate in the language of C'. Let i be the identity interpretation function i:Var(C')--->Var(C') that takes each variable in VAR(C') to itself. There are four possibilities for each atom A occurring in C'. (a) Q occurs in A and A is negated; (b) Q occurs in A and A is non-negated (c) some predicate R distinct from Q occurs in A and A is negated; and finally (d) some predicate R distinct from Q occurs in A and A is non-negated. (a) Suppose A is $-Q\sigma$. Then by the case assumption, Q occurs non-negated in C. Hence, $f(Q) = Var(C')^n$, so $-Q\sigma$ is not satisfied in M under any interpretation. (b) Suppose A = $Q\sigma$. Then by dual argument, $Q\sigma$ is not satisfied in M under any interpretation. (c) Suppose A is $-P_s$, where P is not identical with Q. Since C' is not a tautology by the fact hypothesis and fact C1, the atom P_s does not occur in C'. Hence, the variable sequence σ was never deleted from f(P)=R(P). So M,i do not satisfy $-P_s$. (d) Suppose A is P_s, where P is not identical to Q. By construction M,i do not satisfy P_s, for σ is deleted from the relation f(P)=R(P). Since no atom A occurring in C' is satisfied by M,i, M,i do not satisfy C'. But M,i do satisfy C, for some atom occurring in C in which Q occurs is satisfied in M under any interpretation, by the construction of f(Q). Hence, C does not entail C'.

Case (II): Suppose that every predicate occurring in C also occurs in C' with the same sign. For each n-ary predicate P occurring in C' form $R(P)=Var^n-\{\sigma$: $P\sigma$ occurs non-negated in C'$\}$ if P occurs non-negated in C, and form $R(P)=\{\sigma$:$P\sigma$ occurs negated in C'$\}$ otherwise. This is possible because no predicate occurs both negated and non-negated in C' by hypothesis. Finally, let <M,f> be the structure with domain Var(C') such that f(P) is R(P) for each P occurring in C'. Let i be the identity interpretation. Let $P\sigma$ be a non-negated atom occurring in C. Then by construction, σ is not in R(P), so M,i do not satisfy $P\sigma$. Let $-P\sigma$ be a negated atom occurring in C. Then σ is in R(P), so M,i do not satisfy $-P\sigma$. Hence, M does not satisfy C'. Now assume that there is a j such that M,j do not satisfy dq(C). Hence, for each negated atom $-P\sigma$, $j(\sigma)$ is an element of R(P), and for each non-negated atom $P\sigma$, $j(\sigma)$ is not an element of R(P). Consider an arbitrary $-P\sigma$ occurring negated in C. If $j(\sigma)$ is in R(P), then by construction, there is an atom $-P\tau$ occurring in C' such that $j\sigma=\tau$. Next consider an arbitrary $P\sigma$ occurring non-negated in C. If $j(\sigma)$ is not in R(P), then there is an atom $P\tau$ occurring in C' such that $j(\sigma)=\tau$. Hence, j maps every atom of C to some atom occurring in C', contrary to hypothesis. So C is satisfied in M on every interpretation, and hence does not entail C'. Q.E.D.

Notice that the antecedent of fact C3 is not vacuous, for consider the following two clauses:

1. (.)[Pxy v -Pyz]

2. (.)[Pyy v -Pzz].

Now examine the question whether clause (1) entails clause (2). In this case, the antecedent of fact C3 is violated and the consequent of C3 is false. For notice that clause (1) does entail clause (2).[106] So although the collapsing condition is an elegant characterization of entailment over a special class of clauses, relying on it exclusively will lead us to miss some redundancies in an expanded clausal language.

The very simple structure of function-free clauses might suggest that clausal entailment is easy to decide. But even when the clauses involved are assumed to be *homogeneous* (i.e. all atoms occurring in the two premise and conclusion have the same predicate and sign) the clausal entailment decision problem it is NP-complete.[107]

Theorem NP1:

If C,C' are arbitrary, homogeneous clauses in a predicate Q (whose arity is greater than 1) then to decide whether C entails C' is an NP-complete problem.

Corollary:

The problem to decide for any two clauses C,C' whether C|=C', is NP-hard.

Proof of theorem:[108]

Recall that C|=C' if and only if there is a θ such that each literal occurring in Cθ also occurs in C'. So the problem is in NP, for a nondeterministic machine can guess an appropriate θ and verify that Cθ's literals are all in C' in polynomial time.

The problem of deciding whether an arbitrary, finite, non-directed graph is 3-colorable is NP-complete [Stockmeyer73]. A graph G is a pair <V,R> where R is a set of unordered pairs drawn from V. G is 3-colorable if and only if there is an f:V--->{1,2,3} such that for any x,y in V, f(x)=f(y) only if {x,y}ϵR. Let G,G' be two graphs <V,R>, <V',R'>, respectively. Say that G *collapses into* G' if and only if there is a

---

[106] I am indebted to Ken Manders for this example.

[107] The size of a problem instance is taken to be the sum of the lengths of the clauses involved. Notice that in the strategies pursued by NICOD, HEMP, and CONSIST, this quantity is linearly related to the input sequence size. So the NP-completeness of this decision problem may show up significantly in the complexity of this strategy.

[108] The outline of this reduction is suggested in [Levin73] and was brought to my attention by Richard Statman.

function f from V to V' such that for each x,y ϵ V, if $\{x,y\}\epsilon R$ then $\{f(x),f(y)\}\epsilon R'$. Finally, let K3 be the graph $\langle V'',R''\rangle$, where $V''=\{1,2,3\}$ and $R''=\{\{1,2\},\{1,3\},\{2,3\}\}$.

Lemma: G is 3-colorable if and only if G collapses into K3.

===> Suppose $G = \langle V,R\rangle$ is 3-colorable. Then there is an f:V--->V'' such that for each $x,y\epsilon V$, if $\{x,y\}\epsilon V$ then f(x) is not identical to f(y). For reductio, suppose that there are $x,y\epsilon V$ such that $\{x,y\}\epsilon R$ but $\{f(x),f(y)\}$ is not an element of R''. Since the range of f is a subset of V'', and for each a,b in V'' such that a is not identical to b, $\{a,b\}\epsilon R''$, it follows that f(x)=f(y). So f is not a 3-coloring, which is absurd. Therefore, for every $x,y\epsilon V$, if $\{x,y\}\epsilon R$ then $\{f(x),f(y)\}\epsilon R''$. So G collapses into K3.

<=== Suppose G collapses into K3. Then there is an f such that for all $x,y\epsilon V$, if $\{x,y\}\epsilon R$ then $\{f(x),f(y)\}\epsilon R''$. For reductio, assume that $\{x,y\}\epsilon R$ but f(x) = f(y). But since no pair {x,x} is in R'', for any $x\epsilon V''$, it follows that {f(x),f(y)} is not in R'', which is absurd. Therefore f is a 3-coloring, which proves the lemma.

Now it suffices to show that for any pair of graphs $\langle G,K3\rangle$, we can produce in polynomial time a pair of clauses $\langle C_G,C_{K3}\rangle$, each of which is homogeneous in Q, such that $C_G \models C_{K3}$ if and only if G collapses into K3. Let $G=\langle V,R\rangle$. Enumerate V. For each edge {x,y} of G, put the literal Qxy into C_G if x<y in the enumeration, and put Qyx into C_G otherwise. Clearly, this can be done in polynomial time. Carry out the same encoding for K3, and call the result C_{K3}. Notice that the results of this translation satisfy the conditions of fact C3 above. So by this fact, $C_G \models C_{K3}$ if and only if C_G collapses into C_{K3}. But it is evident from the construction that C_G collapses into C_{K3} if and only if G collapses into K3. So by the lemma, $C_G \models C_{K3}$ if and only if G is 3-colorable. Q.E.D.

This result is indicative of the mathematical power of computational complexity theory. To put the matter intuitively, no known algorithmic approach can decide whether one clause entails another without becoming enmeshed in an intractable search.

Notice, this does *not* imply that for every given pair of clauses, deciding whether one entails the other must involve a hopeless search. It shows only that for infinitely many distinct problem sizes, for each *known* approach, some problem instance of this size will involve the method in a hopeless (exponential) search. Therefore, there may be interesting, easily decidable special cases of clausal equivalence. Moreover, I have not shown that the problem of deciding clausal equivalence is itself intractable. It is hard to see what would make this problem easier than the entailment problem, but strictly speaking, our argument that entailment is difficult to decide does not show that deciding equivalence is.

The following sections examine some special cases of clausal equivalence, and propose some efficient techniques for ignoring redundant hypotheses of several types. But first, we must reap a further consequence of the present theorem. It puts teeth in our claim that testing clausal hypotheses is difficult and should be avoided if possible. That is,

Theorem NP2:

The problem of deciding whether a given homogeneous clause c is inconsistent with a homogeneous conjunction of three negated atoms in the same predicate that occurs in c is NP–complete.[109]

Corollary 1:

The problem of deciding whether a given clause is inconsistent with a finite set of basic statements (e.g. "evidence") is NP–hard.

Corollary 2:

Deciding whether a given homogeneous clause is not satisfied in a given, finite structure is NP–complete.[110]

So we know that if P is not equal to NP, then there is no known algorithm that can decide the suitability relation of CONSIST without being caught in an intractable search in the worst case. And corollary 2 shows that HEMP's satisfaction strategy is no piece of cake either. If P is distinct from NP, then theorem NP2 and its corollaries show that there is something intrinsically difficult about searching through possible counterexamples to a universal hypothesis, and both HEMP and CONSIST face this task.

Recall that on any given evidence, HEMP must find a maximal subset of this evidence that is the diagram of some relational structure for the hypothesis language. This cutting down on the input evidence may offset the difficulty of performing the required satisfaction test. But unfortunately, simply chopping down the evidence in an appropriate way is already an intractable task, as we see in the following theorem.

---

[109] The homogeneous clausal entailment problem with a 3-clause conclusion is NP-complete. Consider an arbitrary instance $\langle c,c' \rangle$ of this problem. Negating the conclusion c' yields the conjunction c'' of the negations of the atoms in c'. This conjunction is inconsistent with the premise if and only if $c \models c'$. Clearly, driving the negation into the conclusion can be done in polynomial time. So the problem is NP-hard. And the inconsistency problem is in NP because, by the fact just mentioned and our limited completeness theorem, the problem can be decided quickly by a device that guesses a substitution θ and then checks whether θ collapses c onto the negation of c', which can be done in polynomial time.

[110] Think of the structure as encoded by its diagram. All we need do is assume that the structure is K3. So the result follows directly from the construction of theorem NP2.

Let L be a first-order language with no function symbols. The Maximal Diagram Search (MDS) problem is the problem of deciding whether a finite subset S of atoms of L contains a diagram of a structure of cardinality no less than k.

Theorem NP3:

The MDS problem is NP-complete.[111]

This result underscores the point that hypothesis tests should be avoided whenever possible, whether the strategy is that of HEMP or of CONSIST. In light of these results, we return to the task of effectively eliminating the repeated test of many variants of the same hypothesis.

6.2.2. Variable-Renaming Redundancies

Clause c is a *variable renaming variant* of clause c' if and only if there exists some substitution θ of variables for variables such that $c=c'\theta$. Any two clauses that are variable renaming variants of one another are logically equivalent. One way to avoid considering renaming variants of the same clause is to choose a unique clause from among each class of renaming variants as *canonical*. A convenient selection is:

> Clause c is canonical if and only if the first variable to occur in c is x_1, and for any variable x_n that occurs in c, variable x_{n-1} occurs before x_n in C.

So for example,

$$(\exists)[(P(x_1,x_2) \vee \neg Q(x_1,x_3)]$$

is canonical but its renaming variant

---

[111] First of all, it is easy to see that the problem is in NP, for all one need do is to guess a subset C of the constants occurring in S and then to check in polynomial time whether every predicate occurring in S is either asserted or denied of every sequence of constants of appropriate arity.

The CLIQUE problem is known to be NP-complete [Garey79] p.54. The problem is to decide for an arbitrary graph G and positive integer k whether some subgraph of G is the complete graph on k vertices. So let ⟨G,k⟩ be an arbitrary instance of CLIQUE. We construct a corresponding instance ⟨E,k⟩ of the Maximal Diagram Search problem as follows. The hypothesis language L will have but one binary predicate P and a distinct constant for each vertex in G. Add the atoms Pab, Pba to E if either a is adjacent to b in (undirected) graph G, or a=b. This construction can clearly be accomplished in polynomial time, for all we need do is to run through the edges of G for each pair of vertices in G. Since there is no negative evidence in E, the only diagram that E might possibly contain is one for which the denotation of P is a universal relation on some subset of the constants occurring in E. Notice that self loops will not interfere, for every self-loop is added. So there is a diagram of a structure of cardinality k in the evidence if and only if there is a complete graph of this cardinality in the given graph. Hence, the Maximal Diagram Search problem is NP-hard.

$$(.)[(P(x_3,x_1) \vee \neg Q(x_3,x_2)]$$

is not.

Even if we require canonical variable patterns in clauses, commuting predicates can lead to logically equivalent clauses with canonical variable patterns, as in the following example:

$$(.)[P(x_1,x_2) \vee Q(x_1,x_2)]$$

$$(.)[Q(x_1,x_2) \vee P(x_1,x_2)].$$

Such redundancies can be eliminated readily by enforcing a canonical order on the first occurrences of the occurring predicates. An analogous consideration motivates the requirement that each occurrence of a negated predicate P be preceded by all non-negated occurrences of P. Henceforth, a canonical clause is taken to have a canonical predicate pattern as well as a canonical variable pattern.

The following notation will be useful in the balance of the thesis. Consider clause c:

$$(.)[P(x_1,x_2), Q(x_2)]$$

This clause is comprised of what we may call a *clausal blank*

$$B=\langle P(*,*), Q(*)\rangle$$

along with a sequence of variable subscripts $\sigma=\langle 1,2,2\rangle$, which we may call a *variable pattern*. Evidently, clause c is specified uniquely by B and σ. It is then natural to speak of c as the *specification* of B corresponding to σ.

Variable patterns are arbitrary sequences of natural numbers. When the specification of a blank corresponding to variable pattern σ is canonical, then we can say that σ is a canonical sequence and B is a canonical blank. Hence, σ is a *canonical sequence* if and only if

1. $\sigma_1 = 1$ and
2. for all $m>1$, the first occurrence of m in σ is preceded by an occurrence of $m-1$ in σ.

Also, B is a *canonical blank* if and only if for any i, $j>i$, P_i occurs before P_j in B. Call the set of all canonical sequences of length n Canon(n). Each element of Canon(n) corresponds to a unique canonical specification of a canonical blank with n occurrences of '*'.

Consider the procedure CONSIST. If CONSIST were to make use of the enumeration {Hyps'(i)} such that Hyps'(n) is the set of canonical clauses in Hyps(n), its strategy would be the same as it is when all of {Hyps(i)} is examined. The same observation holds of HEMP, NICOD, and many obvious variants of these procedures. Moreover, for each canonical clause C of length n in which k distinct variables occur, there are on the order of $(cn)_k$ variants of C in Hyps(n), where c is the arity of the predicate of greatest arity in the vocabulary of H[112], and for any i,j, $(i)_j$ is the *falling factorial* of i with respect to j.[113] By Stirling's approximation, |Hyps(n)| is greater than |Hyps'(n)| by an exponential factor.

Given that there are exponentially fewer canonical clauses of size n than there are elements of Hyps(n), it would clearly be useful to find an elegant way to consider only Hyps'(n) instead of all of Hyps(n) at each stage n. One way to avoid considering these variants is to have a generation procedure for Hyps(n) that has only polynomial generational complexity, and to employ this procedure to enumerate the elements of Hyps(n) that must be tested against the evidence. The next subsection presents such a procedure.

6.2.3. Generating Canonical Clauses

It is easily decidable whether a given clause has a canonical variable pattern.[114] It is even easier to ensure that the occurring predicates occur in canonical order. But there remains the problem of efficiently *generating* just the canonical subset of Hyps(n), for any n. This problem is a finite-set generation problem, of the sort defined previously.

The question would not be of much interest if Hyps(n) did not grow much more quickly than its canonical subset. But in fact, it grows more quickly by at least a falling factorial factor, as we have seen in the previous section. Therefore, the naive procedure of generating Hyps(n) and testing each element to see if it is canonical would have exponential *generational* complexity, even though the corresponding decision procedure is linear in the length of the given clause. So the practical interest of the generation problem is not obviated by the possession of a fast test for canonical variable patterns.

---

[112] c.f. the definition of Hyps(n) in the previous section.

[113] $(i)_j = i!/(i-j)!$.

[114] Set k=1, and look at the first occurring variable. If its subscript is not 1, say 'No'. Otherwise go to the next variable. At the nth variable, if the subscript is greater than k+1, say 'No'. If the subscript is equal to k+1, set k to k+1 and go to the next variable. Otherwise, just go to the next variable. If there are no more variables, say 'Yes'. This procedure uses resources linear in the length of the given clause and returns 'Yes' if and only if the given clause has a canonical variable pattern.

Fortunately, the following, simple recursive function is a computationally elegant generation procedure for the canonical sequences specifying any given clausal blank, and hence for the entire canonical subset of Hyps(n).[115]

$$CANON(0)=\{\Lambda\}$$

$$CANON(n+1) = \{\sigma * k:\ \sigma \in CANON(n) \text{ and } 1 \leq k \leq |\sigma|+1\}$$

where $|\sigma|$ denotes the number of distinct numbers occurring in σ and Λ denotes the empty sequence.

For example, CANON(4) yields the following computation.

$$CANON(0) = \{\Lambda\}$$

$$CANON(1) = \{\Lambda * 1\}$$

$$= \{(1)\}$$

$$CANON(2) = \{(1)*1,\ (1)*2\}$$

$$= \{(1,1),\ (1,2)\}$$

$$CANON(3) = \{(1,1)*1,\ (1,1)*2,\ (1,2)*1,\ (1,2)*2,\ (1,2)*3\}$$

$$= \{(1,1,1),\ (1,1,2),\ (1,2,1),\ (1,2,2),\ (1,2,3)\}$$

$$CANON(4) = \{(1,1,1)*1,\ (1,1,1)*2,\ (1,1,2)*1,\ (1,1,2)*2,\ (1,1,2)*3,\ (1,2,1)*1,\ (1,2,1)*2,\ (1,2,1)*3,\ (1,2,2)*1,\ (1,2,2)*2,\ (1,3,2)*3,\ (1,2,3)*1,\ (1,2,3)*2,\ (1,2,3)*3,\ (1,2,3)*4\}$$

$$= \{(1,1,1,1),\ (1,1,1,2),\ (1,1,2,1),\ (1,1,2,2),\ (1,1,2,3),\ (1,2,1,1),\ (1,2,1,2),\ (1,2,1,3),\ (1,2,2,1),\ (1,2,2,2),\ (1,2,2,3),\ (1,2,3,1),\ (1,2,3,2),\ (1,2,3,3),\ (1,2,3,4)\}$$

If we think of each canonical sequence in CANON(k) as a path in a tree whose vertices are labeled with the natural numbers occurring in these sequences, then the function CANON directs exactly one concatenation for each vertex in the tree. For example, the figure depicts the tree whose paths are the elements of CANON(4). Notice that each "level" in the tree corresponds to a recursive call of the function

---

[115] Hyps'(n), the set in question, is just the disjoint union of the canonical specifications of each canonical blank of length n.

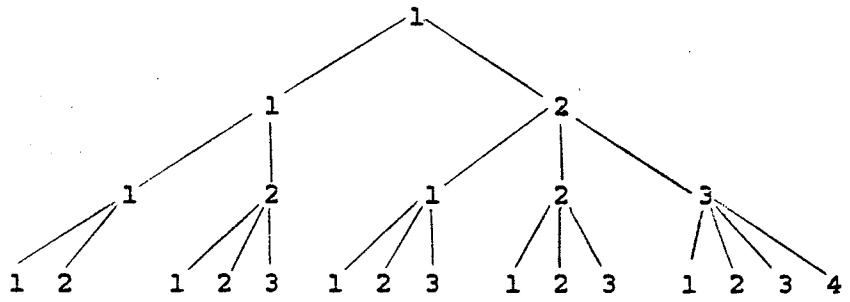


Figure 6-1: A Canonical Tree

CANON.Let c(n) be the effort required to compute CANON(n). Notice that each recursive call of CANON involves a concatenation. Hence c(n) can be estimated fairly as the number of concatenations performed in computing CANON(n). By inspecting the tree, it is evident that c(n) is bounded above by the quantity n|CANON(n)|, for there are n concatenations involved in each canonical string of length n. If the size of the output set is measured as its cardinality, then the generational complexity in n is linear. But remember that the "size" of the input number n is on the order of $\log_2(n)$. Hence, the proper generational complexity of CANON is on the order of 2^n. But if the size of the output set is measured as the sum of the lengths of the strings in the set, size of the set is n|CANON(n)|, so the generational complexity of CANON(n) is constant. This assessment helps capture the intuition that CANON is a very efficient generator of a large set.

6.2.4. Tautological Redundancies

The simple restrictions on clausal form introduced so far suffice to eliminate all equivalences among specifications of a given clausal blank in which no predicate occurs more than once. But when a predicate occurs two or more times in a canonical clause, this clause may be *tautologous* or *non-reduced* (i.e. equivalent to a sub-clause of itself).

By fact C1, a clause c is tautologous if and only if some atom and its negation both occur in c. Tautologies are entirely useless in an inductive system, but once they are considered for empirical test, they are as costly to "handle" as empirically interesting clauses. And there are very many distinct tautologies in Hyps(n). Therefore, tautologies are eliminated from the class of canonical clauses. The test whether a given canonical clause is a tautology is once again easy, but not so easy as the test for canonical variable patterns.[116]

[116] for each pair of occurring literals, check whether one is exactly the negation of the other. For a clause of length n, the complexity of this procedure is merely on the order of n^2.

6.2.5. Non-Reduced Redundancies

All but finitely many clauses have the property that some predicate occurs more than once with the same sign. Such clauses are susceptible to yet another type of redundancy. For example, consider:

$$\models (x)(y)(Pxy \vee Pxz) \leftrightarrow (x)(y)(Pxy)$$

$$\models (x)(y)(z)(Pxy \vee Pxz \vee Pzx) \leftrightarrow (x)(z)(Pxz \vee Pzx)$$

One might expect that these redundancies are all special cases of the following simple fact of propositional logic:

$$\text{If } P \models Q \text{ then } \models (P \vee Q) \leftrightarrow Q$$

But it must be kept in mind that clauses are universally quantified. In the case of clauses, we have instead the following relations:

1. If $\models (.)(A \vee B) \leftrightarrow (.)(A)$ then $(.)(B) \models (.)(A)$ but not conversely.[117]
2. If $A \models B$ then $\models (.)(B \vee A) \leftrightarrow (.)(B)$ but not conversely.[118]

where A,B are arbitrary open formulas and (.)(A) denotes the universal closure of open formula A.

If C has no function symbols, then a corollary of fact C3 is:

Corollary to C3:

> If no predicate occurs both negated and non-negated in C, then C is non-reduced (i.e. equivalent to a sub-clause of itself) if and only if there is a substitution θ of variables for variables such that $c\theta$ is a proper sub-clause of c.

The obvious way to test whether a clause is reduced is to look at its subclauses and to see whether any subclause is entailed by the original clause. Since such entailment checking is NP-hard by theorem NP1, we might expect that the problem of deciding reducedness is intractable as well. But theorem NP1 does not imply the NP-hardness of deciding that a clause is reduced. From the point of view of the reduction in NP1, there is some reason to expect that the problem of deciding

---

[117] Assume (.)(B) does not entail (.)(A). Then there is an M such that for each i, M,i $\not\models$ B but M does not satisfy (.)(A). So there is an M such that M $\models$ (.)(A v B) but M does not satisfy (.)(A). For the negation of the converse, (.)(Pxy) $\not\models$ (.)(Pyx) but (.)(Pxy v Pyx) is not equivalent to (.)(Pxy).

[118] Assume A $\not\models$ B. So for any M and any interpretation i, if M,i $\not\models$ A then M,i $\not\models$ B. Suppose M' does not satisfy (.)(B) Then there is an i', M',i' do not satisfy B. Hence M',i' do not satisfy A either, and so M',i' do not satisfy A v B. The other case is trivial. For the negation of the converse, (.)(Pxy v Pzw) is equivalent to (.)(Pxy), but Pzw does not entail Pxy as an open formula.

whether a clause is reduced is easier than deciding arbitrary entailments. The difficulty with the clausal entailment problem is that the premise may be arbitrarily bigger than the conclusion, and hence can represent any graph whose three-colorability we would like to decide. But in deciding reducedness, the only entailments we need to decide are those in which the conclusion is always the result of deleting some atom or negated atom from the premise. And not every 3-colorability problem instance corresponds (in the sense of the reduction in the proof of NP1) to a clausal entailment problem instance of this limited sort. On the other hand, there may be some other reduction that works. Obvious approaches to solving the problem do seem to run into intractable searches through possible variable substitutions. I leave the NP status of this important decision problem open in this thesis.

It may be easier to see that the problem of generating or recognizing the reduced form of a given, function-free clause is intractable. In this case, the reduced form sought may be arbitrarily shorter than the non-reduced form, so the obvious strategy for solving the problem involves us in entailment checks in which the premise can be arbitrarily longer than the conclusion. This heuristic observation is vindicated by the following theorem:

Theorem NP4:

Given two homogeneous clauses C,C', the problem of deciding whether C' is a reduced form of C is NP-hard.[119]

Corollary:

For arbitrary, function-free clauses C,C', the problem to decide whether C' is a reduced form of C is NP-hard.

The strategy of the proof of this theorem is to show that hard entailment tests cannot be avoided in deciding whether C' is a reduced form of C. Hence, the proof

---

[119] Let $G=\langle V,R\rangle$ be an arbitrary instance of the graph 3-colorability problem. Now form $G'=(G \cup K3)$, renaming vertices, if necessary, to ensure that G and K3 are disjoint. Next, construct $C_{G'}$ and C_{K3} just as before. Recall from the proof of NP1 that G is 3-colorable just in case G collapses into K3. Notice also that (*) G collapses into K3 if and only if $G' = G \cup K3$ collapses into K3. For suppose G collapses into K3. Then the subgraph of K3 onto which G collapses also collapses into K3. So G' collapses into K3, which itself collapses into K3 (i.e. itself). Now suppose G' collapses into K3. Then any subgraph of G' collapses into K3, including subgraph G. This establishes (*). Now construct $C_{G'}$ and $C_{K3'}$ in polynomial time, just as in the proof of NP1. By construction, C_{K3} is a subclause of $C_{G'}$. And by fact C3, G' collapses into K3 if and only if $C_{G'} \not\models C_{K3'}$. So we have that (**) G is 3-colorable if and only if $C_{G'} \not\models C_{K3'}$.

Since C_{K3} is a sub-clause of $C_{G'}$, C_{K3} is equivalent to $C_{G'}$ if and only if $C_{G'} \not\models C_{K3'}$. Observe also that C_{K3} is reduced. So given (**), we have that G is 3-colorable if and only if C_{K3} is a reduced form of $C_{G'}$. It is clear that the construction can be undertaken in time polynomial in G, according to any standard measure of graph size. Since the graph 3-colorability problem is NP-hard, deciding whether C' is a reduced form of C is must also be NP-hard.

tells us very little about how difficult it is to decide whether C' is reduced. But it does suggest that any test procedure that throws non-reduced clauses into reduced form before testing them is intractable. For all I have said here, it may be possible to elude this intractability by generating only non-reduced clauses in the first place. There are many questions to be investigated here, but in order to move on to other issues, I leave most of them unsettled.

6.2.6. Predicate Permutation Redundancies

Even reduced, canonical clauses can be logically equivalent. For consider:

$$(.)[P(x_1,x_2) \vee P(x_2,x_3) \vee P(x_3,x_4)]$$

$$(.)[P(x_1,x_2) \vee P(x_3,x_1) \vee P(x_2,x_4)]$$

That the two clauses are variable-renaming variants can be seen by permuting the first and second conjuncts of the second clause. The problem is that the canonical order on distinct predicates is of no assistance when all the predicates occurring in a clause are identical and of the same sign. But this case can be handled by lexically ordering the variable patterns of the atoms occurring in a clause and by requiring of clause C that for any atoms P,P' of the same sign and predicate such that P precedes P' in C, the variable pattern of P is lexically prior to the variable pattern of P'. This rule excludes the second clause in the above pair, for 24 < 31 but $P(x_2,x_4)$ occurs after $P(x_3,x_1)$. And permuting the second and third disjuncts results in a clause whose variable pattern violates the first constraint on canonical variable patterns. Notice that this requirement is a relation between variable patterns and clausal blanks rather than a property of variable patterns alone. Any pattern that satisfies it with respect to a given blank B is said to be *suited* to B.

It is easy to decide for a given clause whether its variable pattern is suited to its clausal blank, for the lexical order on variable patterns can be decided for two patterns of the same length in time linear in their lengths.[120] Given this procedure, it is easy to decide whether a clause has its atoms in the correct order.[121]

There is also an efficient generator of the set of all canonical specifications of a given clausal blank that satisfy this additional constraint. Recall that all the non-negated occurrences of a predicate come together in a block, followed by the

---

[120] Given σ, τ, begin at the left and check for each whether $\sigma_n > \tau_n$. Whenever this condition fails, return 'No'. Otherwise, return 'Yes'.

[121] begin at the left, and perform the first procedure for the first two atoms of the same predicate and sign. Then proceed to the second and third atom, and so forth. As soon as a violation of the order is discovered, report 'No'. Otherwise report 'Yes' at the end of the clause.

negated occurrences. All the generator in question needs to know, however, is the size of the block and the arity of the signed predicate occurring in the block. This can be accomplished with a list of ordered pairs <x,y> such that x is the length of the block of signed predicates and y is the arity of the predicate in question. Call this sequence a *segmentation* of the blank of the original clause and let any specification of this blank be called a specification of the segmentation as well.

So for example, the blank

<P**,P**,-P**, Q***, -R*>

corresponds to the following segmentation:

<<2,2>,<1,2>,<1,3>,<1,1>>

CANON(6) has 203 elements. The 88 elements of CANON(6) that are *not* suited to the segmentation <<2,3>> are displayed in the following table:

```
111211 111212 112111 112112 112121
112211 112212 112213 112221 112222
112311 112312 112313 112314 112321
112322 112323 111211 121111 121112
121113 121121 121122 121123 122111
122112 122113 122121 122211 122212
122213 122221 122222 122311 122312
122313 122314 122321 122322 122323
123111 123112 123113 123114 123121
123122 123123 123124 123131 123211
123212 123213 123214 123221 123222
123223 123224 123231 123232 123311
123312 123313 123314 123321 123322
123311 123312 123313 123314 123321
123322 123323 123324 123331 123332
123333 123411 123412 123413 123414
123421 123422 123423 123424 123431
123432 123433 123434
```

In the case of the segmentation <<2,4>>, the list would not even fit on the page. Since each of these sequences corresponds to a redundant clause whose test is irrelevant to inductive scope, the importance of a generator that ignores them should be evident.

Indeed, there is such a generator. Procedure CANON2 takes an arbitrary segmentation as input, and returns just the set of all specifications suited to this segmentation. The method is quite similar to CANON, but instead of adding single symbols to the specifications under construction, these specifications are

constructed by adding sequences that correspond to the variable patterns of literals. First we consider two procedures, ATOM and ATOM2, that generate just the correct atom specifications to add to the segmentation specification under construction by CANON2. Then the procedure CANON2 is defined in terms of the component procedures ATOM and ATOM2.

The function ATOM takes two natural numbers y and r as inputs and returns the set of all sequences β of length y such that if $|\sigma|=r$ then $\sigma*\beta$ is a canonical sequence.

$$\begin{aligned} \text{ATOM}(y,r) &= \{\Lambda\} \text{ if } y = 0 \\ &= \{\sigma*k\colon\ \sigma \in \text{ATOM}(y,\max(k,r)) \text{ and } 1 \leq k \leq r+1\} \\ &\quad \text{otherwise.} \end{aligned}$$

The only trick to this definition is that whenever a number k is concatenated to a sequence under construction that is larger than r, any number no greater than n+1 may be added. Hence, ATOM "updates" r (the parameter that keeps track of the upper bound on the size of the numbers that may be added) by maximizing over r and k at each recurrence. For example, consider the case of ATOM(2,3):

```
ATOM(0,3) = {<>}

ATOM(1,3) = {<>*1, <>*2, <>*3, <>*4}
          = {<1>,  <2>,  <3>,  <4>}

ATOM(2,3) = {<1>*1, <1>*2, <1>*3, <1>*4,
             <2>*1, <2>*2, <2>*3, <2>*4,
             <3>*1, <3>*2, <3>*3, <3>*4,
             <4>*1, <4>*2, <4>*3, <4>*4, <4>*5}
          = {<11>,  <12>,  <13>,  <14>,
             <23>,  <22>,  <23>,  <24>,
             <31>,  <32>,  <33>,  <34>,
             <41>,  <42>,  <43>,  <44>,  <45>}
```

The aspirations of ATOM2 are similar, but there are a few additional complications.

ATOM2(σ),r,τ,b):

1. ={τ}
   if σ=Λ

2. =U{ATOM2(σ,max(k,r),τ*k, 1):n<k$\leq$r+1}
   if b=0 and σ=n*Λ;

3. ={ATOM2(σ,max(n,r),τ*n,0)} U
   U{ATOM2(σ,max(k,r),τ*k, 1):n<k$\leq$r+1}
   if b=0, σ=n*τ and not τ=Λ;

4. =U{ATOM2(σ,max(k,r),τ*k, 1): 1$\leq$k$\leq$r+1}
   otherwise.

ATOM2 takes as arguments an arbitrary n-tuple σ and a natural number r. The parameter τ is always "initialized" to the empty string Λ and the boolean parameter b is always "initialized" to 0. We have no interest in the values of ATOM2 for other values of these parameters.

If k is the length of σ, then the value of ATOM2(σ,r,Λ,0) is just the set of all elements of ATOM(k,r) that are lexically greater than σ. That is, ATOM2 is the set of all n-tuples γ such that γ is lexically greater than σ and for any canonical sequence μ in which r occurs, μ*γ is a canonical sequence. So the extra complication of ATOM2 results from the fact that ATOM2 must in a strong sense ignore those sequences that are lexically less than σ.

Consider first some general, architectural considerations. Parameter τ functions as a place to hold the sequence currently under construction, and b serves as a "flag" to indicate whether it is already certain that the sequence under construction is lexically greater than σ. If b is not yet guaranteed to be lexically greater than σ, then b=0, but as soon as the situation changes, b is changed to 1. The sequence under construction grows as σ is successively shortened, so σ disappears just when τ is completed. The parameter r is always taken to be the previous value of r or the number just added to τ, whichever is greater.

Now consider case 1. If σ is empty, τ is finished, so return it. The curly brackets are required because the ultimate output set is formed recursively as a disjoint union of singletons. In case 2, b = 0 and σ is some unit list <n>. Since b is 0, τ is not yet lexically greater than the original σ. On the other hand, so long as b is 0, the number σ_1 is the *least* number that can be added to τ (c.f. 2,3) so τ is never properly less than σ in lexical order. Hence, τ is so far just like the given σ, so to avoid generating a copy of σ, we are compelled to add only numbers

greater than n. But on the other hand, canonicity demands that no number greater than r+1 be added. So each number greater than n and no greater than r+1 is added to τ. Since each result is now lexically greater than the original σ, b is changed from 0 to 1 (for purposes of clarity and symmetry only, for this does not alter the output in any way). In the third case, b is still 0, but we have not yet run out of time to ensure that τ be greater in lexical order than σ. Hence, we take the disjoint union of two sets: one set contains all the results of adding $n<k\leq r+1$, to τ, and since each of these sequences is properly greater than the original σ, b is switched from 0 to 1. The other set contains just the result of copying whatever number σ has in this position and leaving b=0. Again, when b=0, no number less than the number σ has in this position is ever added to τ, or τ would be less than σ in the lexical order. Finally, there is the case 4 in which b=1. In this case, it does not matter what is added, so long as it is less than r+1, for τ is already lexically greater than σ.

For example, consider the computation of ATOM2(<11>,2,Λ,0):

```
ATOM2(<11>,2,<>,0) [case 3]
        ATOM2(<1>,2,<1>,0) [case 2]
                ATOM2(<>,2,<12>,1) [case 1]
                        <12>
                ATOM2(<>,3,<13>,1) [case 1]
                        <13>
        ATOM2(<1>,2,<2>,1) [case 4]
                ATOM2(<>,2,<21>,1) [case 1]
                        <21>
                ATOM2(<>,2,<22>,1) [case 1]
                        <22>
                ATOM2(<>,2,<23>,1) [case 1]
                        <23>
        ATOM2(<1>,3,<3>,1) [case 2]
                ATOM2(<>,2,<31>,1) [case 1]
                        <31>
                ATOM2(<>,2,<32>,1) [case 1]
                        <32>
                ATOM2(<>,2,<33>,1) [case 1]
                        <33>
                ATOM2(<>,2,<34>,1) [case 1]
                        <34>
```

Notice that the pair <11> is missing from the list, for it is the only element of ATOM(2,2) to be lexically less than or equal to the given pair <11>. Also of interest are the places in which the value of b is switched from 0 to 1. This takes place when 2 is added to τ=<1> and when 2 and 3 are added to τ=<>. This is proper, for <1> and <> are subsequences of σ=<11>, but <2>, <12>, and <13> are not.

Finally, consider CANON2.

CANON2(δ,π,τ):

1. = $\{\tau\}$
   if $\delta = \Lambda$;

2. = CANON2(σ,Λ,τ)
   if δ=<0,y>*σ;

3. = U{CANON2(σ,β,γ*β): $\beta \epsilon$ ATOM(y,$|\gamma|$}
   if $\pi=\Lambda$, δ=<x,y>*σ, and
   neither $\delta = \Lambda$ nor x=0;

4. = U{CANON2(σ*<x−1,y>,β,γ*β) and
   $\beta \epsilon$ ATOM2(π,$|\gamma|$,Λ,0)}
   if δ=<x,y>*σ and
   neither $\sigma=\Lambda$ nor $\delta=\Lambda$ nor x=0
   (i.e. otherwise).

where $|\Lambda|$ is 0 and $|\gamma|$ is the greatest integer occurring in γ if γ is not Λ.

Technically, CANON2 has three input parameters. δ is an arbitrary "segmentation" or finite list of ordered pairs of natural numbers. The parameters π and τ, however, are always initialized to the empty sequence Λ. Inductively, τ holds some output sequence "under construction", and π holds the variable specification for signed predicate $\pm$P that was added to τ.

The first case covered in the definition of CANON2 is that in which δ is empty. The segmentation δ gets shorter just as the sequence under construction gets larger, so that δ disappears just when π is completed. Hence, when δ is empty, it is time to return π. The singleton brackets are required because the output set is inductively the disjoint union of these singletons. The next case is that in which δ is not empty, and the first pair occurring in δ is <0,y>. If the first pair in δ is <x,y>, CANON2 recursively decrements x until it is zero, adding the variable specification of an atom at each decrementation. When zero is reached, all the variable specifications for y−ary atoms of the type accounted for by the current pair <x,y> have been added, so it is time to look at the next pair <x',y'> in δ. This is accomplished by decrementing δ. Moreover, the decrementation of δ means that CANON2 is now specifying atoms with a distinct signed predicate. Hence, the previous π is irrelevant and is re−initialized to Λ. Sequence τ is not altered. In the third case, π is the empty string, δ=<x,y>*σ and neither $\delta=\Lambda$ nor x=0. Since $\pi=\Lambda$, we are just beginning to add variable patterns for a new signed predicate of arity y. Hence, any variable pattern of length y that when added to τ results in a canonical

sequence may be added to τ. And this is just the set of y-sequences ATOM(y,|τ|). Otherwise, δ=<x,y>*σ and neither σ=Λ nor δ=Λ nor x=0. In this case, every variable specification of length y that is lexically greater than π is added to τ, and x is decremented by one.

Consider, for example, the case of the segmentation <<2,2>>.

```
CANON2(<<2,2>>,<>,<>)  [case 3]
        CANON2(<<1,2>>,<11>,<11>) [case 4]
                CANON2(<<0,2>>,<12>,<1112>) [case 2]
                        CANON2(<>,<>,<1112>) [case 1]
                                {<1112>}
                CANON2(<<0,2>>,<21>,<1121>) [case 2]
                        CANON2(<>,<>,<1121>) [case 1]
                                {<1121>}
                CANON2(<<0,2>>,<22>,<1122>) [case 2]
                        CANON2(<>,<>,<1122>) [case 1]
                                {<1122>}
                CANON2(<<0,2>>,<23>,<1123>) [case 2]
                        CANON2(<>,<>,<1123>) [case 1]
                                {<1123>}
        CANON2(<<1,2>,<12>,<12>) [case 4]
                CANON2(<<0,2>>,<13>,<1213>) [case 2]
                        CANON2(<>,<>,<1213>) [case 1]
                                {<1213>}
                CANON2(<<0,2>>,<14>,<1214>) [case 2]
                        CANON2(<>,<>,<1214>) [case 1]
                                {<1214>}
                CANON2(<<0,2>>,<21>,<1221>) [case 2]
                        CANON2(<>,<>,<1221>) [case 1]
                                {<1221>}
                CANON2(<<0,2>>,<22>,<1222>) [case 2]
                        CANON2(<>,<>,<1222>) [case 1]
                                {<1222>}
                CANON2(<<0,2>>,<23>,<1223>) [case 2]
                        CANON2(<>,<>,<1223>) [case 1]
                                {<1223>}
                CANON2(<<0,2>>,<31>,<1231>) [case 2]
                        CANON2(<>,<>,<1231>) [case 1]
                                {<1231>}
                CANON2(<<0,2>>,<32>,<1232>) [case 2]
                        CANON2(<>,<>,<1232>) [case 1]
                                {<1232>}
                CANON2(<<0,2>>,<33>,<1233>) [case 2]
                        CANON2(<>,<>,<1233>) [case 1]
                                {<1233>}
                CANON2(<<0,2>>,<34>,<1234>) [case 2]
                        CANON2(<>,<>,<1234>) [case 1]
                                {<1234>}
```

The set generated and the subset of CANON(4) ignored are as follows:

generated ignored

1112 1211
1121 1212
1122
1123
1213
1214
1221
1222
1223
1231
1232
1233
1234

Notice that CANON2, ATOM, and ATOM2 do essentially nothing but build the output strings. No mistaken concatenation is ever performed, and no "search" is required to prevent such mistakes. Little more could be asked of a finite set generator. Needless to say, the generational complexity of CANON2 is constant if the size of the output is measured as the sum of the lengths of the generated output.

An obvious next step would be to incorporate principles that eliminate tautologies into CANON2. This would require that a clausal blank rather than a segmentation be fed as input, for segmentations provide no information as to which pair <x,y> corresponds to which predicate or sign. The obvious procedure would be to check each element of ATOM or ATOM2 proposed to specify the negation of P against each specification already given for P.[122]

6.2.7. Loose Ends

Recall the set CANON2(<<2,2>>,<>,<>):

1112*
1121*
1122*
1123*
1213*
1214*
1221
1222*
1223
1231
1232*
1233*
1234*

[122] recall that each occurrence of -P follows each occurrence of +P in a canonical clause.

The sequences marked with an asterisk are those that specify non-reduced clauses with respect to <<2,2>>. For example, 1222 specifies

$$(.)[Px_1x_2 \vee Px_2x_2]$$

which is logically equivalent to

$$(.)[Px_2x_2].$$

Hence, there remains significant motivation to find an elegant generator that ignores non-reduced clauses. Unfortunately, it is not obvious how this might be accomplished. Unlike sequences suited to a segmentation, it is not the case that a clause is reduced if and only if each finite initial segment of the clause is reduced. For example,

$$(.)[Px_1x_2 \vee Px_3x_4]$$

is non-reduced, but its extension to

$$(.)[Px_1x_2 \vee Px_3x_4 \vee Px_2Px_3]$$

is reduced. So the complete set of these clauses cannot be generated by starting with the empty clause and by adding all the atoms that result in a reduced clause at each addition, or the latter clause would be missed. Hence, any adequate, elegant generator must somehow add at each stage all the atoms that *could* result in a reduced clause if the *right* atoms are added later on. This problem illustrates the sorts of concrete programming issues that the previous discussion of generational complexity was intended to capture formally.

6.3. Taking Inventory

We can now imagine an improved version of CONSIST called CONSIST2 that reads the nth evidence sentence at each stage n, employs CANON2 to generate exactly the specifications suited to each canonical blank of length no greater than n, and finally tests each resulting clause against the evidence read so far. In a strong sense, this improved version of CONSIST safely ignores *a priori* all clauses that have non-canonical blanks or whose variable patterns are unsuited to their blanks. The redundant clauses are ignored *safely*, for the inductive strategy pursued by CONSIST2 is identical to that pursued by CONSIST. Hence, their inductive scopes are the same *a fortiori*, and it is possible to speak of the latter as being more efficient than the former. The redundant clauses are ignored in a *strong sense*

because the generational complexity of CANON2 is constant in $\log_2(n)$. Hence, it cannot be said that CANON2 in any sense performs a "search" of a space larger than the set generated by CANON2 by an exponential factor. And finally, the clauses are ignored *a priori* because the procedure CANON2 that fails to consider them does not receive the evidence against which the generated clauses are tested as an input.

CONSIST2 represents an obvious improvement in inductive efficiency over CONSIST. But no one is yet tempted to call CONSIST2 efficient. For one thing, we have already seen that the consistency test applied to each clause generated by CANON2 must solve an NP-complete problem, and hence may be expected to "explode" in difficulty as n increases. So it is of the utmost practical importance that some simple method be found to prevent the empirical test of any clause whose test can be avoided without altering the inductive strategy.

Second, even if some method is developed to withhold test from clauses that do not require it, just considering whether to test all these clauses is unrealistically expensive. Ideally, more clauses should be ignored. But the quest of ignoring hypotheses *a priori* has been pursued to the point of diminishing returns in this chapter. Further improvements in efficiency that do not alter the inductive strategy employed must make use of the given evidence. That is, further hypotheses must be ignored *a posteriori*. This vital topic is addressed in the next chapter.

Chapter 7
Hypothesis Entailment and Inductive Efficiency

7.1. Entailment and Useless Hypotheses

In the previous chapter, a procedure was developed that can efficiently generate a canonical subset of the set of all clauses of length n that contains, nonetheless, a canonical representative of every logical equivalence class of such clauses. If this smaller, canonical set is exhaustively searched for suitable hypotheses at stage n, then significantly fewer hypotheses are considered or tested at each stage. In fact, the number of hypotheses is reduced by an exponential factor. And this reduction in cost is achieved without altering the inductive strategy employed; and hence with no compromise in inductive generality.

But in light of the evidence, it may be possible to ignore canonical as well as non-canonical elements of H without altering the inductive strategy or compromising inductive generality. Consider a simple analogy. Assume that the problem is to compute f(x)=2x over the natural numbers. No odd number need ever be considered by a procedure that computes this function, no matter what the input. So the odd numbers may be ignored *a priori* by a method for solving this problem. But for any given input n, the even numbers that are not identical to 2n may also be ignored *a posteriori* by such a procedure. In fact, any good algorithm for doubling a number should ignore ignore all possible outputs except for the actual one.

The purpose of this chapter is to investigate the prospects for exploiting the available evidence in order to ignore sentences that do not ultimately appear in the output conjecture. It would constitute a significant improvement over HEMP, NICOD and CONSIST merely to withhold the expensive suitability test from some canonical clauses. But it would be far better to avoid any consideration or representation of the clauses that need not be tested.

7.1.1. Ignoring Suitable Hypotheses

Recall that two conjecturing behaviors are equivalent just in case on each input their outputs are logically equivalent. Devices that pursue the same strategy obviously have identical limiting scopes. Therefore, the inductive strategy is unaltered if any element of Hyps(n) that is properly entailed by some *suitable* element of Hyps(n) is deleted from the nth conjecture. That is, only the set of logically *maximal* elements of the set of suitable clauses in Hyps(n) need be conjectured at stage n. So we have the following principle for restricting the search for a conjecture in Hyps(n):

Principle I: If h is suitable given e then *ignore* any h' such that h⊨h'.

That is, the property of failing to be maximally suitable in Hyps(n) is closed downward in the entailment ordering on the logical equivalence classes of Hyps(n).

7.1.2. Ignoring Unsuitable Hypotheses

If unsuitability is closed *upward* over Hyps(n) with respect to entailment, then we are also free to ignore upward closed sets in Hyps(n). The upward closure of unsuitability is, of course, just the contrapositive of the downward closure of suitability under entailment:

If h⊨h' and conf(e,h) then conf(e,h').

Hempel calls this property the *special consequence condition*, to distinguish it from the stronger *consequence condition*, which requires that any consequence of an arbitrary set of suitable hypotheses must be suitable. The consequence condition entails the special consequence condition, but not conversely, and the special consequence condition entails the equivalence condition, but not conversely. Since Hempel's confirmation criterion satisfies the consequence condition for arbitrary function-free languages it also satisfies the special consequence condition for these languages (and hence over Hyps(n)). The relation of consistency with the evidence fails to satisfy the the consequence condition, but does satisfy the special consequence condition for arbitrary first-order languages.[123] Finally, we saw that the modified Nicod's criterion violates the equivalence condition, even for function-free clauses, and hence must violate the special consequence and consequence conditions over these languages.

So at least in the case of consistency with the evidence and Hempel's criterion,

---

[123] If e,h are consistent they share a model M. If h⊨h' then M is a model of h', so e,h' are consistent.

there is also a useful heuristic for avoiding the consideration of unsuitable hypotheses.

Principle II: If $-S(e,h)$ then *ignore* any h' such that $h' \models h$.

7.2. The Entailment Structure of Canonical Clauses

We now have two principles that may enable us to design a general procedure that ignores gratuitous hypotheses on the basis of the evidence given. The first is to ignore any consequence of an hypothesis known to be suitable, and the second is to ignore any hypothesis with a consequence known to be unsuitable. From a logical perspective, these principles are obvious. But as was emphasized in chapter three, knowing which hypotheses not to consider is just a small step toward knowing how not to consider them. The trick is not to characterize what to ignore, but to provide a method that ignores it. And in some cases (e.g. when the problem is provably intractable) there may simply be no method with the desired input-output behavior that does so.

Principles (I) and (II) imply that upward and downward closed sets in the entailment ordering on canonical clauses may be ignored in light of the evidence. Whether there is a general inductive procedure that can ignore these sets depends on the mathematical structure of entailment on canonical clauses. As it turns out, central aspects of the algebraic and combinatoric properties of these structures are quite familiar to mathematicians. The purpose of this section is to explore some of these properties, with emphasis on those that may be of computational interest.

7.2.1. Variable Patterns and Entailment

Recall that clauses were analyzed in the previous chapter into *clausal blanks* and *variable patterns*. A clausal blank is just some clause with its variable occurrences replaced by asterisks. A variable pattern is just a finite sequence of natural numbers. A variable pattern s is *for* a blank B just in case its length is identical to the number of asterisk occurrences in B is identical to the length of σ. Clause c is the specification of B corresponding to σ just in case c is the result of substituting x_{σ_i} for the ith asterisk occurrence in B.

Variable pattern σ was said to be canonical just $\sigma_1=1$ and $\sigma_n=k$ only if $k-1$ occurs in some position $n'<n$. A clausal blank is canonical just in case its predicates all occur in ascending order according to their subscripts. A clause is canonical just in

case it is the specification of a canonical blank that corresponds to a canonical variable pattern. The set of all canonical specifications of a given blank is called Canon(B), and the set of canonical variable patterns of length n is called Canon(n).

Recall that the procedure CANON2 restricts the class of clauses considered still further, to eliminate what were called predicate permutation redundancies. For reasons of mathematical simplicity, I shall abstract from these further considerations in the following discussion.

If σ,τ are finite sequences of natural numbers, then we say that τ is *more general than* σ just in case there is a substitution θ such that $\sigma=\tau\theta$. This relation will be written as $\sigma\leq\tau$, and is a quasi-order.[124] Also, σ is *equivalent* to τ just in case $\sigma\leq\tau$ and $\sigma\geq\tau$. Notice that there is exactly one canonical sequence in each equivalence class of such sequences. Hence, the relation $\leq$ is a partial order over the set of canonical variable patterns of any given size.

Let c be the specification of B corresponding to τ and let c' be the specification of B corresponding to σ'. Then $\sigma'\geq\sigma$ just in case c'|=c, and that σ is equivalent to σ' just in case c,c' are logically equivalent. So the entailments among different results of specifying the same clausal blank B can be characterized entirely in terms of the $\leq$ relation over variable patterns for B.

7.2.2. Finite Set Partitions and Canonical Sequences

Happily, it turns out that the entailment order of the set of specifications of a given clausal blank with canonical variable patterns is isomorphic to a structure familiar in lattice theory, combinatorics, and computer science applications. A *partition* of a finite set S is just a set of mutually disjoint sets, the union of which is S. Define Part(n) as the set of all partition of the interval [1,n] of the natural numbers.

For any $\pi,\pi'\epsilon$Part(n), π is a *refinement of* π' just in case each cell in π is a subset of a cell of π'. Refinement is a partial order and may be written $\leq$. Then we have:

Fact L1:

<Part(n),$\leq$> is isomorphic to <Canon(n),$\leq$>,
where the isomorphism is simply
For any $\sigma\epsilon$Canon(n), $\Phi(\sigma)$=ker(σ)

---

[124] i.e. the order relation is transitive and reflexive.

and where ker(σ) denotes the kernel of σ:N(n)--->N(n).[125] For example, let σ=(1,2,2,1). $\sigma \in$ Canon(4). Viewed as a finite map, σ can be described as

i	σ_i
1	1
2	2
3	2
4	1

The kernel of σ is just {{1,4}, {2,3}}.

In view of this simple isomorphism, the number of distinct numbers occurring in canonical sequence σ is just the cardinality of $\Phi(\sigma)$. To highlight the analogy, I let $|\sigma|$ stand for the number of distinct numbers occurring in canonical sequence σ as well as for the cardinality of the corresponding partition. Similarly, the length len(σ) of canonical sequence σ corresponds to the cardinality of the disjoint union of the partition $\Phi(\sigma)$. So I also let len(π) denote the cardinality of the disjoint union of partition π.

7.2.3. The Mathematical Structure of Partition Lattices

Since refinement over Part(n) has the same structure as entailment over the clauses corresponding to canonical sequences, we can learn about our logical problem by reviewing what is known about partition lattices.

First of all, for any n, <Part(n),$\leq$> is a finite, semi-modular, relatively complemented lattice with unique maximal and minimal elements 1 and 0, respectively. For n=1,2,3, Part(n) is also modular. If n>3, it is not [Donnellan66] p.197 ff.[126]

---

[125] Let f:A--->B. Then for any a,a' in A, a is f-equivalent to a' just in case f(a)=f(a'). The kernel of f is the partition of A induced by this relation. It is now easy to see that Φ is an isomorphism between Canon(n) and Part(n). First a

Lemma (*): Let σ,σ' be strings of length n. Then ker(σ)$\leq$ker(σ') just in case $\sigma \leq \sigma'$. Proof: Assume ker(σ)$\leq$ker(σ'). If i,j are σ'-equivalent then they are also σ equivalent. So for any positions i,j, if $\sigma'_i=\sigma'_j$ then $\sigma_i=\sigma_j$. Hence $\sigma \leq \sigma'$. Reversing the steps of this argument yields the converse.

We must show that Φ is (A) surjective, (B) injective, and (C) structure preserving. (A) Φ is surjective: Let $\pi \in$ Part(n). Choose any bijective f:π--->N(|π|). There is one because the cardinalities of the sets are the same. Define $(\gamma(\pi))_i$= the value of f for the cell of π containing i. Therefore $\gamma(\pi)$ is a string of length n whose kernel is π. Let σ=the (unique) canonical sequence equivalent to $\gamma(\pi)$. ker(σ) = ker($\gamma(\pi)$), by lemma (*) and the equivalence of σ to $\gamma(\pi)$. By definition, σ is canonical. So there is a canonical string of length n whose kernel is π. (B) Φ is injective: Assume two canonical strings of the same length are not identical. Then they are inequivalent, for exactly one such string is in each equivalence class of strings of length n. By lemma (*), the respective kernels of these strings are inequivalent. (C) Φ is structure preserving: Assume that $\sigma,\sigma' \in$ Canon(n) and $\sigma \leq \sigma'$. By lemma (*), ker(σ)$\leq$ker(σ').

[126] In the diagram of Canon(4), note that the sublattice {1111, 11112, 1123, 1234, 1222} of Canon(4) is the pentagonal lattice, which cannot be a sublattice of any modular lattice.

Figure 7-1: Canon(2), Part(2)

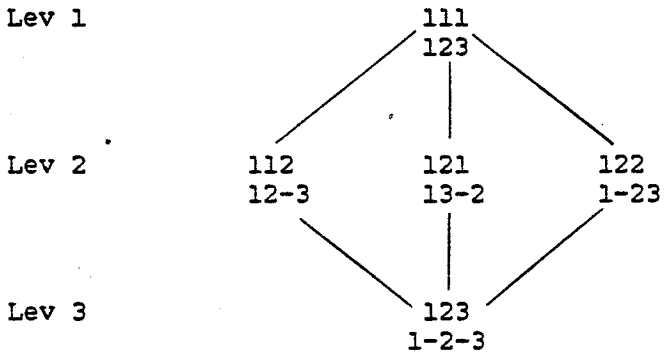


Figure 7-2: Canon(3), Part(3)

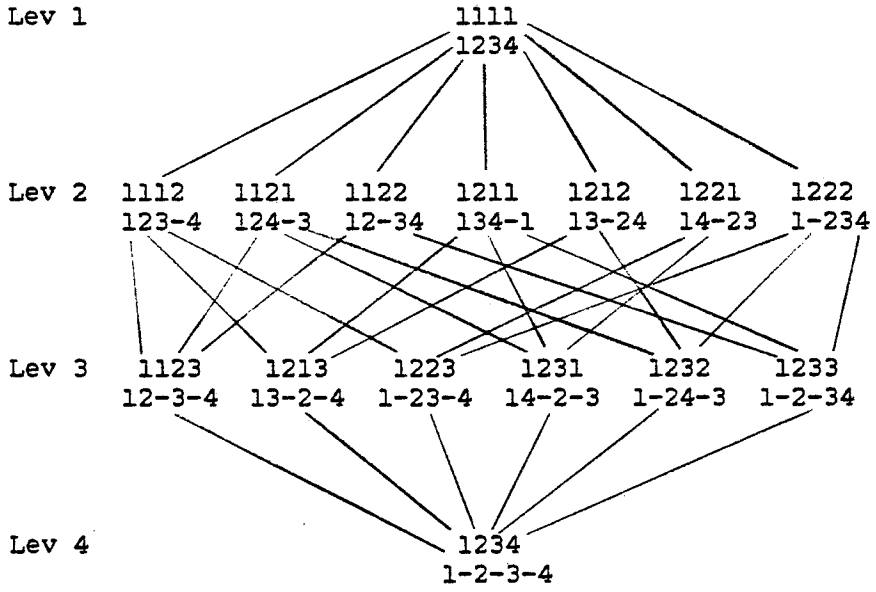


Figure 7-3: Canon(4), Part(4)

The *immediate descendant* relation $\pi << \pi'$ holds just in case $\pi < \pi'$ and there is no π'' such that $\pi < \pi'' < \pi'$. A *path* of length n from π to π' is just a sequence

$$\pi << \pi_1 << \ldots << \pi_n << \pi'.$$

Then each path from 1 (0) to π is identical in length to any other path from 1 (0) to π. Moreover, the length of the path from 1 to π is just the cardinality of π minus one, for if $\pi << \pi'$ then the cardinality of π is one less than the cardinality of π'.

Let Lev(n,k), the kth level of Part(n), be the set of all π in Part(n) such that $|\pi|=k$. Hence Lev(n,k) is also the set of all elements of Part(n) that are reachable from 0 by a path of length k−1. In virtue of the isomorphism between canonical sequences and partitions, the level of a canonical sequence is just the number of distinct numbers occurring in that sequence. The levels show up clearly in the respective figures of Part(2), Part(3), and Part(4), in which each level actually constitutes a "level" on the page.

7.2.4. Sizing Up Partition Lattices

Now that we have a characterization of our potential search space at each stage n, it is desirable to find out how large this space is. The values of the function

$$B_n = |\text{Part}(n)| = |\text{Canon}(n)|$$

are called the *Bell numbers*, and are fundamental quantities in combinatorial theory. Moreover, the values of the binary function

$$S_{n,k} = |\text{Lev}(n,k)|$$

are called the *Stirling numbers of the second kind*, and are equally central to combinatorial theory.

It is immediate that

$$B_n = \Sigma_{1 \leq i \leq n} S_{n,i}$$

for a partition lattice is clearly the disjoint union of its levels, and the cardinality of a disjoint union is the sum of the cardinalities of the arguments to the disjoint union.

That the size of a partition lattice level is unacceptably large for exhaustive search to be feasible is evident from the following recurrence relation:

1. $S_{n,n} = 1$
2. $S_{n,0} = 0$ for $n>0$
3. $S_{n,k} = S_{n-1,k-1} + kS_{n-1,k}$.[127]

Consider just the first addend of (3). Notice that if $n>k>0$, then k will be multiplied by itself n−k times before the right-hand term "bottoms out" to $S_{k,k}=1$, as in the accompanying figure.Therefore, for fixed k, a lower bound on $S_{n,k}$ is

---

[127] The adequacy of the recursion may be seen as follows. (1) Clearly, there is one canonical tuple of length n in which n distinct numbers occur, namely $\langle 1,2,\ldots,n\rangle$. (2) There is no canonical 0-tuple in which more than 0 distinct natural numbers occur, for no numbers occur in the empty tuple. (3) Finally, let $\sigma \in \text{Lev}(n,k)$, and let σ' be the result of truncating the nth position off of σ. Either σ_n occurs in σ' or not. If so, then $\sigma' \in \text{Lev}(n-1,k)$. If not, then $\sigma' \in (\text{Lev}(n-1,k-1)$. Assume $\tau \in \text{Lev}(n-1,k-1)$. Then there is only one canonical extension σ of τ such that $|\sigma|=k$, namely, $\tau^{\frown}k$. Moreover, this extension is distinct from the extension of any other element of Lev(n-1,k-1). So Lev(n-1,k-1) contributes $S_{n-1,k-1}$ elements to $S_{n,k}$. Now assume $\tau \in \text{Lev}(n-1,k)$. Since $|\tau|=k$ already, there are exactly k extensions σ of τ such that $|\sigma|=k$, namely, $\tau^{\frown}1,\ldots,\tau^{\frown}k$, and each of these extensions is unique. So Lev(n-1,k) contributes $kS_{n-1,k}$ elements to Lev(n,k). But Lev(n-1,k) is obviously disjoint from Lev(n-1,k-1), so $S_{n,k} = S_{n-1,k-1}+kS_{n-1,k}$.

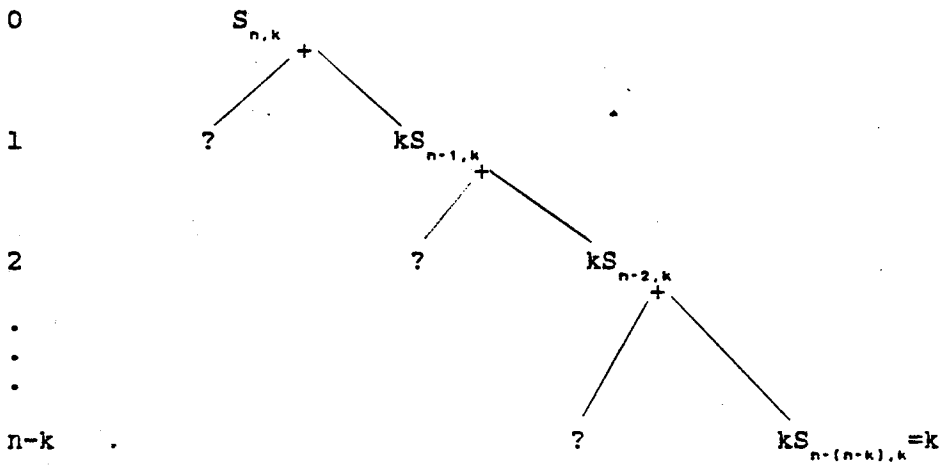


Figure 7-4: Exponential Lower Bound on $S_{n,k}$

$S_{n,k} > k^{n-k}$,

so the size of Lev(n,k) is exponential in n. This suffices to illustrate that exhaustive search is infeasible. A standard,[128] more precise expression for $S_{n,k}$ is

$S_{n,k} = 1/k![\Sigma(-1)^{k-i}(k \text{ choose } i)i^n]$

The accompanying three figures

k\n	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2		1	3	7	15	31	63	127	255	511	1023	2047
3			1	6	25	90	301	966	3025	9330	28501	86526
4				1	10	65	350	1701	7770	34105	145750	611501
5					1	15	140	1050	6951	42525	246730	1379400
6						1	21	266	2646	22827	179487	1323652
7							1	28	462	5880	63987	627396
8								1	36	750	11880	159027
9									1	45	1155	22275
10										1	55	1705
11											1	66
12												1

Figure 7-5: Value Table for $S_{n,k}$

[128] [Aigner79], p. 97.

n	B_n
1	1
2	2
3	5
4	15
5	52
6	203
7	877
8	4140
9	21147
10	115975
11	678570
12	27644437
13	190899322

Figure 7-6: Value Table for B_n

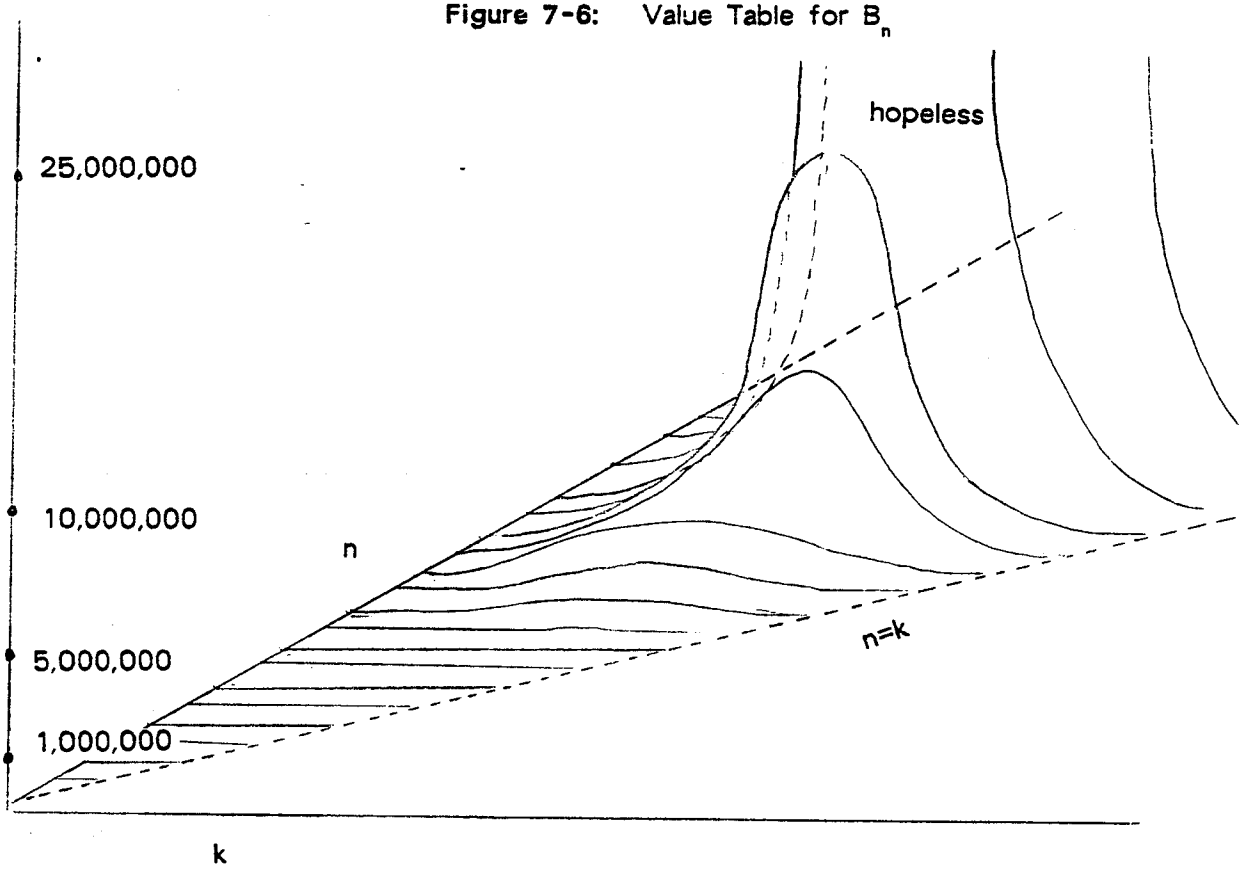


Figure 7-7: Plot of $S_{n,k}$

provide some practical perspective on the impressive explosion of the search space as n increases.

Admittedly, if a predicate occurs more than once in a clausal blank, many of the points in the corresponding partition lattice will be redundancies weeded out by the procedure CANON2. But when each predicate is distinct, each point in the lattice is

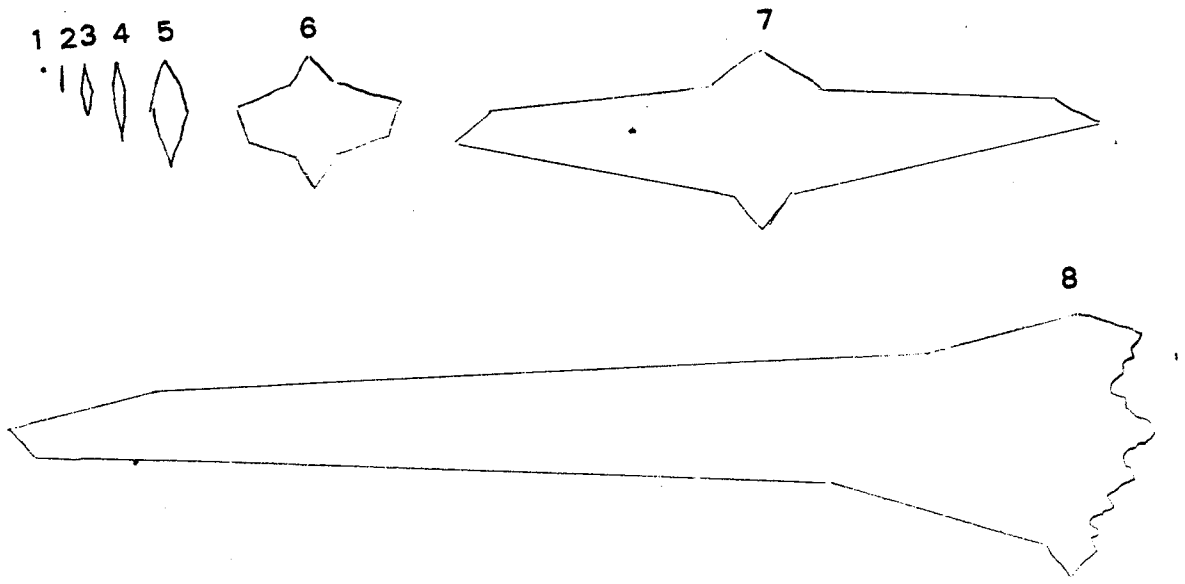


Figure 7-8: The First Eight Partition Lattices Drawn to Scale

generated even by CANON2. And a blank with seven distinct binary predicates, for example, would have 190,899,322 distinct, canonical specifications. The need for a method that ignores upsets and downsets in a partition lattice is therefore quite serious.

7.2.5. Upward and Downward Closed Sets in Partition Lattices

Let $\langle R,\leq\rangle$ be an arbitrary, partially ordered set or *poset* for short. Let S be a subset of R. S is an *upset* (*downset*) of R just in case for each $\pi \in R$ and for any $\pi' \in S$, if $\pi \leq \pi'$ ($\pi \geq \pi'$) then $\pi' \in S$. For any subset S of R, the upset (downset) *generated by* S is just the upward (downward) closure of S in R. These sets are denoted up(S) and down(S), respectively. In the special case when S is a singleton {a}, we dispense with the curly brackets by convention and simply write up(a) (down(a)). An upset or downset that is generated by a singleton is called *principal*, in analogy to principal ideals and filters in lattice theory.

In light of these definitions, and of principles (I) and (II) for ignoring clauses, a significant component of inductive intelligence is the ability to ignore upsets and downsets in partition lattices, which are posets of a particular kind. That is to say,

we want to generate only elements of complements of upsets and downsets in a partition lattice, and to do so in such a manner that we never fail to generate one when there is one. And we want to do so without tacitly considering elements of the upsets and downsets to be avoided.

But for any poset R, it is simply a matter of logic that if S is an upset (downset) then the complement of S in R is a downset (upset).[129] So ignoring an upset (downset) amounts to generating exactly some downset (upset) that is the complement of the given upset (downset).

Since computational elegance can be thought of as the result of allowing general mathematical truths about structure to stand in for repeated, expensive decisions, it is of the utmost importance to our project to examine what structure there is in upsets and downsets in partition lattices.

An arbitrary upset (downset) is just the union of a set of principal upsets (downsets). In particular, Up(S) (Down(S)) is just the union of Up(a) (Down(a)) for all $a \epsilon S$. And principal upsets and downsets of partition lattices have a well-known, elegant, and computationally useful characterization [Aigner79].

Principal Upsets

Consider first the case of principal upsets, for it is simplest. Let $\pi \epsilon$ Part(n). Then

Fact L2:

up(π) is isomorphic to Part($|\pi|$),

The argument for Fact L2 is simple. Each ancestor of π can be thought of as the result of taking the disjoint union of some cells in π. But the space of possible disjoint unions of c sets is isomorphic to the space of possible disjoint unions of c singletons. But a partition consisting entirely of $|\pi|$ singletons is just the bottom of a partition lattice of size $|\pi|$. This result shows that partition lattices have a sort of "upward-nested" structure, depicted in the accompanying figure.

For example, consider the lattice generated by the up(121131). Although 121131 is an element of a lattice with 203 elements, we can easily calculate its principal upset to be the lattice in the accompanying figure.

---

[129] Just take the contrapositive of the definition of an upset (downset).

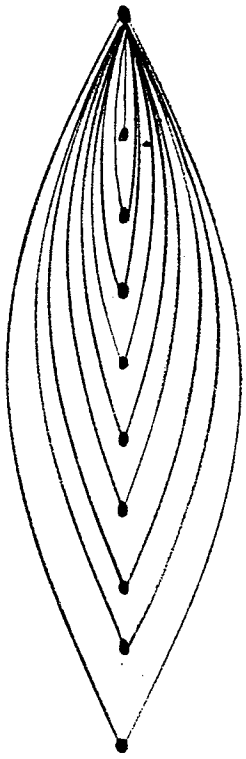


Figure 7-9: The Nested Principal Upsets of a Partition Lattice

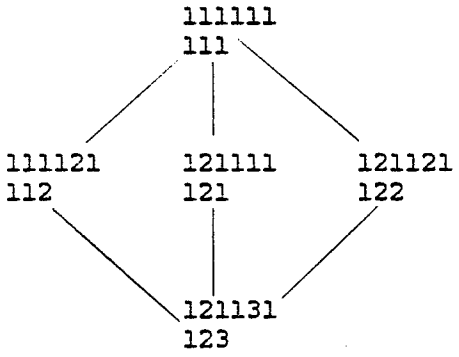


Figure 7-10: up(121131)

Principal Downsets

To formulate the corresponding characterization for principal downsets the *lattice product* operation must first be introduced. The lattice product (S X T) of lattices $S=\langle S,\leq\rangle$ and $T=\langle T,\leq'\rangle$ is given as the structure $\langle (S \times T), \leq''\rangle$, where (S X T) is the Cartesian cross of S and T, and $(x,y)\leq(x',y')$ just in case $x\leq x'$ and $y\leq' y'$. I shall not distinguish AX(BXC) from (AXB)XC, so I may write $L_1XL_2X...XL_n$. Finally, the class of lattices is closed under finite lattice products. Now it can be stated that:

Fact L3: [Aigner79]

down(π) is isomorphic to $\Pi_{c \in \pi}$Part(|c|).

where Π is the generalized *lattice product* operation. The isomorphism f_π from down(π) to Part(|π|) works as follows. Let π be a partition $\{c_1,\ldots,c_n\}$ of N(m). Any refinement of π is the result of removing some c_i from π, forming a partition $\tau=\{s_1,\ldots,s_k\}$ of c_i, and taking the union $\pi \cup \tau$. Therefore, to each such τ we can assign a unique m-tuple σ of partitions--- one partition for each c_i--- such that $[\pi - c_i] \cup \sigma_i = \tau$. This map is clearly an isomorphism between the structures in question.

This result is illustrated in the accompanying figure, in which the first label of each vertex is a sequence of canonical tuples representing an element of Part(3)XPart(2) and whose second label is an element of down(11212), where 11212 is taken to represent the partition that is its kernel.

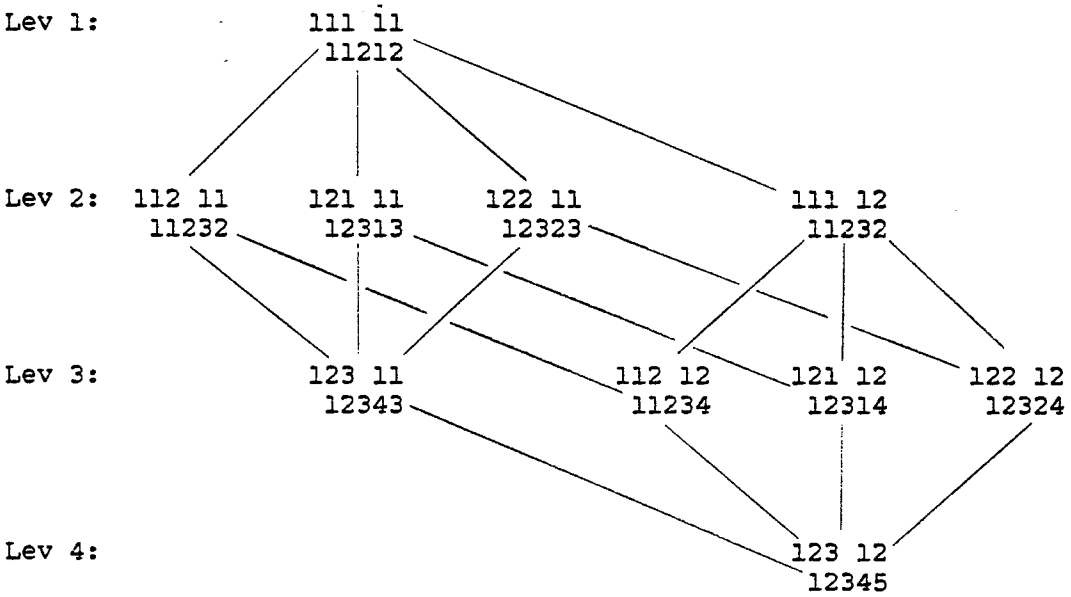


Figure 7-11: The Lattices Canon(3) X Canon(2) and down(11212)

Curiously, down($\{\sigma\}$) is isomorphic to a finite, non-trivial Boolean algebra if each number occurring in σ occurs in σ exactly two times, since the n-element Boolean algebra is is isomorphic to the n-fold lattice product of Canon(2), the one element Boolean algebra. For example, consider the canonical sequence σ=(121233). down(σ) is then the boolean algebra on 3 elements. In the figure, the first vertex label corresponds to Canon(2)^3, the second corresponds to the three-element boolean algebra and the third corresponds to down(121233).

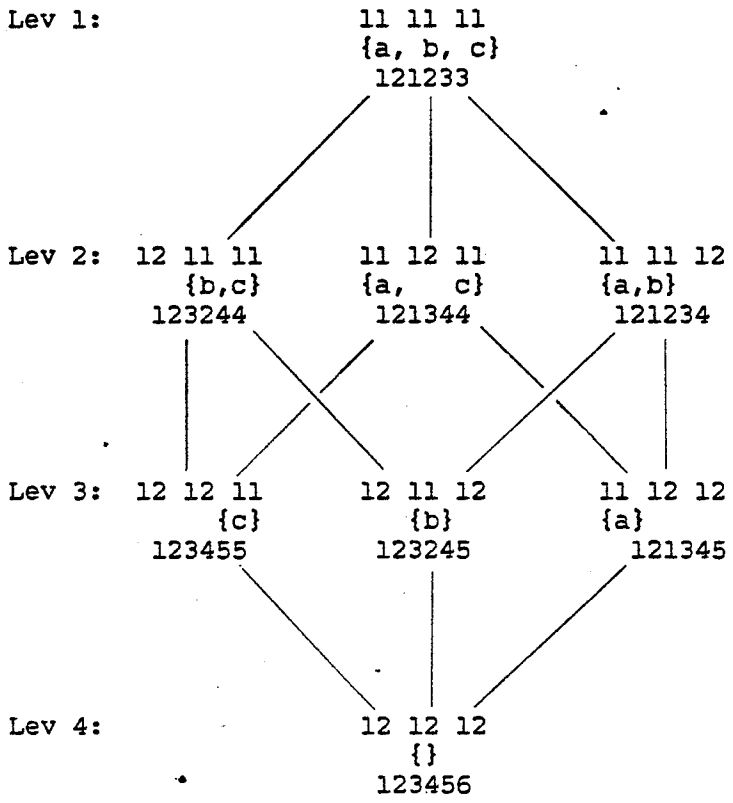


Figure 7-12: Canon(2)^3, the three element Boolean algebra, and down(121233)

Now we know that each principal downset of a partition lattice is a lattice product of partition lattices. A natural question to ask at this point is whether *every* product of partition lattices is a principal ideal in some partition lattice. Indeed, we have that

Fact L4:

Part(n)XPart(m) is isomorphically embeddable in Part(n+m).

The embedding function is

$f(\langle \pi, \pi' \rangle) = \pi \cup (\pi' + n)$.

where π is in Part(n), π' is in Part(m) and $\pi+n$ denotes the result of adding n to each element of each cell of π.[130] (For example, {{1,3},{2}}+4={{5,7},{6}}). This embedding function f can be generalized easily to arbitrary n-tuples of partitions.

$f(\langle \pi_1, \ldots, \pi_m \rangle) =$

$\bigcup_{1 \leq i \leq m} \pi_i + \sum_{1 \leq j \leq i} \text{len}(\pi_j)$.

For example,

---

[130] Function f is clearly from Part(n)XPart(m) to Part(n+m), for the value of f is always a partition of N(n+m). f is injective, due to the fact that the partitions over which the union is taken are of disjoint sets. Moreover, structure is preserved, for suppose that (π,π'), (ρ,ρ') are in Part(n)XPart(m) and that $(\pi,\pi') \leq (\rho,\rho')$, but that $f(\pi,\pi') > f(\rho,\rho')$. Then $\pi \leq \rho$ and $\pi' \leq \rho'$, but $\pi \cup (\pi'+n) > \rho \cup (\rho'+n)$. So each cell of $\pi \cup (\pi'+n)$ is included in some cell of $\rho \cup (\rho'+n)$, and at least one of these inclusions is proper. Then either $\pi > \rho$ or $\pi'+n > \rho'+n$. In either event, the hypothesis contradicted.

f(<{{1,3},{2,4}}, {{1,2},{3}}, {{1},{2},{3}}>)=

{{1,3},{2,4}} U {{1+4, 2+4}, {3+4}} U {{1+7}, {2+7}, {3+7}}=

{{1,3},{2,4},{5,6},{7},{8},{9},{10}}

minor modification of the previous argument yields the more general result

Fact L5:

$\Pi_{i \in I}$Part(i) is embeddable in Part($\Sigma_{i \in I}$)

where the embedding is the generalized version of f.

7.3. Concise Data Structures for Partition Lattice Search

A rose is a rose by any other name--- so far as meaning is concerned. But it is widely acknowledged by computer scientists that the choice of a language or representational system in writing a program can have a significant impact on computational efficiency.

In this section, I propose data structures called PEXPRs ("Partition EXPRessions") for representing useful subsets of a partition lattice. PEXPRs express *constraints* on partitions. Intuitively, the PEXPR '1|3|4' *describes* or *is satisfied by* any of the following elements of Part(4):

{{1,2},{3},{4}}
{{1},{3,2},{4}}
{{1},{3},{2,4}}
{{1},{2},{3},{4}}.

The PEXPR notation '123|4' is taken to be neutral regarding the equivalence of 1,2, and 3. When we really want to say that these numbers are equivalent, we can write '(123)|4,' where numerals enclosed in parentheses are taken to be in the same partition cell. So, for example, '123|4' is satisfied by the following five elements of Part(4)

{{1,2,3},{4}}
{{1},{2,3},{4}}
{{1,2},{3},{4}}
{{1,3},{2},{4}}
{{1},{2},{3},{4}}

while '(123)|4' is satisfied only by the 4-partition {{1,2,3},{4}}.

It prevents confusion in applications to take the "satisfaction" metaphor seriously and to treat PEXPRs model-theoretically, where relational structures correspond to finite set partitions and PEXPRs are axiomatizations of sets of these structures. This formal development is accomplished in the following two subsections.

7.3.1. PEXPR Syntax

Let CON be an arbitrary, finite set. I shall always employ small numerals as CONs. A CEXPR is any finite, non-empty string of CONs surrounded by parentheses. For example, (123) is a CEXPR. A TERM is any finite, non-empty string of CONs and CEXPRs. For example, '5(123)4' is a TERM. A Proper PEXPR is either a TERM or a finite sequence of TERMs separated by vertical bars. So '6|5(123)4|9' is a Proper PEXPR. A Composite PEXPR is some formula in which proper PEXPRs are connected in the usual manner by sentential connectives. So '-[6|4(12)|(3) v 5|(31)]' is a Composite PEXPR. We shall be interested in looking at rather large PEXPRs. Reasonably small problem instances can lead to PEXPRs with over four hundred disjuncts. Hence, a useful convention to keep them on the page is to denote disjunction by stacking the disjuncts vertically, and to denote conjunction by placing two PEXPRs side by side with just a space separating them. So for example the composite PEXPR,

```
(12)|3  (23)
245      2|4
(24)

(45)8
2|7
```

is a notational variant of

[[(12)|3 v 245 v (24)] & [(23) v 2|4]] v [(45)8 v 2|7]

Finally, it is useful to speak of Atomic PEXPRs, which are PEXPRs of the form (ij) or the form i|j.

7.3.2. PEXPR Semantics

A *partition* structure is a relational structure $\langle S,\underline{=}\rangle$, where S is a set and $\underline{=}$ is an *equivalence relation* on S. Evidently, each such structure corresponds to the finite partition of S induced by $\underline{=}$. So we can speak as well of a partition satisfying a PEXPR. A structure M is said to be *for* CON when it is paired with a map δ that takes each CON to some element of S. Now, for satisfaction:

> If P is a Proper PEXPR then M *p-satisfies* P (M |= P) just if for each CON c occurring in P, no CON c' separated from c in P by dashes is such that $\delta(c)\underline{=}\delta(c')$, and for each CEXPR C occurring in P, for each c,c' occurring in C, $\delta(c)\underline{=}\delta(c')$.
>
> If P is a Composite PEXPR, then the usual truth conditions for sentential connectives apply.

Now we can define *p-satisfiability, p-validity, p-entailment* and *p-equivalence* in terms of p-satisfaction in the usual, model-theoretic manner. So for example, the PEXPR '1|1' is p-unsatisfiable, for 1 is equivalent to itself. The PEXPR '(1)' is p-tautologous, for the same reason. The PEXPR '12' is p-valid because it trivializes the antecedent of the definition of satisfaction. So it turns out that '-12' is a strange sort of p-contradiction, which says "it is not the case that either 1 is equivalent to 2 or 1 is not equivalent to 2". PEXPR 12|3|(45) p-entails PEXPR 1(2)|345 and is p-equivalent to the PEXPR (1)2|3 & 12|45 & 3|(45).

The computational utility of PEXPRs as data structures can now be illustrated. For example, let P be the Proper PEXPR

1|2|3|4|5|6|7|8|9.

According to the definition of p-satisfaction just given, P is p-satisfied in just the same partition structures that satisfy the following, standard, logical expression:

-R(1,2)& -R(1,3)& -R(1,4)& -R(1,5)& -R(1,6)& -R(1,7)& -R(1,8)& -R(1,9)&
-R(2,3)& -R(2,4)& -R(2,5)& -R(2,6)& -R(2,7)& -R(2,8)& -R(2,9)&
-R(3,4)& -R(3,5)& -R(3,6)& -R(3,7)& -R(3,8)& -R(3,9)&
-R(4,5)& -R(4,6)& -R(4,7)& -R(4,8)& -R(4,9)&
-R(5,6)& -R(5,7)& -R(5,8)& -R(5,9)&
-R(6,7)& -R(6,8)& -R(6,9)&
-R(7,8)& -R(7,9)&
-R(8,9).

Evidently, the length of the standard expression is on the order of the length of P choose 2, which is on the order of the square of the length of P. So PEXPRs can reduce by a square factor the space required to axiomatize the same set of partition structures. Moreover, for most applications, the PEXPR representation requires no more time to use than the equivalent logical formula does. For example, to decide whether the above PEXPR entails that i is not equivalent to j, one searches, at worst, each pair of constants in the PEXPR to see if they are separated by at least one bar. In the case of the logical formula, however, the worst case would be that in which the desired atom is the last one inspected. In either case, the same number of pairs of constants would be inspected.

7.3.3. PEXPR Normal Forms

Since we have defined atomic PEXPRs, we can speak of CNF (conjunctive normal form) PEXPRs in the usual manner. For example, the following PEXPR is in CNF:

1¦2\2¦4\(54)
2¦3\3¦5
(15)\3¦4
3¦1.

The next example is a logically equivalent DNF (disjunctive normal form) PEXPR, obtained by taking the cross product of the columns in the CNF form.

1¦2, 2¦4, (54)
1¦2, 3¦5, (54)
1¦2, 3¦4, (54)
2¦3, 2¦4, (54)
2¦3, 3¦5, (54)
2¦3, 3¦4, (54)
(15), 2¦4, (54)
(15), 3¦5, (54)
(15), 3¦4, (54)
3¦1, 2¦4, (54)
3¦1, 3¦5, (54)
3¦1, 3¦4, (54)

Another normal form for PEXPRs is PNF, or *partition normal form*. A PEXPR P is called a *state description* just if it is of the following form: '1¦(23)¦(45)'. That is, P is a Proper PEXPR (no semicolons or line breaks), every CON occurs in P exactly once, and each constituent TERM of P is a single CEXPR. A PEXPR P is in PNF just if it is a column of state descriptions. Also, a PNF PEXPR P is *minimal* just if no two state descriptions occurring in P are logically equivalent. For example:

PEXPR:

12¦(34)5

Equivalent Minimal PNF PEXPR:

(12)¦(345)
(1)¦(2)¦(345)
(12)¦(34)¦(5)
(1)¦(2)¦(34)¦(5)

Notice that if P is a state description and ¦CON¦=k then $EC_k(P)$ is a unit set. Hence, the appellation "state description" is appropriate, for a state description says, once for all, everything about some partition in Part(k). Moreover, every consistent PEXPR is equivalent to a PEXPR in PNF.[131] So it is appropriate to call PNF a "normal form".

A normal form of computational interest is *disjoint normal form* or DJNF. A PEXPR in DJNF is a column of Proper PEXPRs such that no two PEXPRs in the

[131] If exactly k CONs occur in P, just disjoin the state descriptions corresponding to each element of $EC_k(P)$.

column are mutually consistent. The advantage of such a representation for a closed set in a partition lattice is evident. If we had a method for generating $EC_k(P)$ for any proper PEXPR P, then if a subset S of Part(k) were represented by a DJNF PEXPR R, applying this method to each disjunct of R would yield a non-redundant list of S.

7.3.4. PEXPR Facts

The following is a representative sample of useful PEXPR tautology-schemata (where x,y,z are metatheoretic variables ranging over elements of CONS, and A,B,C are metatheoretic variables ranging over CONS*, and X,Y,Z range over CEXPR U CONS*).

1. |= (xx) [reflexivity]
2. |= (ABCD) ---> (ACBD) [symmetry]
3. |= [(AB) & (BC)] <-+-> (ABC) [union (entails transitivity)]
4. |= (ABBC) <---> (ABC) [cancellation]
5. |= -(xy) <---> x|y [bar introduction]
6. |= X|Y|Z|W <---> X|Z|Y|W [permutation]
7. |= X|Y <---> X|x v Y|x [non-omnipresence][132]
8. |= [X|Y & X|Z] <---> X|YZ [merge]

A less obvious principle is the following:

|= [X|Y & x|y] <---> [Xx|Yy v Xy|Yx] [combination]

Proof: <--- Suppose partition structure M satisfies Xx|Yy v Xy|Yx. Then M|=[X|Y & X|y & Y|x & x|y] v [X|Y & X|x & Y|y & x|y]. Then M|=[X|Y & x|y] & [[X|y & Y|x] v [Y|y & X|x]]. So M|=X|Y & x|y. ---> Suppose M|=[X|Y & x|y]. But by non-omnipresence, X|Y implies [X|x v Y|x] and [X|y v Y|y], so [X|Y,x|y] is p-equivalent to P=

X|Y & x|y & X|x & X|y
X|Y & x|y & X|x & Y|y
X|Y & x|y & Y|x & X|y
X|Y & x|y & Y|y & Y|y

(recall that by convention, each row is a disjunct). But by non-omnipresence, X|Y implies [X|y v Y|y] and [X|x v X|y]. So [X|Y & x|y & X|x & X|y] is p-equivalent to

---

[132] Contrapositive of transitivity: (Xy) & (yZ) ---> (XZ), and bar introduction.

X|Y & x|y & X|x & X|y & X|y
X|Y & x|y & X|x & X|y & Y|y

and [X|Y & x|y & Y|y & Y|y] is p-equivalent to
X|Y & x|y & Y|y & Y|y & X|x
X|Y & x|y & Y|y & Y|y & X|y.

Substituting p-equivalents for p-equivalents in P, we find that X|Y,x|y is p-equivalent to
X|Y & x|y & X|x & X|y & X|y
X|Y & x|y & X|x & X|y & Y|y
X|Y & x|y & X|x & Y|y
X|Y & x|y & Y|x & X|y
X|Y & x|y & Y|y & Y|y & X|x
X|Y & x|y & Y|y & Y|y & X|y.

But notice that the first and second disjuncts entail the third, and the fifth and sixth disjuncts entail the fourth. So the X|Y & x|y is p-equivalent to [X|Y & x|y & X|x & Y|y] v [X|Y & x|y & Y|x & X|y], which is p-equivalent to Xx|Yy v Xy|Yx. Q.E.D.

The importance of the combination principle is that it provides, along with the merge principle, a procedure for converting a disjunction of Proper PEXPRs and a disjunction of paren-free atomic PEXPRs into a Proper PEXPR. For example, consider the PEXPR

5|7 (12)4|5
6|7 34|(15)

Notice that distributing conjunction over disjunction just amounts to taking the cartesian cross of columns in our notation. Taking this cross product yields a logically equivalent PEXPR

5|7,(12)4|5
5|7,34|(15)
6|7,(12)4|5
6|7,34|(15)

Finally, the result of applying the combination or merge principle (depending on which applies) to each disjunct is

(12)47|5 [merge]
347|(15) [merge]
(12)46|57 [combination]
(12)47|56
346|(15)7 [combination]
347|(15)6

Similarly, the union principle provides a way to convert any disjunction of bar-free, Proper PEXPRs and disjunction of bar-free atomic PEXPRs into a disjunction of Proper PEXPRs. For example, consider

(27) (12)(45)
(57) (34)(15)

which is equivalent to

(27),(12)(45)
(57),(12)(45)
(27),(34)(15)
(57),(34)(15)

By the union principle, we have

(127)(45)
(12)(457)
(27)(34)(15)
(34)(157).

Soon, we shall see applications of these principles in various lattice search algorithms.

7.3.5. PEXPRs and Partition Lattices

Recall that our interest in PEXPRs arises from an interest in partition lattices. Let CON(n) be the numerals '1',...,'n', and let δ be the usual interpretation of these numerals. Then the set of all partition structures whose domain elements are named by numerals in CON(k) according to δ is just the partition lattice Part(n).

The set of all elements of Part(n) that satisfy a PEXPR P whose constants are all members of CON(n) may be called the *n-elementary class* of P, or $EC_n(P)$ for short. In this manner, each PEXPR is associated with some subset of Part(n), for any n.

If P is parenthesis-free, then $EC_n(P)$ is a downset in Part(n) for a parenthesis-free PEXPR can assert that objects are in distinct partition cells, but not that any objects are in the same cell. Therefore, splitting any cell of a model of a paren-free PEXPR yields a model of this PEXPR. Similarly, if P is bar-free, then $EC_k(P)$ is an upset in Part(k) since such a PEXPR cannot assert that any two objects are in distinct cells. Hence, fusing together any two cells of a model of P yields a model of P. Herein lies the utility of PEXPRs in representing partition lattice upsets and downsets.

These considerations lead to a fact that is crucial to the application of standard complexity theory to problems involving upsets and downsets in partition lattices. An upset (downset) R in Part(n) can also be specified by the smallest subset S of Part(n) such that R=Up(S) (Down(S)). Call this set S the *maximal generating set* of R. One might imagine that maximal generating sets are also a rather efficient way to code upsets and downsets in Part(n). But:

Theorem P1: In the worst case, the cardinality of the maximal generating set of the k-elementary class of a bar-free (paren-free) PEXPR of length k is on the order of $2^{k/2}$.[133]

So PEXPRs can encode upsets and downsets in a partition lattice with exponentially fewer symbols than are required to encode the set by means of a list of its maximal generators. This is theoretically important, for recall that the complexity class to which a problem is assigned depends on how "bulky" its encoding is. Hence, encoding a problem involving partition lattice upsets and downsets in terms of maximal generators may make the problem seem exponentially easier than when the upsets and downsets are encoded by PEXPRs. We shall encounter this issue again very soon.

7.4. The Trouble with Partition Lattice Search

Recall the CONSIST strategy. At stage n, we must conjecture an axiomatization of all and only the clauses in which there are n or fewer variable occurrences that are consistent with the evidence received by stage n, where the evidence is a finite set of atoms or negations of atoms true in some unknown structure.

For the sake of simplicity in presentation, let us restrict attention to the lattice of canonical specifications of a single clausal blank B in which no predicate occurs more than once. As we have seen, the entailment structure on this set is isomorphic to Part(n), where '*' occurs n times in B.

7.4.1. Avoiding Upsets and Downsets

Recall that we are free to ignore any hypothesis entailed by a hypothesis consistent with the evidence, and that we are free to ignore any hypothesis that entails an hypothesis inconsistent with the evidence. So we can imagine an intermediate stage of empirical test in which some subset R=(S U F) of Part(n) has already been inspected, where S is the set of all "successes" or variable patterns of specifications of B that are consistent with the evidence, and F is the set of "failures", or variable patterns of specifications of B that are inconsistent with the evidence.

---

[133] Proof sketch: Consider the composite PEXPR P=[1¦2 & 3¦4 &...& k-1¦k]. [1¦2 & 3¦4] is logically equivalent to Q=[13¦24 v 14¦23] by the combination principle. But [Q & 5¦6] is logically equivalent to 135¦246 v 136¦245 v 145¦236 v 146¦235 by a second application of the combination principle. When this recursive application of the combination principle is completed, we have a disjunction with $2^{k/2}$ logically independent Proper PEXPRs as disjuncts, in which each numeral from 1 to k occurs exactly once, which is equivalent to the given PEXPR. State descriptions for the maximal elements of its elementary class can be obtained merely by placing parentheses around the terms between bars. Since there is exactly one element of Part(k) that satisfies each k-state description so obtained, the maximal generating set of the elementary class of the given PEXPR has cardinality on the order of $2^{k/2}$.

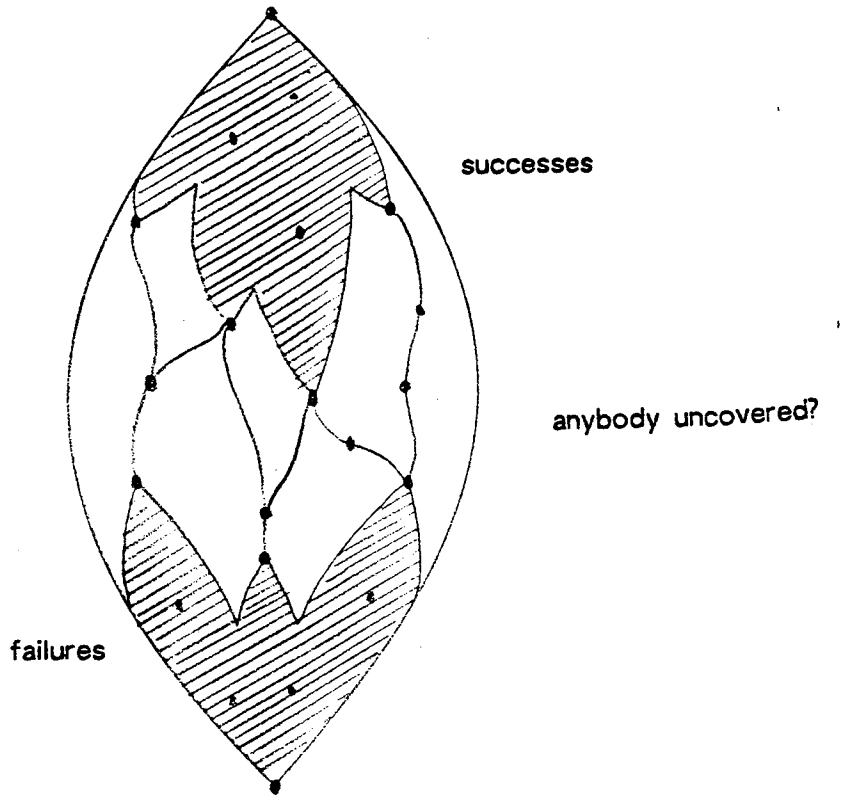


Figure 7-13: Avoiding Upsets and Downsets Simultaneously

The obvious next step is to find a new specification Π of B to test. But Π should not be a refinement of any element of F, and Π should not refine into any element of S. That is, Π should be in Part(n)−(Up(S) U Down(F)). So the problem is,

> Given S and F, to find an element Π of Part(n)−(Up(S) U Down(F)) if there is one, and to say there is no such Π if there is no such partition.

Recall that Up(S), Down(F) can in many cases be encoded with exponentially fewer symbols as PEXPRs. It would therefore seem sensible to encode Up(S),Down(F) by such expressions. Accordingly, let P_S be a bar-free PEXPR whose elementary class is Up(S), and let P_F be a paren-free PEXPR whose elementary class is Down(F). Recall that such PEXPRs must exist if S,F are non-empty.[134] So the problem may now be restated:

The PEXPR Elementary Class Complement Search (PECCS) Problem

[134] In case S or F is empty, its corresponding PEXPR can be an arbitrary contradiction.

Given:
A natural number n;

P is a consistent, DNF, bar-free PEXPR
whose constants are drawn from set CON=$\{c_1,...,c_n\}$;

N is a consistent, DNF, paren-free PEXPR
whose constants are also drawn from CON.

To Decide:
there is a $\Pi \epsilon$ Part(n) such that
not $\Pi \models$P and not $\Pi \models$N.

There is no reason to expect a general, tractable solution to this simply stated problem, for:

Theorem NP5:

The PEXPR Elementary Class Complement Search Problem is NP-complete.

Proof:

First, the PECCS problem is easily seen to be in NP. Let M be a non-deterministic machine that guesses a partition Π in Part(n) and that verifies whether Π fails to satisfy P and fails to satisfy N. This verification amounts to finding pairs of constants in Π and computing propositional truth values on the basis of a given assignment, both of which tasks are easily seen to require time only polynomial in the size of the input formula.

It remains to show that the PECCS problem is NP-hard. This is accomplished by showing that a special case of the clausal satisfiability problem that is known to be NP-complete reduces to PECCS in polynomial time.

The 3-Satisfiability of Sign-Homogeneous Clauses (3-HSAT) Problem

Given: A set T of clauses on the propositional variables $\{q_1,...,q_k\}$ such that each clause contains no more than 3 disjuncts and for each clause C, either all disjuncts occurring in C are negated or all disjuncts occurring in C are non-negated.

To Decide: Is there a satisfying truth-assignment for T?

The 3-HSAT problem is shown to be NP-complete in [Gold78].

Now we need to define a polynomial reduction function f such that f(T)=<P,N,n>, where <P,N,n> is an instance of the PECCS problem whose

answer is 'yes' just in case the answer to T is 'yes'. Let T be an arbitrary instance of 3-HSAT, where T=(P'U N'), P' is a set of clauses whose propositional variables all occur non-negated, and N' is a set of clauses whose propositional variables all occur negated. Let Occ(T) be the set of all propositional variables occurring in T. For each $q_i \epsilon$ Occ(T), define $g(q_i)$='(2i-1,2i)' and $g(-q_i)$='(2i-1|2i' If F is a propositional formula on the vocabulary of T, then let g(F) be the result of applying g to each propositional variable occurring in F. For any finite set of clauses T, Define NEG(T) to be the result of disjoining the conjunctions that result from negating each clause and driving the negation in using the DeMorgan and double negation rules. Finally, define

$$f(P'U\ N') = \langle g[NEG(N')],\ g[NEG(P')],\ 2|Occ(P'U\ N')|\rangle = \langle P,N,n\rangle$$

If T=P'U N' is a finite set of sign-homogeneous 3-clauses, then g[NEG(P')], g[NEG(N')] are each DNF PEXPRs such that the former is paren-free and the latter is slash-free. So f(T) is indeed an instance of the PECCS problem.

f is evidently easy to compute. NEG can be computed in time constant in the size of T, as can g. And to compute n requires only counting propositional variables and multiplying by 2, which can be done in time polynomial in the size of T as well.

Now, let T= (P' U N') be an arbitrary instance of 3-HSAT, and let <N,P,k> = f(T). All that remains is to show that T is satisfiable if and only if there is a partition in Part(k) that does not p-satisfy either N or P.

---> Suppose T = (P' U N') is satisfiable. Then there is a satisfying truth assignment σ:Occ(T)--->{t,f} for T. Define $\Pi[\sigma] \epsilon$ Part(m) such that for each i and j>i from 1 to 2m, i,j are in the same cell of $\Pi[\sigma]$ if and only if there is a k from 1 to m such that j=2k and i=2k-1 and σq_k=t. We show that $\Pi[\sigma]$ does not p-satisfy either P or N. Consider P first. Recall that P=g(NEG(N')). Recall also that $\sigma|$=N'. Since N' is a finite set of disjunctions whose disjuncts are negated atoms, for each disjunction D in N' there is a sentential variable q_i that occurs in D such that σ does not satisfy q_i. So for each disjunction D' occurring in NEG(N') there is a q_i that occurs in D' such that σ does not satisfy q_i. Note that q_i does not occur negated in NEG(N'). Hence, $g(q_i)$ = '(2i-1,2i)', by the definition of g. So by the definition of $\Pi[\sigma]$, 2i-1, 2i are in distinct cells of $\Pi[\sigma]$. So $\Pi[\sigma]$ does not p-satisfy '(2i-1, 2i)'. Hence, for each disjunct occurring in P there is an atomic PEXPR (n,m) such that (n,m) is not p-satisfied by $\Pi[\sigma]$. Hence, P is not p-satisfied by $\Pi[\sigma]$. I omit the dual argument for the case of N.

<--- Suppose there is a partition Π in Part(k) that fails to p-satisfy either P or N. By the definition of f, recall that k=2m, where m=|Occ(T)|. Now define $\sigma[\Pi]$ so that for all i between 1 and m, $\sigma|=q_i$ if and only if 2i, 2i-1 are in the same cell of Π. We show that $\sigma[\Pi]$ satisfies both P' and N'. Take the case of N'. Since Π does not p-satisfy P, we know that for each disjunct D occurring in P, there is an i from 1 to 2|Occ(T)| such that the atomic PEXPR '(2i-1,2i)' is not p-satisfied by Π. Hence, 2i-1, 2i are not in the same cell of Π. Hence, $\sigma[\Pi]$ does not satisfy q_i, by definition. Since P=g(NEG(N')), σ does not satisfy NEG(N'). So σ does satisfy N'. I omit the dual argument for the case of P'. Q.E.D.

7.4.2. Avoiding Upsets Only

The proof that the PECCS problem is NP-hard relies on a reduction of a satisfiability problem to the PECCS problem. So perhaps the difficulty of the problem resides in the fact that the PEXPRs involved can express negation, and hence can be mutually inconsistent. This line of thought leads naturally to the suggestion that a partition lattice be searched by avoiding *only* downsets or upsets, but never both at once. In this way, negation could be avoided, and perhaps a polynomial algorithm could be found.

Consider the strategy of avoiding only upsets. First, we assume that some points in the lattice have somehow been tested and have passed this test. Call the set of all such partitions S. No partition in Up(S) need be considered, for its corresponding clause is entailed by some tested, successful clause, and hence its addition to the current successes would be logically gratuitous.

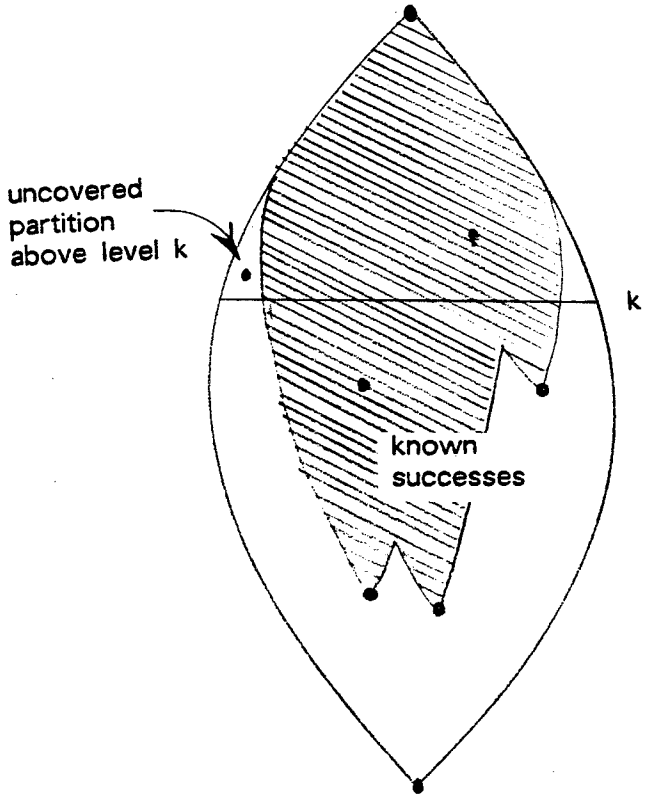


Figure 7-14: Avoiding Upsets

So our task is to generate some partition in the lattice that is not in the upset of S. A moment's reflection reveals that as the cardinality of a partition goes down, its

chance of being covered goes up. That is, an arbitrary upset in Part(n) always covers a higher percentage of a shallow level than it covers of a deep level. So if we are interested in testing as few lattice points as possible, a sensible heuristic is to start testing deep in the lattice, so that any empirical success will cover as large a portion of the uninvestigated part of the lattice as possible. But once one is finished testing all uncovered partitions at a level deep in the lattice, one must find some point at a higher level in the lattice that is not yet covered by a success already discovered lower in the lattice. That is, given an arbitrary cloud of points lower in the lattice along with the specification of a level, the job is to generate some point at a higher level in the lattice that is not in the upset of the given cloud just in case there is one. But if one could solve this generation problem quickly, the generator would provide a fast solution to the following decision problem:

The Partition Lattice Upset Complement Search (UCS) Problem

Given:
a subset S of Part(n);
a natural number k between 1 and n.

To Decide:
there is $\Pi \in$ Part(n) such that $|\Pi| \leq k$ & not $\Pi \in$ Up(S).

But sadly enough, this plausible approach is met with a stronger negative result than that which faced the previous approach--- a fact which is not so obvious on the face of it.

Theorem NP6:

The partition lattice upset complement search (UCS) problem is NP-complete.

Proof: First UCS is in NP, for consider a nondeterministic machine that guesses a partition Π in Canon(n), and then checks whether $|\Pi| \leq k$ and whether Π is in the complement of Up(S). The cardinality check requires but one pass through Π. And checking whether no element of S is a refinement of Π can be performed in time polynomial in the size of S and Π. For let Π,Π' be elements of Part(n). Π is a refinement of Π' just in case there is no pair i,j of natural numbers between 1 and n such that i,j are in the same cell of Π' but are not in the same cell of Π. Hence, we need only to generate each such pair i,j (there are n choose 2 of them) and to locate the cells of Π and Π' of which they are respective elements. Locating i and j in Π and Π' require at worst two passes through each partition, or 4n steps. But the function 4n(n choose 2) is just $4n^3-n$, which is polynomial in the sizes of the input partitions.

The non-trivial part is to show that UCS is NP-hard. This is accomplished by showing that *The Minimal Set Cover (MSC) Problem* reduces to the UCS problem in polynomial time.

The Minimal Set Cover (MSC) Problem

Given:
a finite set R;
a subset Γ of the power set of R;
a natural number c.

To Decide:
there is a subset Γ' of Γ such that $|\Gamma'| \leq c$ and $U\Gamma'=R$

The MSC problem is known to be NP-complete [Garey79].

The MSC problem can be transformed polynomially into a form MSC' that illuminates the analogy between set cover minimization and the UCS problem. Let $\langle R,\Gamma,c\rangle$ be an arbitrary instance of this problem, where $\Gamma=\{G_1,\ldots,G_m\}$, so that $|\Gamma|=m$ and so that $|R|=p$. It reduces tedium in what follows to establish the convention that i,j range from 1 to m and i',j' range from 1 to 2m. Now define,

for each $r \in R$,
$S(r) = \{i: r \in G_i \in \Gamma\}$

$g(R)=R'=\{S(r): r \in R\}$

$g'(\langle R,\Gamma,c\rangle) = \langle g(R),c\rangle$

Notice that since R' encodes both the cardinality of R and the structure of Γ, there is no point retaining Γ. So an instance of MSC' is just a pair $\langle R',c\rangle$, where R' is n arbitrary set of sets of numbers from 1 to m, for any given m. The corresponding question for MSC' is whether

(*) there exists some subset S' of numbers from 1 to m such that $|S'| \leq c$ and for each $S \in R'$ there is an $i \in S'$ such that $i \in S$.

Observe that g' can be computed in polynomial time. So all we need to know is that

Lemma: there is a subset Γ' of Γ such that $|\Gamma'| \leq c$ and $U\Gamma'=R$ if and only if there exists some subset S' of natural numbers from 1 to m such that $|S'| \leq c$ and for each $S \in R'$ there is an $i \in S'$ such that $i \in S$. That is, $\langle R,\Gamma,c\rangle \in$ MSC if and only if $f(\langle R,\Gamma,c\rangle)$ $(=\langle R',c\rangle) \in$ MSC'.

---> Let $\langle R,\Gamma,c\rangle$ be an instance of MSC. Suppose there is a subset Γ' of Γ such that $|\Gamma'| \leq c$ and $U\Gamma'=R$. Define $S[\Gamma']$ so that for all i from 1 to m, $i \in S[\Gamma']$ if and only if $G_i \in \Gamma'$. So $|S[\Gamma']| = |\Gamma'| \leq c$. Suppose $S \in R'$. Then by the definition of g, there is an $a \in R$ such that $S=\{i: a \in G_i\}$. Since $U\Gamma'=R$, there is a $G_i \in \Gamma'$ such that $a \in G_i$. So by the definition of $S[\Gamma']$, $i \in S[\Gamma']$.

<--- Let $\langle R',c\rangle$ $(=f(\langle R,\Gamma,c\rangle))$ be an instance of MSC'. Suppose that there is a subset S' of natural numbers from 1 to m such that $|S'| \leq c$ and for each $S \in R'$ there is an $i \in S'$ such that $i \in S$. Define $\Gamma[S']$ so that $G_i \in \Gamma[S']$ if and only if $i \in S'$. So $|\Gamma[S']| = |\Gamma'| \leq c$ Let a be an arbitrary element of R. Then by the definition of g, $S(a) \in R'$. Then by hypothesis, there is a k such that $k \in S(a)$ and $k \in S'$. So by the definition of $\Gamma[S']$, $G_k \in \Gamma[S']$. So $U\Gamma[S']=R$, which completes the proof of the lemma.

Let <R,c> be an arbitrary instance of MSC', where |U[R]|=m. We can think of each $S \epsilon R$ as uniquely coded by a binary m-tuple σ such that for each i from 1 to m, $\sigma_i=1$ if $i \epsilon S$, and $\sigma_i=0$ otherwise. Rephrasing the question for sets coded as Boolean sequences yields

> There is an m-tuple σ' such that there are as few as c distinct occurrences of '1' in σ and for each $\sigma \epsilon R'$ there is an i from 1 to m such that $\sigma'_i=\sigma_i=1$.

For convenience, let $|\sigma|$ denote the number of occurrences of '1' in σ and let $\sigma U \sigma'$ denote the m-tuple σ'' such that σ'' is the characteristic function of the union of the sets S,S', where σ is the characteristic function of S and σ' is the characteristic function of S'.

Now we define a reduction function f(<R,c>) = <S,k>, where <R,c> is an instance of MSC' (with sets coded as Boolean sequences) such that MAX(U(R))=m and <S,k> is an instance of UCS.

Notice that the partitions of a set correspond uniquely to the equivalence relations on that set. Hence, I shall employ the name of a partition as an equivalence relation. That is, $i \Pi j$ just in case there is a cell $c \epsilon \Pi$ such that $i,j \epsilon c$.

The principal intuition behind the reduction f is that "covering" each element of a finite set in the MSC problem is analogous to "avoiding refinement" into any element of a set of partitions in the UCS problem. Hence, each element of the set R should correspond to some partition in S. We have already assisted the analogy by transforming the elements of this set into Boolean m-tuples. So the next step is to encode an arbitrary Boolean m-tuple σ with a partition Π in Part(2m).

The trick is to represent a '0' in position i of σ as the inequivalence of 2i and 2i-1 according to Π, and to represent a '1' in position i of σ as the equivalence of 2i and 2i-1 according to Π. So we can think of each pair 2i, 2i-1 from 1 to 2m as encoding the ith "bit" of σ. Finally, we want Π to be the most refined partition that satisfies these constraints.

Now, for each $\sigma \epsilon R$, define $\Pi(\sigma)$ so that
for each i', $i'<j'$, $i' \Pi(\sigma) j'$ just if $\sigma_{j'/2}=1$ and $i'=j'-1$.

Codes(R)=$\{\Pi[\sigma]:\sigma \epsilon R\}$

Codes(R) is therefore exactly the set of partition-codes for the Boolean sequences in R. Notice, in particular, that any pairs not of the form 2i, 2i-1 are inequivalent according to each element of Codes(R). This implies that each element of Codes(R) is a refinement of the partition

$\Pi_0(m) = \{\{1,2\},\{3,4\},\{5,6\},\ldots,\{2m-1,2m\}\}$.

Hence, for each $\Pi \epsilon$ Codes(R), $|\Pi| \geq m$.

Next define:

for each $1 \leq x < y \leq m$,
for each i,j>i, $i\Pi(x,y)j$ just if x=i and y=j

Axioms(m)={Π(i,j):not [j is divisible by 2 and i=j−1]}.

Finally, define
f(<R,c>) =
<(Axioms(2|U[R]|) U Codes(R)), (|U[R]|+c)> =
<S,k>

This completes the definition of f.

First, f can be computed within time polynomial in the size of the input string encoding of <R,c>. By construction, |U[R]| = MAX(U[R]). The MAX can be found by starting with X=0, and by setting X to the next element of U[R] if this element is greater than X thereafter. This process is clearly linear in the size of R. And addition over natural numbers can be performed in polynomial time. Moreover, Codes(R) can be generated in polynomial time by constructing Π_0, and by splitting the cell {2i−1, 2i} just in case σ_i=0. Finally, |Axioms(2m)| is bounded above by the quantity (2m choose 2), or $4m^2+2m$. All unordered pairs drawn from a given set can be generated in polynomial time, and for each such pair {x,y}, Π(x,y) can be generated in time polynomial in R by placing x,y in the same cell, and by separating every other i and j.

Two auxiliary definitions are required in the course of the proof. Intuitively, one converts a Boolean sequence that covers R into a partition that avoids refinement into f(R), and the other converts a partition that avoids refinement into S into a sequence that covers $f^{-1}(S)$.

For each i,

$$\sigma[\Pi]_i = \begin{cases} 1 \text{ if not } (2i)\Pi(2i-1) \\ 0 \text{ otherwise} \end{cases}$$

For each i',j'>i', $i'\Pi[\sigma]j'$ just in case
there is an i such that i'=2i−1, j'=2i, and $\sigma_{j'/2}$=0

Lemma:

Let <S,k>=f(<R,c>). Then there is a $\Pi \in$ Part(n) such that $|\Pi| \leq k$ & not $\Pi \in$ Up(S) just in case there is a Boolean m-tuple σ' such that there are no more than c distinct occurrences of '1' in σ' and for each $\sigma \in R'$ there is an i such that $\sigma'_i=\sigma_i=1$.

---> Assume that there is a $\Pi \in$ Part(n) such that $|\Pi| \leq k$ & not $\Pi \in$ Up(S). Call this partition Π', and let m=|U[R]|. Since Π' is not in Up(S), we have that (a) no element of Axioms(2m) is a refinement of Π' and (b) no element of Codes(R) is a refinement of Π'.

(a) implies, by the definition of refinement, that for each $\Pi \epsilon$ Axioms(2m) there are i',j' such that i'Πj' but not i'Π'j'. So by the definition of Axioms(2m), we have that for each x,y>x between 1 and 2m such that either y is not divisible evenly by 2 or x is not y−1, there are i',j'>i' such that i'Π(x,y)j' but not i'Π'j'. But by the definition of Π(x,y), i'Π(x,y)j' just in case i'=x and j'=y. Hence, for each i',j'>i' such that j' is not divisible evenly by 2 or i' is not j'−1, it is not the case that i'Π'j'. So for each i',j'>i', i'Π'j' only if j' is divisible evenly by 2 and i'=j'−1. So Π' is a refinement of Π_0(m) = {{1,2},{3,4},...,{2m−1,2m}}. Notice that $|\Pi_0(m)|$=m. Any refinement of Π_0(m) is the result of splitting some of the cells of Π_0(m). So there is some subset Q of {1,2,...,m} such that for each iϵQ, not 2i−1Π'2i. Notice that $|\Pi|=|Q|+|\Pi_0(m)|=|Q|+m$, for each "splitting" of a cell in Π_0(m) raises the cardinality of Π' over that of Π_0(m) by one. But by hypothesis, $|\Pi'| \leq k=m+c$. Hence, (a') $|Q| \leq c$.

(b) implies, by the definitions of Codes(R) and of refinement, that for each $\sigma \epsilon$R there are i',j'>i' such that i'$\Pi(\sigma)$j' but not i'Π'j'. But since i'Π'j' only if there is an i such that j'=2i and i'=2i−1, and since each such i is an element of Q, we have that for each $\sigma \epsilon$R there is an iϵQ such that 2i−1$\Pi(\sigma)$2i but not 2i−1Π'2i. This implies, by the definitions of $\sigma[\Pi']$ and $\Pi(\sigma)$ that for each $\sigma \epsilon$R, there is an iϵQ such that σ_i=1 and $\sigma[\Pi']$=1. Since (by (a')) $|Q| \leq c$, there is a σ' (namely $\sigma(\Pi')$) such that there are no more than c distinct occurrences of 1 in σ', and for each $\sigma \epsilon$R there is an i such that $\sigma'=\sigma=1$.

<--- Assume there is a Boolean m−tuple σ such that '1' occurs no more than c times in σ and for each $\sigma \epsilon$R there is an i such that $\sigma'_i=\sigma_i=1$. Call this m−tuple σ'. Let Q be the set of all i such that $\sigma'_i=1$. By the definitions of $\Pi(\sigma)$ and $\Pi[\sigma']$, it follows that (*) for each $\sigma \epsilon$R, there is an iϵQ such that 2i$\Pi(\sigma)$2i−1 but not 2i$\Pi[\sigma']$2i−1. Moreover, the definition of $\Pi[\sigma']$ guarantees that $\Pi[\sigma']$ is a refinement of Π_0(m). Hence, (1) no element of Axioms(m) is a refinement of $\Pi(\sigma')$. Moreover, the definition of $\Pi[\sigma']$ implies that (2) $|\Pi[\sigma']|=|\Pi_0(m)|+|Q|=m+|Q| \leq m+c$.

And (*) implies, along with the definition of Codes(R) that $\Pi[\sigma']$ is a refinement of no element of Codes(R). So (3) $\Pi[\sigma']$ is not in Up(Codes(R)). So by (1), (2), and (3), we have that there is a Π' (namely $\Pi[\sigma']$) such that $|\Pi| \leq m+c=k$ and no element of Axioms(m)UCodes(R) (=S) is a refinement of Π. Q.E.D.

The chain of polynomial reductions defined in this proof is complicated, so a simple, intuitive example of its operation is in order. Let our original instance <R,Γ,c> of MSC (the "Minimal Set Cover" problem) be

$R=\{a,b,c,d\}$

$\Gamma=\{G_1, G_2, G_3\}$,

$G_1=\{a,b\}$
$G_2=\{c,d\}$
$G_3=\{a,c\}$

$c=2$

The object is to find whether the union of some c=2 or fewer elements of Γ is identical to R. By inspection, the answer is 'yes', for $\{a,b\} \cup \{c,d\} = \{a,b,c,d\}$.

The corresponding instance $\langle R',c\rangle$ of MSC looks like this:

$R'= \{S(a), S(b), S(c), S(d)\}$

$S(a)=\{1,3\}$
$S(b)=\{1\}$
$S(c)=\{2,3\}$
$S(d)=\{2\}$

$c=3$

The problem is to decide whether there is subset S' of of the set $\{1,2,3\}$ such that the cardinality of S' is no greater than c and for each $S \in R'$ there is an $i \in S'$ such that i is also an element of S.

Next, we encode the S(r) as boolean sequences representing their characteristic functions over the universe $\{1,2,3\}$.

$R' = \{S(a), S(b), S(c), S(d)\}$

$S(a) = \langle 1,0,1\rangle$
$S(b) = \langle 1,0,0\rangle$
$S(c) = \langle 0,1,1\rangle$
$S(d) = \langle 1,1,0\rangle$

$c=2$

The corresponding question is whether there are c=2 or fewer distinct positions between 1 and 3 such that for each Boolean triple in R', a '1' occurs in one of these positions. Again, the answer in this case is 'yes', for if we choose positions 1 and 2, then a and b each have a '1' in position 1, and c and d each have a '1' in position 2. Evidently, this is just a notational variant of the original problem.

Finally, R' transforms into the following set S=(Codes U Axioms) of partitions and c transforms into k=m+c:

Codes: 1 2 3

a: {{12} {3}{4} {56}}
b: {{12} {3}{4} {5}{6}}
c: {{1}{2} {34} {56}}
d: {{12} {34} {5}{6}}

Axioms:

{{13} {2}{4}{5}{6}}
{{14} {2}{3}{5}{6}}
{{15} {2}{3}{4}{6}}
{{16} {2}{3}{4}{5}}
{{23} {1}{4}{5}{6}}
{{24} {1}{3}{5}{6}}
{{25} {1}{3}{4}{6}}
{{26} {1}{3}{4}{5}}
{{35} {1}{2}{4}{6}}
{{36} {1}{2}{4}{5}}
{{45} {1}{2}{3}{6}}
{{46} {1}{2}{3}{5}}

k=3+2

Notice that Codes does in fact code R' so that the code of b has 2i−1 and 2i in the same cell just in case b has a '1' in position i. The question now is the one in which we are interested, namely, is there a partition $\Pi \epsilon$ Part(2m) such that $|\Pi| \leq k=3+2=5$ and Π is not in Up(S). Notice that the answer is again 'yes' (as it had better be!), and the desired partition is {{1}{2}{3}{4}{56}.

This result is much stronger than the previous one because the input consists of an arbitrary subset of the lattice rather than of a PEXPR description. Hence the theorem addresses exactly the "bottom up" lattice search we were contemplating. Interestingly, this stronger result corresponds to the observation that, unlike the case of paren-free PEXPRs, the cardinality of the maximal generating set of the elementary class of a bar and negation free PEXPR is not exponential in the length of the PEXPR. In other words, there is no "masking" of the difficulty of this problem by an exponential expansion of the generating set of the elementary class of a bar-free PEXPR as there was in the previous problem which involved paren-free PEXPRs. The present result therefore strengthens the suspicion that avoiding upsets and downsets of subsets of a partition lattice (regardless of level) is intractable as well. But I haven't settled the question.

7.4.3. Avoiding Downsets Only

Although we have seen that avoiding upsets and downsets is likely to be difficult, and that avoiding upsets alone is more likely to be difficult, perhaps avoiding downsets is easy. After all, we know that the structure of partition lattices is not symmetric from top to bottom.

On this proposal, one has at some stage a set F of partitions corresponding to clauses tested and found to be empirically unsuitable. It makes some intuitive sense to begin such a search high in the lattice, for any failure found at a high level covers more partitions than any failed point at any other level. But once all the points at a level have been tested, one must generate a partition at a lower level that is not covered by any failure found at a higher level, and it is possible that all such partitions are covered. So the task, then, is to find some untested clause Π at or below a given level, that is not in the downset generated by F. Since there is an exponential difference in size between maximal generating sets and paren-free PEXPRs representing the same downset, we should (in good faith) employ the "unpadded" PEXPR notation.

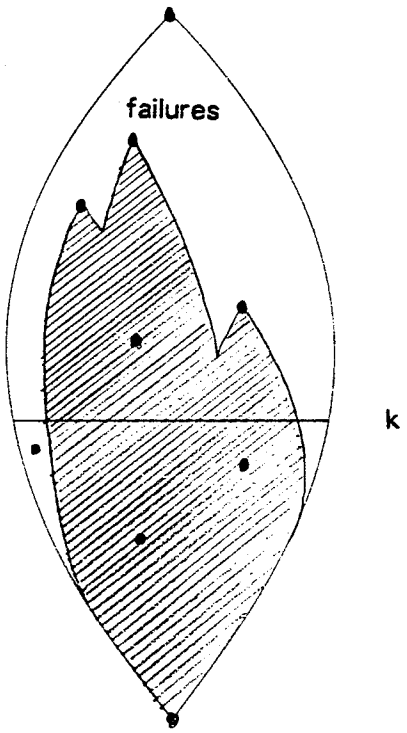


Figure 7-15: Avoiding Downsets Only

That is, we want to generate a partition below a given level that does not satisfy a given, paren-free PEXPR.

If we had a fast generator of this sort, we would have *eo ipso* a fast decision procedure for the following decision problem:

The Paren-Free PEXPR Elementary Class Complement Search (SFPCS) Problem

Given:
A Paren-Free DNF PEXPR P;

Natural numbers k,n>k.

To Decide:

there is a $\Pi \epsilon$ Part(n) such that $|\Pi| \geq k$ and
Π does not satisfy P.

The reader may well have anticipated as much, but it is not entirely obvious that:

Theorem NP7:

The paren-free PEXPR elementary class complement search (SFPCS) problem is NP-complete.

Proof: That SFPCS is in NP is evident from the fact that counting the cells of a guessed partition can be done in time linear in the partition size, and testing whether $\Pi|=P$ is polynomial, by the argument in the proof that is in NP.

It remains to show that SFPCS is NP-hard. We show that there is a polynomial reduction of MSC' to SFPCS, in a manner quite analogous to that of the proof that UCS is NP-hard. Let <R,c> be an instance of MSC', as described in the proof that UCS is NP-hard and let m=|R|.

As before, let i',j' range from 1 to 2m and let i,j range from 1 to m. Now define the paren-free PEXPR $P(\sigma)$ for each Boolean m-tuple σ, as follows:

$D(\sigma)[0]$=the empty string.

$$D(\sigma)[i+1] = \begin{cases} D(\sigma)[i]*'\&\ 2i+1|2i+2' & \text{if } \sigma_i=1 \\ D(\sigma)[i] & \text{otherwise} \end{cases}$$

$D(\sigma)=D(\sigma)[m]$.

Next, let

$Code(R)= Disjunction_{\sigma \epsilon R} D(\sigma)$

Finally, we define the reduction function

$f(<R,c>)=<Code(R),\ 2|U[R]|-c>=<P,k>$.

Code(R) can be computed in time polynomial in the size of R, for the definition of $Q(\sigma)$ is essentially a polynomial algorithm for generating $Q(\sigma)$ from σ. And $2|U[R]|-c$ is evidently easy to compute. So f is computable in time polynomial in the size of <R,c>.

Let <R,c> be an instance of MSC' (where elements of R are viewed as Boolean sequences. Let $|U[R]|=m$, and let <P,k> = <Code(R),m−c> = f(<R,c>). It remains to show that there is a $\Pi \epsilon$ Part(m) such that $|\Pi| \geq k$ and Π does not satisfy P just in case there is a σ' such that $|\sigma| \leq c$ and for each $\sigma \epsilon R$ there is an i such that $\sigma'_i=\sigma_i=1$.

---> Assume that there is a $\Pi \epsilon$ Part(m) such that $|\Pi| \geq k$ and Π does not satisfy P. Call the guaranteed partition Π'. Hence, for each disjunct D occurring in P, there is an i such that $2i-1|2i$ occurs in D but $2i-1\Pi'2i$. Hence, (a) there is some subset Q of {1,2,...,m} such that for each disjunct D occurring in P there is an $i \epsilon Q$ such that $2i-1|2i$ occurs in D but $2i-1\Pi'2i$. Since the result of identifying $2i-1$ with $2i$ in Π' has cardinality one less than the cardinality of Π', we have that $|\Pi'|=2m-|Q|$. But $|\Pi| \geq k=2m-c$, by hypothesis. Hence, $|Q| \leq c$. By (a) and the definition of Code(R) we have that for each $\sigma \epsilon R$ there is an $i \epsilon Q$ such that $2i-1|2i$ occurs in $D(\sigma)$ and $2i-1\Pi'2i$. So by the definition of $D(\sigma)$ we have that for all $\sigma \epsilon R$ there is an $i \epsilon Q$ such that $\sigma_i=1$ and $2i-1\Pi'2i$. Now, define $\sigma[\Pi']$ so that for each i, $\sigma[\Pi']_i=1$ if $(2i-1)\Pi'2i$; and =0 otherwise. Hence, for all $\sigma \epsilon R$, there is an $i \epsilon Q$ such that $\sigma_i=\sigma[\Pi']_i=1$. But since $|Q| \leq c$, we have that there is a σ' (namely $\sigma[\Pi']$) such that $|\sigma| \leq c$ and for each $\sigma \epsilon R$ there is an i such that $\sigma'_i=\sigma_i=1$.

<--- Suppose there is a Boolean m-tuple σ such that $|\sigma| \leq c$ and for each $\sigma \epsilon R$ there is an i such that $\sigma'_i=\sigma_i=1$. Call it σ'. Let Q be the set of all i such that $\sigma_i=1$. So $|Q| \leq c$, by hypothesis. Hence, for each $\sigma \epsilon R$ there is an $i \epsilon Q$ such that $\sigma'_i=1$ and $\sigma_i=1$. By the definition of $D(\sigma)$, for each $\sigma \epsilon R$, there is an $i \epsilon Q$ such that $\sigma'_i=1$ and $2i-1|2i$ occurs in $D(\sigma)$ and $\sigma'_i=1$. Define $\Pi[\sigma]$ so that for all $i',j'>i'$, $i'\Pi[\sigma]j'$ just in case there is an i such that $j'=2i$, $i'=2i-1$, and $\sigma_i=1$. Then we have that for all $\sigma \epsilon R$, there is an $i \epsilon Q$ such that $2i-1|2i$ occurs in $D(\sigma)$ and $2i-1\Pi[\sigma']2i$. Then $|\Pi|=2m-|Q|$, for $|\Pi_1|=2m$ and two distinct cells in Π_1 are merged for each $i \epsilon Q$. But since $|Q| \leq c$, we have that $|\Pi| \geq 2m-c$. So by the definition of Code(R), there is a Π' (namely $\Pi[\sigma']$) such that $|\Pi| \geq 2m-c$ (=k) and for each disjunct D occurring in Code(R) (=S) Π' does not satisfy D. Q.E.D.

7.4.4. The Silver Lining

Since the days of logical positivism, the "sophisticated" money has been bet against positive results in the logic of discovery. Such sophisticates may take the above results as superfluous confirmation of what they already knew. But these results are hardly devastating to the general project of finding useful hypothesis generators. The following discussion reviews some of their limitations.

To begin with, NP5 and NP7 apply only to problems in which the given upward and downward closed sets are coded as PEXPRs. And my proofs of these results cannot be extended to the case in which the given upward and downward closed sets are presented in terms of their maximal generating partitions. This is because the maximal generating set for an upset encoded by a bar-free PEXPR can be exponential in the size of the PEXPR. Since the "output" of the transformation function is too big, the function cannot be computed in polynomial time.

Nothing just said implies that some *other* proof of NP-completeness could not be found for the upset complement search problem when the upset is encoded by its maximal generating set. But for all that has been shown, the entire difficulty of the PEXPR version of the problem may just be a result of the fact that the maximal generating set for the elementary class of a PEXPR can be exponentially larger than the PEXPR itself. Perhaps the maximal generating set already encodes (in some mysterious manner) all the difficult search that must be performed in the PECCS problem. Or perhaps not. Aside from the interest of the question to this thesis, its resolution would be of independent, mathematical interest in its own right.

Second, intractability and uncomputability theorems are not very robust in supporting broad, intuitive, philosophical claims. In particular, the theory of NP-completeness is but a species of worst-case complexity theory. So there may be infinitely many instance sizes for which each instance of this size is "easy", and it is also possible that most instances of every size are easy. The upper bound skims like a light canopy over the worst problem instances of each size without telling us anything about the deeper topography of the problem.

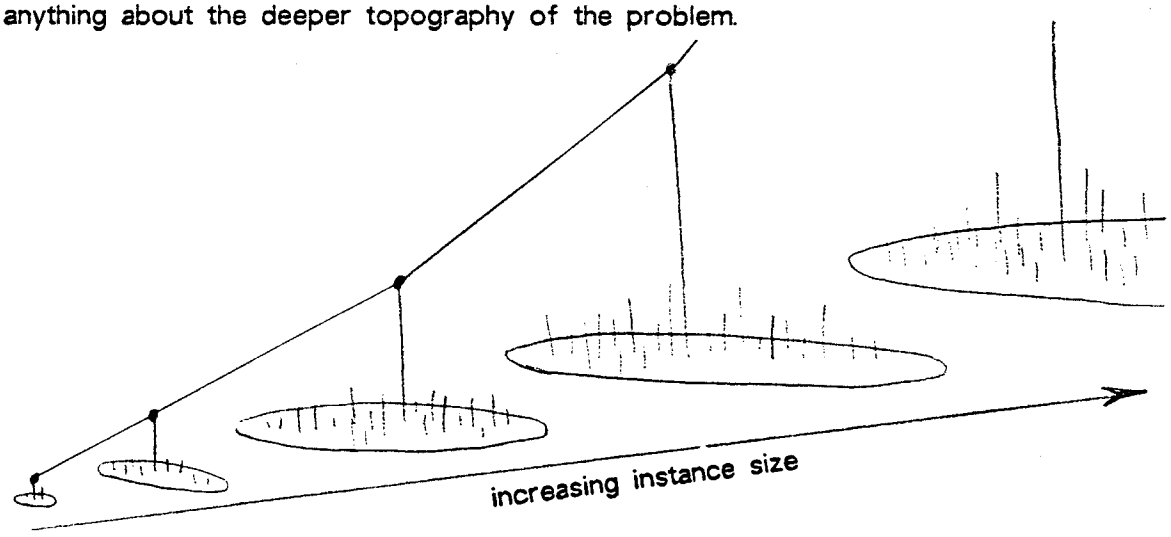


Figure 7-16: Worst-Case Complexity

An obvious step, then, is to isolate difficult and easy restrictions of the original problem.[135] When push comes to shove, we may decide that a restricted problem is all we really wanted to solve anyway. One way to find out whether "most" of a problem is easy is to study the problem's expected complexity. Imagine a problem P with a restriction P' such for some polynomial function p, all but p(n) instances of size n of P are instances of size n of P'. Imagine further that a program that solves P can solve P' in polynomial time. Then for each instance size, the easy instances "wash out" the hard instances in the average resource consumption of the program over all instances of this size. Hence, the expected complexity of P is polynomial even though its worst-case complexity is not. So a proof that an NP-complete problem has polynomial expected complexity tells us that some "slight" (polynomial) restriction of the problem yields a tractable problem.

In addition to restricting the problem to be solved, altering the criteria for having solved it can elude the intuitive impact of an intractability result. For example, it might be easy to solve a problem correctly ninety percent of the time but costly to solve it correctly in every case. For practical purposes, being right ninety percent of the time may be more than sufficient.

But these recipes for evading the *intuitive* impact of an NP-completeness theorem should not be taken to undermine its *formal* significance. If one really wants to find a search-free way to solve a problem, then an NP-completeness result is devastating to the project. Such results provide a mathematical explanation of sorts for one's repeated failure to find a search-free method. The explanation is this. Nobody else who has sought such a solution for an NP-complete problem has ever met with success. But if you had found one for your problem, you would *ipso facto* have found a search-free solution to everybody else's NP-complete problems as well. So unless you have a cute approach to your problem that nobody else has applied to their own problems, there is little point even trying to find a search-free solution to yours by means of the usual repertoire of procedural tricks.

But the formal generality of these results is more of a help than a hindrance to the logic of discovery. They tell us where not to waste valuable time trying to find search-free solutions that will be very difficult to find even if they exist. Instead, NP-completeness theorems focus our attention on smaller gains in restricted problems, where success is more likely to be encountered. It may just turn out that

---

[135] P' is a restriction of P just in case the characteristic function of P' is a restriction of the characteristic function of P to a smaller instance universe. I do not mean that P' is a subset of P over the same universe, for this would be a different problem rather than a restriction of the same problem [Garey79], p. 63.

inductive intelligence lies hidden in one of the dark, mathematical corners the theorem leaves unexplored. An intractability proof can suggest where these corners are by isolating a particular aspect of the problem that is difficult. So an NP-completeness theorem in a problem domain is not an invitation for brooding and dismay. Rather, it serves to focus inquiry where easy progress can still be expected.

7.5. Test-Oriented Procedures for Partition Lattice Search

It is time to examine some procedures that seem, in many cases, to ignore hypotheses *a posteriori* in light of the evidence without compromising the inductive strategy of CONSIST. In this section, I present some "generate and test" approaches in which suitability testing is treated as an isolated subroutine that does not interact with clausal generation in any way except to say for any given clause whether that clause is suitable or not on the given evidence. An immediate consequence of this orientation is that the test procedure must solve an NP-complete problem (Theorem NP2) and so any known implementation will be hopelessly slow in some cases due to hypothesis test alone.[136] But rather than being a signal to give up, the intractability of hypothesis test is a keen motivation to try to minimize the number of tests performed and the number of hypotheses considered.

An obvious generate-and-test approach to computing the CONSIST strategy is to choose an arbitrary element of Part(n), and to test it. If the partition passes the test, place it in S. Otherwise, place the partition in F. Next, generate some other element of Part(n) that is in neither Down(F) nor Up(S). Test this partition, assign it to S or to F, and continue, until no partition can be found that is neither in Down(F) nor in Up(S). More precisely,

```
SEARCH(n)
begin
        set H:=S:={};
        while there is an h∈Part(n)-(Up(S) U Down(F)) do
        begin
                choose some h∈Part(n)-(Up(S) U Down(F));
                test h;
                if h passes test then set S:=(S U {h})
                else set F:=(F U {h})
        end
end.
```

Notice that the command "choose" is vague. In fact, how the choice is defined can

---

[136] Recall that in the inductive strategy CONSIST, hypothesis size grows with evidence size. Since the size of a consistency problem is the size of the given hypothesis and the given evidence, the overall procedure just described is likely intractable in the size of the input evidence.

have a significant impact on the complexity of the overall procedure. For example, suppose that the choice is a function of n, S, and F, and is defined to be one of the most refined partitions in Part(n)-(Up(S)*U Down(F)). In a sense, this is exactly the criterion embodied in Shapiro's "refinement operators". In the worst case for this procedure, every clause in the lattice is false, so every clause in the lattice is tested--- even though no clause that entails a failure is tested and no clause entailed by a success is tested. There is an obvious, dual disaster for the approach that searches the lattice from bottom to top.

Computer science has developed a technique for coping with these simple, worst-case search disasters. For example, consider an arbitrary, totally ordered finite set $\langle Q, \leq \rangle$ and given, "black box" decision procedure for property R on Q such that R is closed downward in Q according to $\leq$. Think of Q as a very simple, finite lattice of hypotheses, and of R as a test procedure defined on Q. The problem is to find the greatest element of R in Q according to $<$.

A naive method is to start at one end-point of Q and to traverse Q in descending order until some element of Q passes the test. Notice that the method just outlined does have the property of never testing a hypothesis lower than a successful hypothesis or higher than a failed hypothesis. But no computer scientist would search Q this way, for such a search must test every element of Q in the worst case when the lowest element of Q is the only element of R. A more sensible approach follows, where [x,y] is an arbitrary interval of Q such that $x<y$, and the *midpoint* of [x,y] is the first z in [x,y] such that the number of $w>z$ in [x,y] is no more than one greater than the number of w' less than z in [x,y].

```
ChainSearch([x,y]);
        if |[x,y]|=2 then
        begin
                if y passes test then return y
                else if x does not pass test then return 'R is empty'
                else return x
        end
        else if the midpoint p of [x,y] passes test
                then ChainSearch([p,y])
        else ChainSearch([x,p]).
```

ChainSearch, with its balanced, "halving" strategy, finds the maximal element of R without ever considering more than on the order of log(|Q|) points in Q, and without ever testing a point lower than a successful point or higher than a failed point. This exponential reduction of effort in the worst case is not to be scorned. But on the other hand, the *best* case for the naive procedure requires only constant time (e.g. two tests: one for the least failure and one for the greatest success) while the best case for ChainSearch is, again, log(n).

Since the worst-case complexity of the "halving" strategy of ChainSearch does not dominate the worst-case complexity of the obvious, serial search in all cases, it is sensible to compare the expected complexities of the procedures. Since the complexity of ChainSearch is log(n) no matter where the maximal element of R is in Q, its expected complexity is also log(n). Now consider the naive, rote search. There are n possible positions for the maximal element p of R. If p is the ith point in the order, then i tests are performed by the procedure that starts at the bottom of the order and moves up, by rote. Assuming that each position is equiprobable, the expected complexity of this approach (in terms of tests performed) is therefore $(1+2+\ldots+n)/n=n(n-1)/n=n-1$. But $f(n)=n-1$ is not polynomially related to $g(n)=\log(n)$. So the halving policy of ChainSearch has an exponential *expected* advantage--- which is a good reason to employ it, if possible.

Between any two connected points x,y in a lattice L there is a sublattice [x,y], which consists of the set of all points in the intersection of Up(y) and Down(x). This sublattice is called the *interval* between x and y in L. An obvious question is whether the efficient ChainSearch method just presented can be extended in a useful manner to intervals in a lattice, as a generalization of its application to intervals in simply ordered sets only.

Since each path from the top of the lattice to the bottom is a chain, nothing prevents us from applying ChainSearch to each such path. The set of all the results of these searches then has every suitable point in the lattice in its upward closure. For example, consider the lattice Canon(10). We begin by testing the clause specified by the partition Π_1=(12)|(34)|(56)|(78)|(90). Suppose it passes the test. Then we search the interval of Canon(10) between Π and the lattice bottom. A midpoint in this interval is Π_2=(1)|(2)|(3)|(4)|(56)|(78)|(90). Suppose Π_2 fails. Then we look at the interval $[\Pi_1,\Pi_2]$, and generate a midpoint, say Π_3=(12)|3|4|(56)|(78)|(90). Suppose Π passes the test. Since $|[\Pi_2,\Pi_3]|=2$ and since both endpoints pass the test, Π_3 is a maximal suitable specification of B with respect to the given evidence. So for a single path, ChainSearch works just fine.

But the new complication that arises in sets that are not simply ordered is this: once a maximal success is found on one such path in Part(n), this success will cover the initial segments of very many other paths, so it would be foolish to search all possible paths independently, using ChainSearch. Indeed,the degree of overlap is so large compared to n that the problem of generating an element Π of the complement of (Up(S) U Down(F)) is the more critical matter by far.

But suppose we had a way to find, for any disjoint subsets S,F of Part(n), two connected points in the lattice that are not in Up(S) U Down(F) that are *maximal with respect to S,F* in the following sense: the upper point does not refine a point not in Up(S) and the lower point has no point not in Down(F) as a refinement. In effect, such a procedure would locate the endpoints of entirely unsearched intervals in Part(n). The usual ChainSearch algorithm can be applied to such an interval by selecting an arbitrary midpoint, testing, selecting an arbitrary midpoint of the resulting interval, and so forth.

More formally, the procedure SEARCH selects endpoints for an uncovered lattice interval and then calls the analogue of ChainSearch just described on the result.

```
SEARCH(n)
begin
        set conjecture:=S:=F:={};
        while there is a Π∈Part(n))-(Up(S) U Down(F)
        begin
                choose some minimally refined Π'
                        in Part(n)-(Up(S) U Down(F));
                choose some maximally refined Π"
                        in Down(Π')-(Up(S) U Down(F));
                ChainSearch([Π',Π"])
        end;
        output conjecture
end.

ChainSearch([Π',Π"]);
        if |[Π',Π"]| = 2 then
        begin
                if y passes test then set conjecture:=conjecture U {y}
                else if x does not pass test then do nothing
                else set conjecture:=conjecture U {x}
        end
        else
        begin
                choose a Π such that
                        Π is a midpoint of [Π',Π"];
                if Π passes the test then
                begin
                        ChainSearch([Π,Π"]);
                        set S:=(S U {Π})
                end
        end
        else
        begin
                choose a Π such that
                Π is a midpoint of [Π',Π"];
                if Π fails the test then
                begin
                        ChainSearch([Π',Π]);
                        set F:=(F U {Π})
                end
        end
end.
```

The procedure works as follows. When S,F are empty, the top of the lattice is the (unique) maximally coarse partition not covered by S or F, and the bottom of the lattice is the maximally coarse partition covered by the top of the lattice but not by S or F. So the top and the bottom of the lattice are sent to ChainSearch. Chainsearch chooses an arbitrary element of the middle level of Part(n) and tests it. If the selected point passes muster, it is placed in S, and ChainSearch recurs in the usual way on the sublattice between this point and the bottom of Part(n). If it fails, the point is placed in F and Chainsearch recurs on the sublattice between this point

and the top of Part(n). This "halving" search continues until a sublattice that consists only of two connected points is reached, one of which must be a maximal suitable point. This point is placed in the conjecture, and control passes again to SEARCH. Now the top and bottom of Part(n) are no longer, respectively, the most refined and least refined elements of Part(n) not in Up(S) U Down(F). Hence, new endpoints are chosen and some path disjoint with Up(S) U Down(F) is tested by ChainSearch.[137]

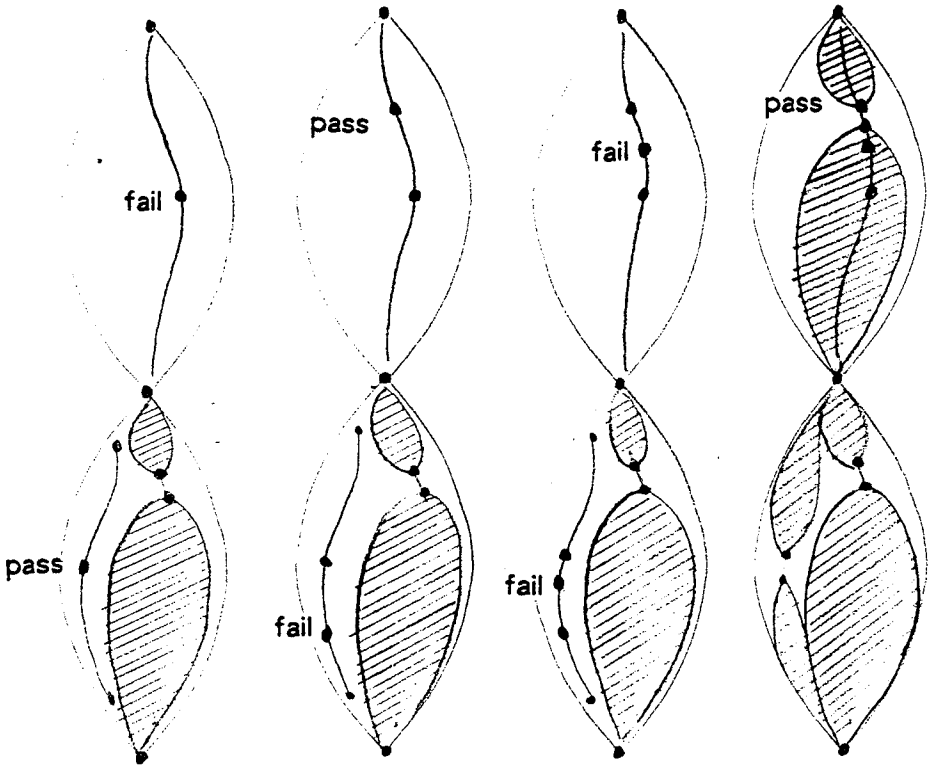


Figure 7-17: The Operation of SEARCH

This continues until Up(S) U Down(F) = Part(n), at which time the conjecture is exactly the maximal suitable subset of Part(n).

One might suspect that there is some difficulty in selecting a midpoint of a given pair of connected points in Part(n), as is required in ChainSearch. But this problem is computationally trivial. For example, consider the partitions Π=(12)|(34567890) Π'=(12)|(34)|(5)|(67)|(8)|(9)|(0). Any element of Down(Π) satisfies 12|34567890 and any element of Up(Π') satisfies (12)(34)(5)(67)(8)(9)(0). Hence, any element of the intersection satisfies (12)|(34)(5)(67)(8)(9)(0). Since Π is at level 2 and Π' is at level 7, their midpoints are at 5. Any such midpoint is the result of adding three slashes

[137] There must be such a path, for suppose there is not. Notice that the bottom endpoint is connected by some path to the upper endpoint, for it is chosen to be in the downset of the first endpoint. So every path must intersect with Up(S) U Down(F). But then one of the endpoints is in Up(S) U Down(F), contrary to the conditions under which they were chosen.

between CEXPRs in the expression (12)|(34)(5)(67)(8)(9)(0). A simple choice rule is to add each slash as far to the left as possible. The result of this process is (12)|(34)|(5)|(67)|(8)(9)(0). Since these are all the slashes permissible at level 5, we may take combine the last three CEXPRs to obtain (12)|(34)|(5)|(67)|(890). It is easy to see that each of these steps can be performed in time squared in the input size. And the process is no more difficult if the upward and downward closed sets in which the sought partition must be found are encoded as PEXPRs.

7.5.1. Finding an Element at a Level of Part(n)-(Up(S) U Down(F))

The crucial sub-problem for SEARCH is to find some minimally refined element Π of Part(n)-(Up(S)) U Down(F)) such that $|\Pi| = k$, for any given subsets S,F of Part(n) and $k \leq n$. Since this problem includes the special case in which F is empty, we know that it is NP-complete, for deciding whether there is a partition at or below level k in Part(n) that is not an element of Up(S) has been shown to be an NP-complete problem. If we could quickly decide for any given level whether such a partition exists at that level, then we could decide quickly whether there is one at that level or lower by iterating the procedure over each lower level.[138] Hence, any known technique for solving this problem will involve a hopeless search in some cases. Nonetheless, we can seek heuristics for the non-worst cases and still call the result elegant, even if it is not polynomial in the worst case.

An obvious approach is to enumerate the kth level of Part(n), and for each partition Π in this level, to check whether Π is a refinement of an element of F or is coarser than an element of S. These checks are linear in the size of S and F, so the only real difficulty is generating Lev(n,k). But since the cardinality of Lev(n,k) is the Stirling number $S_{n,k}$ of the second kind, this quantity is exponential in n when k is fixed at n/2. And when there exists no partition at level k that is not covered by S or K, all of Lev(n,k) will be generated and tested. So if we fix the cardinalities of S and F, but sample them from larger and larger partition lattices, the time required for this method grows exponentially in the worst case (e.g. when no partition of the sort sought exists). And, of course, increasing n is one way to increase the size of the input, so the resource consumption of this obvious procedure must grow exponentially in the input size in the worst case.

Due to the NP-completeness of the problem we are attempting to solve, such hopeless search is likely to be encountered on some problem instance for any

---

[138] The number of levels is at worst linear in n.

approach. But there is reason to suspect that the naive level search is a bit more naive than is necessary, for the enumeration of Lev(n,k) is not dependent on the constitution of S and F. We may be able to do far better than this in many cases by "constructing" the partition sought in light of S and F. The hope is that paying attention to the constraints imposed by S and F earlier rather than later will, in many cases, eliminate the gratuitous consideration of partitions that cannot possibly satisfy these constraints.

Consider a concrete example. Suppose we are searching Part(10) for a partition at level 5 that is coarser than no element of S and that is a refinement of no element of F, where F and S are given as follows.

F:
(123456789)|(0)
(12345678)|(90)
(123)|(4567890)
(45)|(12367890)

S:
1|(23)|4|5|6|7|8|(90)
1|2|3|4|5|6|(789)|0
1|2|(34)|5|6|(78)|9|0
1|2|3|(45)|(67)|8|9|0

For a sense of the magnitude of the search space involved, the cardinality of Part(10) is one hundred fifteen thousand nine hundred seventy five, and the cardinality of Lev(10,5) is forty-two thousand five hundred twenty-five. If we can find a way to solve the problem without inspecting over forty-two thousand partitions in the worst case, then we shall have made some progress.

One way to "construct" the desired partition out of S and F is to think of the descriptions of elements of F and S as PEXPRs, and to transform the disjunction of these PEXPRs into a state description of a partition at the target level that is not connected to any element of S or of F. More specifically, Down(F) is just the 10-elementary class of the result F' of removing parentheses from F, while Up(S) is the 10-elementary class of the result S' of removing bars from S.

F':
123456789|0
12345678|90
123|4567890
45|12367890

S':
1(23)45678(90)
123456(789)0
12(34)56(78)90
123(45)(67)890

All the constants in S' that are not within parentheses can be eliminated without altering what S' says in any way. So we have

F':
123456789|0
12345678|90
123|4567890
45|12367890

S':
(23)(90)
(789)
(34)(78)
(45)(67)

The task now is to find some Π of cardinality five that does not satisfy (F' v S'). So Π must satisfy -S' & -F'. That is, Π must satisfy the CNF PEXPR

P:

(10)	(19)	(14)	(14)	2|3	7|8	3|4	4|5
(20)	(10)	(15)	(24)	9|0	7|9	7|8	6|7
(30)	(29)	(16)	(34)	7|9			
(40)	(20)	(17)	(46)				
(50)	(39)	(18)	(47)				
(60)	(30)	(19)	(48)				
(70)	(49)	(10)	(49)				
(80)	(40)	(24)	(40)				
(90)	(59)	(25)	(15)				
	(50)	(26)	(25)				
	(69)	(27)	(35)				
	(60)	(28)	(56)				
	(79)	(29)	(57)				
	(70)	(20)	(58)				
	(89)	(34)	(59)				
	(80)	(35)	(50)				
		(36)					
		(37)					
		(38)					
		(39)					
		(30)					

In this case, it is fairly easy to find a desired partition. Notice that a partition satisfies P just in case it satisfies at least one element of each column of P. Consider the conjunction of the first atom of each column:

(10), (19), (14), (14), 2|3, 7|8, 3|4, 4|5

By the union principle, this conjunction is p-equivalent to

(1490), 2|3, 7|8, 3|4, 4|5

By the merge principle, we have

(1490), 2|3, 7|8, 35|4

which is satisfied by the 10-partition of cardinality five described uniquely by

(1490)|(35)|(27)|(86)

In a sense, a satisfying partition was in the first place we looked. But a little reflection reveals a good deal of luck in this computation. For example, the following element of the cartesian cross of the columns of P is not satisfiable, for the atoms (90) and 9|0 both occur in it.

(90),(59),(25),(15),9|0,7|8,3|4,4|5

And if P were not satisfiable, each such conjunction would fail to be satisfiable. But there are one million, one hundred sixty-one thousand two hundred sixteen such conjunctions in the full Cartesian cross product of the columns of P. This number far exceeds the cardinality of Lev(10,5), and even exceeds the cardinality of the entire lattice Part(10). Hence, examining each element of the Cartesian cross product of the columns in P until a satisfiable conjunction is found is, in this case, dismally worse than the exhaustive search of Lev(10,5) upon which we are attempting to improve.

But a rote search through all possible elements of the cross product of the columns of P is not the only way to ensure that a satisfying partition will be found if there is one. Perhaps there exist general heuristics that keep the explosion under feasible control. Consider the first two columns of P. Notice that the only atom that occurs in the first column but not in the second is (90). Since (a v b) & (a v c) is logically equivalent to a v (b & c), each of these atoms can be removed from the two columns before the cross product is taken. So the conjunction of the first two columns is logically equivalent to:

(10)
(20)
(30)
(40)
(50)
(60)
(70)
(80)
(90), (19)
(90), (29)
(90), (39)
(90), (49)
(90), (59)
(90), (69)
(90), (79)
(90), (89)

By the union principle, we have

(10)
(20)
(30)
(40)
(50)
(60)
(70)
(80)
(190)
(290)
(390)
(490)
(590)
(690)
(790)
(890)

But notice that each of the ternary CEXPRS in this column entails some binary CEXPR in the column. Hence, the entire column (disjunction) is logically equivalent to

(10)
(20)
(30)
(40)
(50)
(60)
(70)
(80)

So in this case, at least, the result of combining two columns is shorter than either of the columns combined. So even if every subsequent pairing must be examined,

we have reduced the number of conjunctions to be examined from ninety-six thousand seven hundred sixty-eight to six thousand five hundred fifty-two.

Now consider the third column and the column we have just obtained by distributing columns one and two. It is ominous that the columns to be combined this time share only three atoms, (10), (20), and (30), in common. This leaves 90 pairs of atoms to consider in crossing the next two columns. The reader can easily verify, however, that once the cross product is taken and disjuncts that entail other disjuncts are eliminated, the result is a PEXPR Q with seventy-eight rather than with ninety-three disjuncts.

(10)				
(20)				
(30)				
(15)(40)	(14)(50)	(14)(60)	(14)(70)	(14)(80)
(16)(40)	(16)(50)	(15)(60)	(15)(70)	(15)(80)
(17)(40)	(17)(50)	(17)(60)	(16)(70)	(16)(80)
(18)(40)	(18)(50)	(18)(60)	(18)(70)	(17)(80)
(19)(40)	(19)(50)	(19)(60)	(19)(70)	(19)(80)
(25)(40)	(24)(50)	(24)(60)	(24)(70)	(24)(80)
(26)(40)	(26)(50)	(25)(60)	(25)(70)	(25)(80)
(27)(40)	(27)(50)	(27)(60)	(26)(70)	(26)(80)
(28)(40)	(28)(50)	(28)(60)	(28)(70)	(27)(80)
(29)(40)	(29)(50)	(29)(60)	(29)(70)	(29)(80)
(35)(40)	(34)(50)	(34)(60)	(34)(70)	(34)(80)
(36)(40)	(36)(50)	(35)(60)	(35)(70)	(35)(80)
(37)(40)	(37)(50)	(37)(60)	(36)(70)	(36)(80)
(38)(40)	(38)(50)	(38)(60)	(38)(70)	(37)(80)
(39)(40)	(39)(50)	(39)(60)	(39)(70)	(39)(80)
cont. next column	cont. next column	cont. next column	cont. next column	end.

The prospect of crossing this disjunction with the fourth column of P is frightening, for the result contains twelve hundred forty-eight elements. But once again, a simple heuristic intercedes--- to an extent. For exampie, notice that the atom '(40)' occurs in the fourth column of P. Therefore, any such disjunct is already guaranteed not to be a refinement of (45)|(12367890), and hence need not be extended to ensure this fact. Such disjuncts may simply be added to the result of crossing the rest of the disjuncts of Q with the fourth column of P. The same point can be made regarding '(50)','(14)','(15)','(24)','(25)','(34)', and '(35)'. Eliminating the disjuncts in Q in which any one of these atoms occurs results in a disjunction with only thirty disjuncts. Crossing these remaining disjuncts with the sixteen atoms of the fourth column of P yields a disjunction Q' with at worst five hundred forty disjuncts. So by relying on our two, simple heuristics, we end up with a disjunction of five hundred forty, rather than forty-eight thousand three hundred eighty-four disjuncts,

which is what the cardinality of the naive cross product of the columns of P would have been.

Now consider the remaining columns, which correspond to the elements of S.

```
2|3     7|8     3|4     4|5
9|0     7|9     7|8     6|7
7|9
```

Notice that the first and third columns share the atom '7|9'. By the same reasoning as before, this atom may be removed from each column and may be added to the result of taking the Cartesian cross of the remaining atoms, to yield:

```
7|9
2|3, 7|8
9|0, 7|8.
```

Now consider the third column. Just as before, any disjunct D of the result C of combining columns 1 and 2 in which an atom of column 3 occurs may be added to the result of crossing the result of deleting D from C with column 3. In this case, '7|8' occurs in column 3 as well as in the second and third disjuncts of D. Hence, the result of combining columns 1,2, and 3 is just

```
2|3,7|8
9|0,7|8
7|9,3|4
7|89
```

There are no atoms shared between this disjunction and column four, so the result is:

```
2|3, 7|8, 4|5
9|0, 7|8, 4|5
7|9, 3|4, 4|5
7|89, 4|5
2|3, 7|8, 6|7
9|0, 7|8, 6|7
7|9, 3|4, 6|7
7|89, 6|7
```

So we have only eight disjuncts instead of 3x2x2=24 disjuncts to cross with the five hundred forty disjuncts that result from combining the columns corresponding to elements of n. When these eight disjuncts are crossed with the five hundred forty disjuncts of the expression for Down(F) developed previously, the result has four thousand two hundred forty disjuncts as opposed to the one million, one

hundred sixty-one thousand two hundred sixteen disjuncts in the full Cartesian cross product of the columns of P. Even if the overall algorithm is intractable, it is clear that many interesting instances have been brought within the light of feasibility by the application of some simple, safe, search-bounding heuristics.

Each of the four thousand disjuncts generated must be checked for satisfiability by an element of Lev(10,5). Since this satisfiability check is easy (each disjunct is a simple conjunction of atoms or "negated" atoms) we have in this case a improvement over checking each of element of the forty-two thousand five hundred twenty-five elements of Lev(10,5) refinement from F or into S. And if the cardinalities of S and F had been smaller, the advantage would have been correspondingly more dramatic.

A natural concern arises when the cardinality of S U F approaches that of the lattice from which they are drawn. Perhaps in this case the more naive, exhaustive search of the entire lattice level is favorable to the heuristic cross product method. We can gain a feel for this by examining the operation of the latter method in a small lattice, where S,F are very large subsets of this lattice. Accordingly, let the lattice be Part(4), and let S,F be given as follows:

S:
(123)|(4)
(12)|(34)
(1)|(234)
(2)|(134)
(13)|(24)
(124)|(3)

S:
(1)|(2)|(34)
(1)|(23)|(4)
(12)|(3)|(4)
(13)|(2)|(4)
(1)|(24)|(3).

S U F includes all but four points of Part(4). Converting F to an expression whose elementary class is Down(F) yields

(14)\(13)\(12)\(12)\(12)\(13)
(24)\(14)\(13)\(23)\(14)\(23)
(34)\(23)\(14)\(24)\(23)\(34)
\(24)\\\(34)

Combining the first and second columns yields

(14)
(24).

Combining this with the third column yields

(14)
(13)(24)

Combining this with the fourth column results in

(124)
(14)(23)
(13)(24)

The next cross leads to

(124)
(14)(23).

The final cross results in

(14), (23)

which is satisfied by the top of the lattice and by the one partition at level two that was not included in F, namely (14)|(23).

Now for the set S. Its upward closure is expressed by

3|4 2|3 1|2 1|3 2|4

or by the p-equivalent expression '3|2|14'. Conjoining the result for S with the previous result for P yields

(14),(23),3|2|14.

There is no intuitive hint of "exponential explosion" in this example. To optimistic eyes, this example suggests that although there are many cross products to perform when S U F is a large portion of Part(n), the heuristics are more effective in reducing the number of disjuncts to cross at each stage. And even if the heuristics are weak or ineffectual when S U F is a small portion of P, there are fewer cross products to perform than points to inspect at the target level of the lattice.

This optimism is extremely tenuous, and demands a much more careful and detailed analysis. Merely simulating the algorithms on particular problem instances tells us

nothing at all about how the resource consumption of the two methods is related over arbitrary problem instances, or even over some interesting, infinite subset of these instances. In particular, it would be interesting to know whether there exists a problem instance of each size such that the resource consumption of the level search procedure grows exponentially more quickly than that of the logical algorithm. And if the PEXPR crossing algorithm presented in this chapter does not dominate the naive level search algorithm, it may still be possible to isolate conditions under which one strategy dominates the other and *vice versa*. These conditions, if easily decidable in S,F, and k, could be employed to choose effectively between the level search and heuristic cross product strategies. A procedure that could select the best strategy for the prevailing conditions would then be more efficient than either of these strategies could be by itself. I have not yet settled any of these issues.

7.6. "Constructive" Procedures for Partition Lattice Search

In the previous section, I proposed two "generate-and-test" approaches to computing the inductive strategy of CONSIST. The popular image of a generate-and-test procedure is that hypotheses pop out of one unit, as in an automobile factory, and are shunted on a conveyer belt to another unit that sorts them as suitable or unsuitable in light of the evidence available.

An immediate consequence of this picture is that the selection of the next hypothesis to test is unaltered if evidence not relevant to the acceptance or rejection of a previously tested hypothesis is materially altered. Or to restate the point, no evidence irrelevant to the acceptance or rejection of any previously considered hypothesis can serve to guide or to restrict the search among alternative hypotheses. Hence, there is a *prima facie* sense in which a generate-and-test method seems needlessly hobbled. If the goal is to generate the maximal, suitable subset of Hyps(n) at stage n, then it would seem sensible to bring the evidence in to constrain the search for such a set as early as possible. Indeed, as the evidence is brought more intimately into the picture, discovery tends to look less like conjecture and refutation and more like "constructing" an hypothesis out of the evidence provided. Moreover, the NP-completeness theorems presented earlier applied only to generate-and-test approaches to computing the strategy of CONSIST. There may be new intractability theorems to be found for the constructive approach, but at least I am not yet certain that this approach must solve an NP-complete problem.

So how might the maximal, suitable subset of the lattice of specifications of a given clausal blank be *constructed* from the given evidence? Recall that in this discussion, suitability is just consistency with the evidence. Notice that the counterexamples to specifications of a blank involving p atoms are p-tuples of atoms or negated atoms in the evidence, where the ith atom in the tuple has the same predicate but opposite sign as the ith predicate in the blank. If there are no such p-tuples in the evidence, every element of the specification lattice is suitable and we are done.

Now suppose that we can build sequences of atoms of the sort just described from the evidence provided. We know immediately that some elements of the lattice under search must be rejected. Moreover, the weakest such clause (there must be exactly one) can be constructed directly from the p-tuple so obtained by substituting variables uniformly for constants, by changing the signs on all the atoms occurring in the tuple, and by disjoining the basic formulas that result. So for example, suppose we are searching the specifications of the blank B= <P**,-Q***>. Moreover, assume we are supplied with the evidence

Pab, -Qabc, Qada, -Pcc

Notice that 'Pab' is irrelevant to any specification of B, as is '-Qabc', for they can never be involved in a counterexample to any specification of B. Hence, we may delete them from the evidence so far as B is concerned. Now we can form the pair

<-Pcc, Qada>

This sequence forms a counterexample to the specification

$(Px_1x_1 \vee -Qx_2x_3x_2)$

or to the partition

(12)|(35)|(4),

which is uniquely associated with this specification. But there are evidently no more counterexamples to specifications of B available in the given evidence. So the problem becomes simply to find the maximal generators of the complement of the downset of Π=(12)|(35)|(4).

Once again, it is tempting to employ PEXPRs. Down(Π) is just the 5-elementary class of the PEXPR

12¦35¦4

so Part(5)-Down(Π) is just the 5-elementary class of the negation of '12¦35¦4'. That is, the complement is the 5-elementary class of

(13)
(15)
(14)
(23)
(25)
(24)
(34)
(45)

Now it is easy to read off the most refined partitions (i.e. the logically strongest clauses) that satisfy this expression: just extend each disjunct in the most refined manner by adding any constant not mentioned in the given disjunct as a distinct, unit cell. In this case we have

(13)¦(2)¦(4)¦(5)
(15)¦(2)¦(3)¦(4)
(14)¦(2)¦(3)¦(5)
(23)¦(1)¦(4)¦(5)
(25)¦(1)¦(3)¦(4)
(34)¦(1)¦(2)¦(5)
(45)¦(1)¦(2)¦(3).

So much for the innocent joy of considering easy cases. Notice that the implicit strategy employed here is as follows:

1. Formulate all counterexamples to specifications of B that can be found in the evidence.

2. Convert each counterexample into the unique partition that corresponds uniquely to the pattern of its constants in the usual manner.

3. Convert these counterexamples into paren-free PEXPRs.

4. Negate the disjunction of the results.

5. Solve for a p-equivalent DJNF PEXPR P.

6. Solve for the maximal generators of the n-elementary class of P by placing any constant not mentioned in a disjunct into a distinct unit cell.

Notice also that the strategy is to find the maximal elements of the complement of a set of failures. In the theory of NP-completeness, *maximization* problems are

typically studied as decision problems. If the maximization problem is to find a maximal such and such, the corresponding decision problem is to decide whether there is a such and such whose measure is bounded below by a parameter k. If such a decision problem is difficult then the maximization problem is as well, for a fast solution to the maximization problem would provide a fast solution to the corresponding decision problem. Now recall that it has already been shown that to discover whether there is a partition at or below Lev(n,k) that does not satisfy some paren-free PEXPR P is NP-complete. Once again, it is frustrating that our intractability theorem concerns only the problem in which a given downset is encoded by a PEXPR, while the problem at hand encodes the downset to be avoided by the maximal, generating set of the downset which may be exponentially larger. But there is at least some reason to worry that the constructive strategy faces an NP-hard problem.

Another difficulty is inherent in the constructive technique's consideration of counterexamples in the evidence. If B is a p-tuple, and if there are m evidential instances of the correct sign for each predicate occurring in B, then there will evidently be m^p counterexamples to consider, which quantity can evidently be exponential in the size of the evidence in the worst case. Hence, the method is intractable even if the maximization problem just discussed is not NP-complete.

We can conceive of a malicious case in which the compiled evidence leads our constructive method on a "grand tour" of the entire lattice to be searched when every hypothesis in the lattice is refuted. This would happen when the first counterexample to be found is the bottom of the lattice, the next to be found are all on the penultimate level of the lattice and so forth. An obvious remedy is to eliminate any counterexample P formed in the evidence such that there is a substitution θ and another counterexample P' in the evidence such that $P\theta=P'$. That is, any counterexample to a clause that entails a clause for which there is a counterexample should be deleted from the evidence (so far as blank B is concerned). Since the tests involved are linear, and there are only polynomially many pairs of p-tuples once the p-tuples are considered, this expedient would be tractable and beneficial when there are few counterexamples in the evidence compared to the size of the lattice. But of course there can be many counterexamples for each point in the lattice, in which case this method would be as bad as considering every point in the lattice to be searched.

So once again, there is the possibility of isolating the point of diminishing returns when it would make sense to revert to a generate-and-test strategy or to an

exhaustive search. Such a decision could be made as soon as it is determined how many atoms of the appropriate sign occur in the evidence for each predicate occurring in B. If the product of these cardinalities is on the order of the size of the lattice to be searched, the constructive approach could be abandoned in favor of a generate and test approach.

Chapter 8
Conclusion

While the first half of the thesis was intended to sketch the logic of discovery quite broadly, the latter half should be thought of more as a narrow "core sample" into the topic. I began the technical second half of the thesis by narrowing the focus to a particular, logically oriented inductive inference problem. A procedure M was taken to AE–U–identify world w just in case for each purely universal hypothesis h true in w there is some point in reading its evidence after which each of M's conjectures entails h, and for each hypothesis h' false in w, there is a point after which no conjecture of M entails h'. The problem is to AE–U–identify every world whose domain elements are describable in the evidence. This problem contrasts with the inductive inference problems studied by computer scientists, in which the method is required to converge "all at once" to an adequate conjecture.

Next, I proposed three suitability relations familiar in philosophy of science to see whether relying on them in the short run results in inductively general behavior in the sort of problem under consideration, with respect to AE identification. It turns out that a variant of a Hempelian method, a method based loosely on Nicod's criterion, and a method that relies merely on consistency with the evidence, all constitute general solutions to the problem. And lest it be thought that these methods could be improved to converge "all at once" to an adequate theory, I proved that *no possible* method that is a function of the evidence, be it effective or otherwise, can have this property.

In the following chapter, the issue of inductive efficiency is raised, for the methods just described are patently inelegant. But upon reflection, it is not obvious what efficiency means for AE–convergent computations. To investigate this notion, I presented the standard (short–run) theory of complexity, and proceeded to discuss the efforts of Angluin, Gold, Daley, and Smith to apply complexity–theoretic concepts to problems of AE or "all at once" identification. First, I examined the significance of NP–completeness results for conjecturing behaviors in light of the view that an identification problem is in a complexity class if and only if it is solved

by some behavior that is in this class (in the short run). Lower complexity bounds on particular conjecturing behaviors are not of very general significance in light of such a theory, for the question always arises whether a more easily computed conjecturing behavior solves the same problem. But then it turns out that there is a fast solution for *every* EA-identification problem in this sense. We can obtain more intuitive results by augmenting a limiting inference problem with a suitability relation that must be satisfied by each of a method's conjectures. From this point of view, NP-completeness theorems like those of Gold and Angluin take on a greater interest, for they are about a limiting inference problem augmented by the requirement that a given suitability relation be satisfied by one's conjectures in the short run.

Next, the Daley/Smith theory was criticized for its relativization to the input order of the evidence and its want of a substitute for the usual asymptotic complexity classes of the short-run theory, due to the fact that resource bounds are defined on functions rather than natural numbers in this theory. Without a natural notion of size for functions, there is no natural way to speak of an inference problem as being "polynomial" or "exponential". Four potential solutions to this technical difficulty were proposed, with mixed results.

Next, I turned to the more general setting of AE-identification problems. A "milestone" theory of AE complexity was introduced, which presupposes a concept of verisimilitude. Such a concept was proposed, but the resulting complexity theory is counterintuitive. Ultimately, I retrenched to the position of comparing the short-run complexities of inductive strategies, which are equivalence classes of computable functions in the usual sense. Finally, I addressed the relationship between the complexity of deciding a test relation as opposed to the generational complexity of a conjecturing device that observes this relation.

The next chapter proposes a particular strategy for the procedure CONSIST, whose conjectures are always consistent with the evidence. Then, we proceeded to investigate procedures that compute this strategy more efficiently by ignoring redundant hypotheses, regardless of the evidence at hand (i.e. *a priori*). I noted two aims. The first is to avoid unnecessary empirical tests, but the second is to avoid even considering hypotheses that needn't be tested. In pursuit of these goals, the problem of variable-renaming variants was confronted, and a method for ignoring exponentially many such redundancies was proposed. Next I addressed the problem of predicate permutation redundancies, with equal success. These techniques are applicable not only to HEMP, NICOD and CONSIST, but to Shapiro's model inference

systems as well. In fact, these techniques apply to any program that searches among clausal hypotheses.

Finally, the problem of non-reduced clauses was considered. A clause is non-reduced just in case it is logically equivalent to a sub-clause of itself. Here, success was not so easy to obtain. It was demonstrated that an apparently simple case of clausal entailment is NP-hard, and that deciding whether a given clause is a reduced form of another given clause is NP-hard. These facts do not in themselves prove that it is hard to generate all the reduced clauses of a given class or that deciding whether a given clause is reduced is intractable. But they do tell us that many obvious approaches to solving the problem are likely intractable.

An interesting spin-off of the entailment theorem is that deciding clausal consistency with the evidence is NP-hard, even for the very simple clauses considered in this thesis. So consistency tests are to be avoided if possible, which underscores the importance of finding ways to withhold unnecessary tests. It was also noted that the problem of finding some maximal diagram for a structure for the hypothesis language in a given set of evidence is NP-hard. Hence, HEMP's strategy of cutting down the evidence before testing its hypotheses does not elude the intractability of satisfaction testing.

In chapter seven, I investigated the exploitation of entailment structure among hypotheses to eliminate the consideration of redundant ones. It turns out that in our setting (but not necessarily in *every* setting) any logical consequence of a successful hypothesis may be ignored with no attendant loss in generality; as may any hypothesis that entails a failed hypothesis. So the hypotheses to be ignored are in upward and downward closed sets under the entailment order.

Since the hypotheses to be ignored are in closed sets, it makes sense to investigate the mathematical structure of such sets. I therefore presented some general mathematical facts about partition lattices, which are isomorphic to the structure of entailment over the specifications of a clausal blank. Then I introduced the PEXPR language whose sentences are useful data structures for encoding subsets of a partition lattice, and showed that such an expression can encode a closed set in exponentially less space than is required to list the maximal generators for this set.

Next, I turned to three obvious approaches for computing the strategy of CONSIST, and showed that each one of them has a subrfoutine that must solve an NP-complete problem or a problem so closely related to an NP-complete problem

as to be suspicious. A careful consideration of these worst-case results still leaves open many ways to achieve tractable performance in a broad range of special cases. Therefore, I proceeded to develop some methods that would be feasible in many cases, but ineluctably intractable on some. The proposed generate-and-test procedures employ a strategy that has logarithmic complexity when applied to chains. But to apply this strategy we ran squarely into one of the NP-completeness results of the previous section. A PEXPR based strategy that seemed to be feasible in many cases was presented and simulated through a non-trivial example in a lattice with over a hundred thousand points. Finally, a constructive procedure that attempts to "build" its conjecture up out of the evidence was presented. Despite plausible considerations to the contrary, this obvious procedure falls prey to just the sorts of difficulties that plague the generate-and-test methods already reviewed. Which brings us to the present.

The story just reviewed is a mixed bag of positive and negative results. Several key themes emerge. The first is that computational complexity is not a mathematician's dream. It is real; and almost tangible. We attempted to search lattices of clausal hypotheses from top to bottom, from bottom to top, by successively halving their intervals, by performing logical operations, and by searching their levels exhaustively. We considered constructing the hypothesis from the data, as well as testing hypotheses only after they are proposed. We thought of hypotheses as formulas, closed sets, partitions, and sequences of numbers. But the elegant sort of generation algorithm suggested by the early successes with variable-renaming variants and predicate permutation redundancies eluded us at every turn. Computational complexity is real and it is of the keenest significance to the logic of discovery, both practically and theoretically.

The next theme is that the logic of discovery is not trivial. The possibility of an elegant solution to an inductive problem cannot be dismissed *a la* Popper and the Positivists, with a wave of the hand. Negative claims and positive claims in the logic of discovery are equally claims, and mathematical ones at that. Indeed, an important feature of the logic of discovery that is illustrated in this thesis is our utter ignorance of it. I submit to the reader: is the inductive strategy of CONSIST an NP-hard problem or worse? This is a clear, mathematical question that has an answer. But it is not trivial to answer. And if the answer is yes, which restrictions of these problems are hard and which are easy? Are these restrictions intuitively as interesting as the original, intractable problem for the purposes of discovery? What modifications of the output conventions for Turing machines render the hard problems easy? There are also questions about inductive scope. For example, what

is an example of a class of first-order theories that is not AE-identifiable? How simple a vocabulary will support such a theorem? Are there hierarchies of AE-identifiable classes? And so forth. Along with the myriad mathematical questions, we also encounter some non-trivial philosophical questions. For example, what is efficiency for AE-convergent computations?

If the unpopularity of the logic of discovery among philosophers and methodologists derives from the fact that epistemologists don't know what to *do* with it, then this worry should be dispelled by now. There is plenty to do, for so little has been done.[139] And ignorance is only bliss while one is ignorant of one's ignorance.

In short, I wish to leave the impression that there is really something here to investigate. It is generally interesting, sometimes maddening, rarely trivial, and occasionally of significant practical utility.

---

[139] I in no way intend to demean such efforts as computer scientists and philosophers have expended on the logic of discovery. Rather, I intend to emphasize how large the void in our knowledge of the subject remains.

Appendix: Proofs for Chapter One

1. Proof of Fact 1

Let S be our n+1-ary R.E. relation of suitability. Let P_s be the program which returns 'yes' in finitely many steps on an input n+1-tuple if and only if the tuple is in S. Since our programming language is acceptable, there exists a universal program U, such that when U is supplied program $P_{s'}$ an input i for $P_{s'}$ and some integer j as inputs, it simulates $P_s(i)$ for exactly j steps, so that

$$U(P_{s'}, i, h, j) = \begin{cases} P_s(i,h) \text{ if } P_s(i,h) \text{ returns an answer in } j \text{ steps.} \\ \text{'nothing yet' otherwise.} \end{cases}$$

Recall that the hypothesis language H is recursive. This implies that H is recursively enumerable. Any recursively enumerable set can eventually be listed on a tape by some machine. Call H's listing-machine P_H. Now we define a third machine M, which takes an S-situation as input as follows:

Stage 0:
Read the input S-situation. Call it I;
Set HYPS = {};
Go to stage 1;

Stage m:
Set k = 1
Generate hypothesis h_m of H using P_H and add it to HYPS;
UNTIL EITHER ANS is not 'nothing yet' OR k = m + 1 DO
SET ANS = $U(P_{s'}, \langle I, h_m \rangle, k)$;
IF ANS is not 'nothing yet' THEN RETURN ANS;
ELSE DO Stage m+1.

This construction is commonly referred to as "dovetailing" in the recursion theoretic literature. The program will test S for 1 step on the first hypothesis in the enumeration, for 2 steps on the first two hypotheses, ..., for n times for the first n hypotheses, and so forth.

Note that if there is an hypothesis h which bears S to I, then $P_s(I, h)$ = 'yes' in some number k of steps of execution. Suppose h_w is such an h. Without loss of generality, we can also suppose that for no h_x, for $x < w$, is it true that $P_s(I, h)$ =

'yes' in p or fewer steps (for if there is an h such that i R h, some h' has this property). Since there is a stage s of M in which s $\geq$ w and s $\geq$ m, h_w will eventually be output by M at stage s. Furthermore, it is clear by the fact that M has only one OUTPUT statement, that no hypothesis is ever output under any other circumstances. Hence M is strongly adequate with respect to S.

2. Proof of Fact 2

Let R = {<x,y,z>: z = 1 and the program with Goedel index x does not return an output on input y}. Let R' = {<x,y,z> such that z = xy+2} Let suitability relation S = R U R'. R is co-R.E. but non-recursive because R is the complement of the halting problem, which is recursively enumerable but non-recursive. Moreover, R' is recursive. Any recursive set is both R.E. and co-R.E. Moreover, the intersection of any two R.E. sets is again R.E., so by the DeMorgan law, the union of any two co-RE. sets is co-R.E. It is important to show that S is not R.E. Notice that if S were R.E., R would be R.E. as well, by enumerating S, and picking off tuples whose third elements are '1' (no element of R' has '1' as third element, so the distinction can be made by simple matching). But R is not R.E., so S is not. Finally, we use Fact 1 to produce the strongly adequate hypothesis generator for relation R and hence for relation R U R'.

3. Proof of Fact 3

===> Use the construction of Fact 1 on relation S' to yield a strongly adequate hypothesis generator M for S'. Since S' is a subset of S, M makes no errors of commission in any S-situation according to S. Since for any S-situation W, S' thinks at least one of the hypotheses bearing S to W is suitable, M is guilty of no errors of omission with respect to S. <=== Since M makes no error in any situation, the set {$<w_1,\ldots,w_n,M(w_1,\ldots,w_n)>$: $<w_1,\ldots,w_n>$ is an S-situation} is the desired subrelation S' of S. Moreover, S' is R.E., for there is a machine M' which when fed any tuple $<w_1,\ldots,w_n,h>$, runs M on the S-situation part and waits to see if the output is h, in which case M' returns 'yes'.

4. Proof of Fact 4

Let S be an arbitrarily "hard" suitability relation for hypotheses in H. Code H by Goedel numbers, and consider the relation S" just like S except h bears S to S-situation W if and only if h", whose Goedel number is twice that of h, bears S" to W. A program for S" would, with the addition of the coding machine, result in a

program for S, so S" is just as hard as S. Moreover, S" does not count infinitely many hypotheses (those with odd Goedel numbers) as suitable in any situation. Those H's with odd Goedel numbers form a recursive set ODD, for oddness is recursive and Goedel numbering is computable. (All that is required is an arbitrary 1-1 effective map of H into the complement of an arbitrary recursive subset of H. The above construction shows there is one.) There is an R.E. suitability relation S' which counts the hypothesis 1 as suitable in any possible S-situation. As long as there is one such relation, there are all its infinitely many finite variants S'_i which have a suitable hypothesis for every S-situation and which allow only hypotheses in ODD to be suitable. Now consider the relations S" U S'_1, S" U S'_2,.... Each of these is as unsolvable as S, for a machine which decides tuples in S" U S'_i can easily be turned into a machine to decide S. Let the machine M_s for S take an arbitrary tuple T of the correct type. First, it looks at the last element h of T, and runs the characteristic machine of ODD on it. If the result is 'No', then M_s returns 'No'. Otherwise, M_s codes h, divides the code number by two, and decodes the result to get h". Finally, h" is substituted for h in T to form T'. Now M_s runs the assumed machine, M_s, on T', and returns whatever the result of this computation is. Finally, Fact 3 guarantees the strongly adequate hypothesis generation machine for each S" U S'_i, of which there are infinitely many.

REFERENCES

[Aigner 79] Martin Aigner.
Combinatorial Theory.
Springer Verlag, New York, 1979.

[Angluin 78] Dana Angluin.
On the Complexity of Minimum Inference of Regular Sets.
Information and Control 39:337–350, 1978.

[Angluin 80] Dana Angluin.
Finding Patterns Common to a Set of Strings.
Journal of Computer and System Sciences 21:46–62, 1980.

[Angluin 81] Dana Angluin.
A Note on the Number of Queries Required to Identify Regular Languages.
Information and Control 51:76–87, 1981.

[Angluin 82] Dana Angluin and Carl H. Smith.
A Survey of Inductive Inference Methods.
Technical Report 250, Yale University, October, 1982.

[Aristotle 41] Richard McKeon (editor).
The Basic Works of Aristotle.
Random House, New York, 1941.

[Arrow 51] K. J. Arrow.
Social Choice and Individual Values.
Wiley and Sons, New York, 1951.

[Blum 75] Lenore and Manuel Blum.
Towards a Mathematical Theory of Inductive Inference.
Information and Control 28:125–55, June, 1975.

[Boolos 80] George S. Boolos and Richard C. Jeffrey.
Computability and Logic.
Cambridge University Press, Cambridge, 1980.

[Bradshaw 80] Gary L. Bradshaw, Pat Langley, and Herbert A. Simon.
BACON.4: The Discovery of Intrinsic Properties.
In *Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence*, pages 19–25. 1980.

[Bruner 56] Jerome S. Bruner, Jaqueline J. Goodnow, and George A. Austin.
A Study of Thinking.
John Wiley and Sons, New York, 1956.

[Carnap 36] Rudolph Carnap.
Testability and Meaning.
Philosophy of Science 3:419–471, 1936.

[Carnap 50] Rudolph Carnap.
Logical Foundations of Probability.
University of Chicago Press, Chicago, 1950.

[Case 78] J. Case and C. Smith.
Anomaly Hierarchies of Mechanized Inductive Inference.
In *Proceedings of the Tenth ACM Symp. on Theory of Computing*, pages 314–319. 1978.

[Chang 73] C. C. Chang and H. J. Keisler.
Model Theory.
North–Holland, Amsterdam, 1973.

[Crespi–Reghizzi 71] Stefano Crespi–Reghizzi.
Reduction of Enumeration in Grammar Acquisition.
In *Proc. of the Second Int. Joint Conf. AI*, pages 546–552. 1971.

[Daley 84] Robert P. Daley and Carl H. Smith.
On the Complexity of Inductive Inference.
Technical Report, University of Maryland, September, 1984.

[Dennett 85] Daniel C. Dennett.
Brainstorms: Philosophical Essays on Mond and Psychology.
MIT Press, Cambridge, Massechusetts, 1985.

[Donnellan 66] Thomas Donnellan.
Lattice Theory.
Pergamon Press, Oxford, 1966.

[Feldman 67] Jerome Feldman.
First Thoughts on Grammatical Inference.
Technical Report 55, Stanford University Artificial Intelligence Memo, 1967.

[Feldman 72] A.W. Biermann and J.A. Feldman.
A Survey of Grammatical Inference.
In *Frontiers of Pattern Recognition*, pages 31–54. Academic Press, New York, 1972.

[Fischer 74] M. J. Fischer and M. O. Rabin.
Super–exponential Complexity of Presburger Arithmetic.
SIAM-AMS Proceedings 7:27–41, 1974.

[Fu 75] K.S. Fu.
Syntactic Methods in Pattern Recognition.
Academic Press, New York, 1975.

[Garey 79] Michael R. Garey and David S. Johnson.
Computers and Intractability: A Guide to the Theory of NP-Completeness.
W.H. Freeman and Company, New York, 1979.

[Glymour 77] Clark Glymour.
Indistinguishable Space–Times and the Fundamental Group.
Minnesota Studies in the Philosophy of Science 8:50–60, 1977.

[Glymour 80] Clark Glymour.
Theory and Evidence.
Princeton University Press, Princeton, New Jersey, 1980.

[Glymour 84] Clark Glymour.
Inductive Inference in the Limit.
Erkenntnis 21:00-00, 1984.

[Gold 65] E. Mark Gold.
Limiting Recursion.
J.S.L. 30:28-48, 1965.

[Gold 67] E. Mark Gold.
Language Identification in the Limit.
Information and Control 10:447-474, 1967.

[Gold 78] E. Mark Gold.
Complexity of Automaton Identification from Given Data.
Information and Control 10:447-474, 1978.

[Green 69] C. Cordell Green.
Theorem Proving by Resolution as a Basis for Question Answering.
Machine Intelligence 4:183-205, 1969.

[Hempel 43] C. G. Hempel.
A Purely Syntactical Definition of Confirmation.
Journal of Symbolic Logic 54:00-00, 1943.

[Hempel 65] C. G. Hempel.
Aspects of Scientific Explanation.
Collier Macmillan, London, 1965.

[Hopcroft 79] John E. Hopcroft and Jeffrey D. Ullman.
Introduction to Automata Theory, Languages, and Computation.
Addison-Wesley, Reading, Mass., 1979.

[Horning 69] J.J. Horning.
A Study of Grammatical Inference.
PhD thesis, Stanford University, August, 1969.

[Hunt 66] Earl Hunt, Janet Marin, and Philop J. Stone.
Experiments in Induction.
Academic Press, New York, 1966.

[Kugel 77] P. Kugel.
Induction, Pure and Simple.
Inform. Contr. 35:276-336, 1977.

[Laudan 77] Larry Laudan.
Progress and its Problems.
University of California Press, Berkeley, 1977.

[Laudan 80] Larry Laudan.
Why was the Logic of Discovery Abandoned?
In *Scientific Discovery, Logic, and Rationality.* D.Reidel, Dordrecht, 1980.

[Levi 83] Isaac Levi.
The Enterprise of Knowledge.
M.I.T. Press, Cambridge, Mass., 1983.

[Levin 73] L. A. Levin.
Universal Sorting Problems.
Problemy Peredaci Informacii 9:115-116, 1973.

[Machtey 78] Patrick C. Fischer (editor).
The Computer Science Library: An Introduction to the General Theory of Algorithms.
North Holland, Amsterdam, 1978.

[Manders 86] Kenneth Manders.
Epistemological Aspects of Conceptual Innovation in Mathematics.
1986.

[Manuel 68] Frank E. Manuel.
A Portrait of Newton.
MIT Press, Cambridge, Mass., 1968.

[McMullin 84] Ernan McMullin.
A Case for Scientific Realism.
In Jarrett Leplin (editor), *Scientific Realism.* University of California Press, Berkeley, 1984.

[Michalski 83] Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (editors).
Machine Learning: An Artificial Intelligence Approach.
Tioga Publishing Co., Palo Alto, Ca., 1983.

[Minsky 69] Marvin Minsky and Seymour Pappert.
Perceptrons: An Introduction to Computational Geometry.
MIT Press, Cambridge Mass., 1969.

[Nickles 80] Thomas Nickles (editor).
Scientific Discovery, Logic, and Rationality.
D.Reidel, Dordrecht, 1980.

[Osherson 86] D.N. Osherson, M. Stob, and S. Weinstein.
Mechanical Learners Pay a Price for Bayesianism.
1986.

[Pao 69] T. S. Pao.
A Solution of the Syntactical Induction-Inference Problem for a Non-Trivial Subset of Context-Free Languages.
Technical Report, Moore School of Electrical Engineering, University of Pennsylvania, August, 1969.

[Pao 78] T.W.Pao and J.W. Carr, III.
A Solution of the Syntactical Induction-Inference Problem for Regular Lantuages.
Comput. Lang. 3:53-64, 1978.

[Pitt 84] Leonard Pitt.
A Characterization of Probabilistic Inference.
Technical Report 319, Yale University, June, 1984.

[Putnam 63] Hilary Putnam.
'Degree of Confirmation' and Inductive Logic.
In Arthur Schilpp (editor), *The Philosophy of Rudolph Carnap.* Open Court, LaSalle, Illinois, 1963.

[Quine 69] Willard Van Orman Quine.
Ontological Relativity & Other Essays.
Columbia University Press, New York, 1969.

[Rawls 64] J. Rawls.
Legal Obligation and the Duty of Fair Play.
In J. Murphy (editor), *Civil Disobedience and Violence*. University of California Press, Belmont, California, 1964.

[Reichenbach 49] Hans Reichenbach.
The Theory of Probability.
University of California Press, Berkeley, 1949.

[Reynolds 70] J. C. Reynolds.
Transformational Systems and the Algebric Structure of Atomic Formulas.
Machine Intelligence 5:135-153, 1970.

[Robinson 65] J.A. Robinson.
A Machine Oriented Logic Based on the Resolution Principle.
JACM 12, January, 1965.

[Rogers 67] Hartley Rogers.
Theory of Recursive Functions and Effective Computability.
McGraw-Hill, New York, 1967.

[Salmon 66] Wesley C. Salmon.
The Foundations of Scientific Inference.
University of Pittsburgh Press, Pittsburgh, 1966.

[Sen 70] A. K. Sen.
Collective Choice and Social Welfare.
Holden-Day, San Francisco, 1970.

[Shapiro 81] Ehud Y. Shapiro.
Inductive Inference of Theories from Facts.
Research Report 192, Yale University: Department of Computer Science, February, 1981.

[Solomonoff 64] R. Solomonoff.
A formal theory of inductive inference.
Inform. Contr. 7:1-22,224-254, 1964.

[Stockmeyer 73] L. J. Stockmeyer.
Planar 3-Colorability is NP-Complete.
SIGACT News 5.3:19-25, 1973.

[van Fraassen 80] B. C. van Fraassen.
The Scientific Image.
Clarendon Press, Oxford, 1980.

[Winston 75] Patrick Henry Winston.
Learning Structural Descriptions from Examples.
In *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.