Machine Learning, 8, 167-219 (1992)

Abductive Explanation-Based Learning: A Solution to the Multiple Inconsistent Explanation Problem

WILLIAM W. COHEN wcohen@research.att.com
AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

Editor: Paul Rosenbloom

Abstract. One problem which frequently surfaces when applying explanation-based learning (EBL) to imperfect theories is the *multiple inconsistent explanation problem*. The multiple inconsistent explanation problem occurs when a domain theory produces multiple explanations for a training instance, only some of which are correct. Domain theories which suffer from the multiple inconsistent explanation problem can occur in many different contexts, such as when some information is missing and must be assumed: since such assumptions can be incorrect, incorrect explanations can be constructed. This paper proposes an extension of explanation-based learning, called *abductive explanation-based learning* (A-EBL) which solves the multiple inconsistent explanation problem by using set covering techniques and negative examples to choose among the possible explanations of a training example. It is shown by formal analysis that A-EBL has convergence properties that are only logarithmically worse than EBL/TS, a formalization of a certain type of knowledge-level EBL; A-EBL is also proven to be computationally efficient, assuming that the domain theory is tractable. Finally, experimental results are reported on an application of A-EBL to learning correct rules for opening bids in the game of contract bridge given examples and an imperfect domain theory.

Keywords. Explanation-based learning, theory revision, theory specialization, probably approximately correct learning

1. Introduction

1.1. Motivation

An important problem in machine learning is that of extending explanation-based learning (EBL) methods to domain theories which are imperfect: i.e., theories which are either incomplete or incorrect. A solution to this problem would be an important step toward integration of explanation-based and similarity-based approaches to learning. One difficulty which frequently surfaces when appying EBL to imperfect theories is the *multiple inconsistent explanation problem*, also called the *multiple explanation problem* (Fawcett, 1989; Pazzani, 1988; Rajamoney & DeJong, 1988). The multiple inconsistent explanation problem occurs when a domain theory produces multiple explanations for a training instance, only some of which are correct. Domain theories which suffer from the multiple explanation problem can occur in many different contexts.

Incomplete domain theories. One situation in which the multiple explanation problem occurs is when some information is missing and must be assumed. For instance, in domain theories which involve some element of plan recognition, the goals of one or more

agents are typically unknown and must be assumed (Pazzani, 1988). Often, there are several plausible assumptions which would suffice to explain an action, but only explanations based on the correct assumption will be valid. The multiple explanation problem also arises when the domain theory is missing inference rules and some external mechanism (Fawcett, 1989; Hall, 1988; Roy & Mostow, 1988; Mahadevan, 1989) is used to supply these rules as needed. Typically, the newly-supplied inference rules are not necessarily valid, and hence invalid explanations can be constructed.

Incorrect domain theories. Finally, the multiple explanation problem can occur when the domain theory is incorrect because it contains one or more over-general rules. This may be a result of flaws in whatever knowledge acquisition system was used to acquire the domain theory. In other circumstances, the imperfections in the domain theory may be deliberate. For instance, an incorrect but tractable approximation to an intractable domain theory may be used, as in Rajamoney and DeJong (1988); or, EBL may be used inductively (as in Carbonell et al., (1987); Flann and Dietterich (1989); and Hirsh (1990)) to find some particular specialization of an over-general domain theory.

Standard explanation-based systems cannot handle the multiple explanation problem. Typically, they either assume that there will be only a single explanation for each instance, or that all explanations will be valid. Existing approaches to the multiple explanation problem have relied on additional mechanisms to either evaluate (as in Fawcett (1989) and Pazzani (1988)) or validate (as in Rajamoney and DeJong (1988)) the various explanations of a training example. However, the evaluation metrics that are proposed in Fawcett (1989) and Pazzani (1988) have not been rigorously justified, and are to some extent domain specific; the validation technique described in Rajamoney and DeJong (1988) requires the ability to interact with the outside world via experimentation.

There are two contributions of this paper. The first contribution is to formalize a certain type of knowledge-level EBL called *explanation-based learning for theory specialization*, or EBL/TS. The second and more important contribution is to propose and validate an extension of EBL/TS called *abductive explanation-based learning* (A-EBL) which addresses the multiple explanation problem. In contrast to the techniques mentioned above, A-EBL is a general technique which does not require the ability to conduct experiments and which makes no assumptions about the domain theory beyond those made by EBL/TS. A-EBL does, however, require additional information in the form of negative examples.

A-EBL uses set covering techniques and negative examples to select a minimal consistent subset of the set of possible explanations of a training example. We show by formal analysis that selection of this subset leads to good convergence properties: in the worst case, A-EBL converges only logarithmically slower than EBL/TS. The number of examples needed for A-EBL to produce a good approximation to an unknown concept increases only with the complexity of the unknown concept, not with the number of explanations per training example. A-EBL is also shown to be computationally efficient, assuming a tractable domain theory; execution time increases fairly slowly as the number of explanations per training example increases. A-EBL has also been experimentally validated, although it has not been systematically tested on all of the types of domain theories described above.

A final desirable feature of A-EBL is that it is, in many senses, a natural extension of standard EBL. If A-EBL is invoked in a situation in which each instance has only a single explanation, the result will be exactly the same as for EBL/TS. Moreoever, the conditions

necessary for A-EBL to "work" (in a formal sense which we will introduce in Section 3) are less restrictive than the conditions necessary for EBL/TS to work.

This paper is an expanded version of Cohen (1990d); Section 3.1 also draws on results presented in Cohen (1990a).

1.2. Overview

Before presenting A-EBL, we would like to clearly state the problem that it is intended to solve. It is not sufficient to merely say that the goal is to solve the problem solved by "standard EBL systems" in the presence of multiple explanations because there are many different kinds of EBL systems. Therefore, we begin by defining the "theory specialization problem" and presenting a standard algorithm for solving this problem *without* multiple explanations. This algorithm, called EBL/TS, formalizes a particular type of inductive EBL system. It will serve as a sort of yardstick against which the A-EBL algorithm will be measured.

We then present the A-EBL algorithm. We show that A-EBL is no more restrictive than EBL/TS, and also show that the worst-case asymptotic behavior of A-EBL is acceptable. More precisely, we show that when examples are selected stochastically, both algorithms converge using a sample size polynomial in the complexity of the concept being learned and the accuracy of the hypothesis produced. In short, we present a comparative analysis of A-EBL and EBL/TS viewed as *probably approximately correct* learning algorithms (Blumer et al., 1986; Valiant, 1984). The analysis shows that the sample complexity of A-EBL is within a logarithmic factor of that of EBL/TS according to this criterion.

The formal analysis gives a rigorous understanding of the asymptotic behavior of the algorithm given stochastically presented examples. In addition to experimentally evaluating A-EBL in this context, we also experimentally evaluate the behavior of the algorithm given a *small* set of examples provided by a friendly and informative teacher who does *not* know the internal workings of the learning algorithm. In particular, we use the examples given in a well-known introductory book on playing bridge (Sheinworld, 1964) to test the algorithm's behavior on several domain theories which are "imperfect" in slightly different ways. In addition to helping us evaluate A-EBL's behavior given small samples, this section also serves as an extended example of what sorts of imperfections in a domain theory can be corrected. We conclude by discussing related work, identifying further research problems relating to A-EBL, and summarizing our results.

In this paper, we will assume that the domain theory is a Horn clause theory. We will also assume that it is *tractable*, in a strong sense: we assume that all proofs of a goal can be generated in time polynomial in the size of that goal. We emphasize that this is a fairly strong assumption about the domain theory; many Horn clause theories produce an exponential (or even infinite) number of proofs.

2. Abductive EBL

For the reasons discussed above, we preface our discussion of A-EBL with the definition of a particular problem (which we call the theory specialization problem) that is solvable

by EBL, and also a definition of a particular EBL algorithm (which we call EBL/TS) that solves this problem.

2.1. The theory specialization problem

Flann and Dietterich (1989) have observed that explanation-based generalization (EBG), although often used to improve performance of a problem solver, can also be used to acquire new knowledge. The input of EBG is a domain theory T defining some concept C_T and a training example that is an instance of C_T; the output of EBG is a rule R that generalizes the training example. Since the new rule is in the deductive closure of T, it must be that anything recognized by that rule is also recognized by T as belonging to C_T; in other words, it must be that R encodes *sufficient* conditions for membership in C_T.

$$R(x) \Rightarrow C_T(x)$$

Alternatively, one could interpret R to be *necessary and sufficient* conditions for membership in a new concept C, of which the training example is also a member.

$$R(x) \equiv C(x)$$

This is an inductive leap, since it is not shown that R is also a necessary condition for C. In this case, EBG is being used as a mechanism for generating a specialization of the original domain theory. Many of the examples in the literature on EBL have this character: for instance, in Section 7 of DeJong and Mooney (1986) a theory of social interactions is specialized to form a concept roughly corresponding to "suicide."

However, this is a somewhat oversimplified model of EBL; many EBL systems that perform specialization learn a set of rules, rather than a single rule. For example, LEAP (Mahadevan, 1985) can be viewed as learning the specialization "useful transformation from a specification to a design artifact" given a domain theory for the related concept "legal transformation from a specification to a design artifact;" the specialization is encoded by a *set* of useful transformation rules. In this case independent applications of EBL are used to produce a series of rules $R_1, \ldots, R_n$, and the disjunction of these rules defines the new concept C:

$$R_1(x) \vee \ldots \vee R_n(x) \equiv C(x)$$

Since the set of rules $R_1, \ldots, R_n$ is a new theory that is less general than the original domain theory, this use of EBL will be called "theory specialization." The *theory specialization problem* is to correct an over-general domain theory by finding a specialization of that theory which accurately models the world. Adopting the terminology of Mitchell et al., (1986), the theory specialization problem can be stated more precisely as follows.[1]

Definition 1 *The* theory specialization problem *(TSP) is defined as follows.*

- *Given:*
  - *A* domain theory *T defining a base concept* C_T *which is a superset of the concept to be learned.*
  - *An* operationality criterion $\mathcal{O}$ *indicating what predicates from T can be used in the learned concept definition.*
  - *A* sample S^+ *of positive examples of an unknown concept C.*
  - *A* sample S^- *of negative examples of an unknown concept C, (optional).*
  - *A* symbol *to use as the name of the unknown concept C.*
- *Find:*
  - *A domain theory T' defining the unknown concept C.*

Notice that the examples are not only examples of the concept C_T defined by the domain theory, but are also examples of the unknown concept *C*.

The theory specialization problem is closely related to the problem of "theory-based concept specialization" as defined in Flann and Dietterich (1989).[2] Further motivations for study of the theory specialization problem, and other techniques for solving it, can be found in Carbonell et al., (1987); Drastal et al., (1989); Ginsberg (1988); Hirsh (1988); Laird (1988); and Rosenbloom and Aasman (1990).

In all of our examples of theory specialization, the rules of the specialized theory T' cannot call one another. This means that the concept *C* is defined as the disjunction of several conjunctively described sets (the sets defined by $R_1, \ldots, R_n$). Also notice that we are not excluding the possibility that additional constraints—for instance, efficiency constraints—might be placed on the specialization. In particular, constraining the operationality predicate to only be true on predicates which are "efficiently evaluable" is not disallowed, although it is not required.

As an example of a theory specialization problem, consider the domain theory shown in Figure 1. This is a simplified domain theory for opening bids in the game of contract bridge.[3] The first rule states that given a hand of *opening-strength* that has two *biddable* suits, one should bid the suit that is *preferred*. The next few rules state that any 4, 5, or 6 card suit is *biddable*; that a hand that is a *weak-opener* or a *strong-opener* is of *opening-strength*; and finally, that given two biddable suits, one should *prefer* either the longer suit (the one with most cards) or the higher suit. (In bridge, the "highest" suit is spades, preceded by hearts, diamonds, and clubs.) All predicates used but not defined in Figure 1 are operational.

This domain theory is a simplification of a more realistic theory manually extracted from a textbook on contract bridge. As in the theory that it simplifies, the rules for the *prefer* predicate in this domain theory are over-general: in particular, the conditions under which one should choose the longer suite and/or the higher suit are not fully specified. The inaccuracy of the theory is reflected by the name of the concept defined by the theory: bids suggested by the theory are plausible but not necessarily correct.

A corrected version of the theory of Figure 1 is shown in Figure 2. The theory of Figure 2 is a specialization of the theory of Figure 1; it specializes the *prefer* predicate by stating that one should prefer the higher suit only on hands with two suits of equal length or on weak hands with 5-4 distribution, and the longer suit in other cases. The rest of the theory

C_T=*plausible-bid(H,S)*

opening-strength(H) ∧ biddable(H,S1) ∧ biddable(H,S2) ∧ prefer(H,S1,S2) ⇒ plausible-bid(H,S1)
4cards(H,S) ⇒ biddable(H,S)
5cards(H,S) ⇒ biddable(H,S)
6cards(H,S) ⇒ biddable(H,S)
strong-opener(H) ⇒ opening-strength(H)
weak-opener(H) ⇒ opening-strength(H)
longer(H,S1,S2) ⇒ prefer(H,S1,S2)
higher(S1,S2) ⇒ prefer(H,S1,S2)

Figure 1. Domain theory for opening bids.

C=correct-bid(H,S)

opening-strength(H) ∧ biddable(H,S1) ∧ biddable(H,S2) ∧ prefer(H,S1,S2) ⇒ correct-bid(H,S1)
6cards(H,S) ⇒ biddable(H,S)
5cards(H,S) ⇒ biddable(H,S)
4cards(H,S) ⇒ biddable(H,S)
strong-opener(H) ⇒ opening-strength(H)
weak-opener(H) ⇒ opening-strength(H)
higher(S1,S2) ∧ 4cards(H,S1) ∧ 4cards(H,S2) ⇒ prefer(H,S1,S2)
higher(S1,S2) ∧ 5cards(H,S1) ∧ 5cards(H,S2) ⇒ prefer(H,S1,S2)
higher(S1,S2) ∧ 6cards(H,S1) ∧ 6cards(H,S2) ⇒ prefer(H,S1,S2)
higher(S1,S2) ∧ 5cards(H,S1) ∧ 4cards(H,S2) ∧ weak-opener(H) ⇒ prefer(H,S1,S2)
longer(H,S1,S2) ∧ 5cards(H,S1) ∧ 4cards(H,S2) ∧ strong-opener(H) ⇒ prefer(H,S1,S2)
longer(H,S1,S2) ∧ 6cards(H,S1) ∧ 4cards(H,S2) ⇒ prefer(H,S1,S2)
longer(H,S1,S2) ∧ 6cards(H,S1) ∧ 5cards(H,S2) ⇒ prefer(H,S1,S2)

Figure 2. Accurate theory for opening bids.

is essentially identical to the theory of Figure 1; the only additional change is that the name of the top-level concept has been changed from *plausible-bid* to *correct-bid*.

In Figure 2, the theory of Figure 1 has been specialized by explicitly adding conditions to certain rules. EBG does not explicitly add such conditions; however, under certain circumstances the effect can be the same. The next section will describe an algorithm that specializes a theory by repeatedly using EBG.

2.2. The EBL/TS algorithm

Crucial to explanation-based generalization is the notion of an *explanation structure*, which is a generalized version of the proof of a training example. Examples and elaboration of these definitions can be found in Mitchell et al., (1986).

Definition 2 *Let p_x be an AND-OR proof for the training example x.*

- An abstract proof *for x is obtained by pruning from the proof tree p_x every subtree with a root which is operational (that is, a root on which the operationality predicate succeeds.)*[4]
- *An* explanation structure *for a proof p_x is obtained by replacing every instantiated rule used in the proof p_x by the associated general rule.*
- *An* abstract explanation structure *for a proof p_x is obtained by pruning from the explanation structure every subtree with a root which is operational.*

Using these constructs, the operation of explanation-based generalization, and hence algorithms which use it, can be easily specified. Figure 3 describes the EBL/TS learning algorithm, which uses EBG to solve the TS problem, using a set of examples of the specialized theory. The basic idea is very simple: the EBG algorithm is used to generalize each example, and the union of these generalizations is returned as a hypothesis. For completeness, we also present our algorithm for explanation-based generalization. Note that we define EBG as a procedure for generalizing a *proof*, rather than an instance. This distinction has not been important before, because previous systems assumed a single proof for each instance.

As an example, consider the operation of EBL/TS given the domain theory of Figure 1 and the examples of Table 1. In this example, whether it is appropriate to bid the higher of two suits or the longer of two suits depends on two things: the strength of the hand, which is reflected in the structure of the proof of the subgoal *opening-strength*; and the nature of the two suits under consideration, which is reflected in the structure of the proofs of the two *biddable* subgoals. This means that the over-general *prefer* predicate can be correctly specialized simply by embedding it in an EBL rule: the context in which the predicate is used serves to specialize it. *Therefore, even though the domain theory itself is over-general, generalizations of proofs constructed using the domain theory are* not *over-general.* Thus in this case, EBL/TS can be used to correct the domain theory, without explicitly refining the over-general rules that it contains; instead, EBL/TS *implicitly* refines overgeneral rules by embedding them in macro-rules.

For instance, given the domain theory and operationality predicate of Figure 1, the training examples of Table 1, and the symbol *correct-bid* to use as the name of the unknown concept, EBL/TS will output the following set of rules. These rules define a specialization of the original concept of *plausible-bid*.

strong-opener(H) ∧ 6cards(H,S1) ∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 6cards(H,S1) ∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 6cards(H,S1) ∧ 4cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 5cards(H,S1) ∧ 4cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)

This set of rules does not cover all of the cases covered by the theory of Figure 2; however, given enough examples, EBL/TS will find a set of rules which together are equivalent to the theory of Figure 2. In this case, a total of twelve rules (shown in Figure 4) are needed. Notice that the rules generated by EBG have not been added to the original domain theory as macro-rules, but have been used to construct a definition of a new concept different from the concept defined by the original domain theory; thus EBL/TS is a sort of knowledge-level learning (Dietterich, 1986).

Algorithm $EBL/TS(T, \mathcal{O}, S^+, N)$:

Inputs:
a domain theory T defining a concept C_T
an operationality predicate $\mathcal{O}$
a set S^+ of positive examples of an unknown concept C
a name N to use for the unknown concept
Output:
a hypothesis H for the unknown concept C
begin
$H \leftarrow \emptyset$
for each positive example $x \in S$ **do**
if $x \notin H$ **then**
$p_x \leftarrow$ the proof that $x \in C_T$
$R \leftarrow EBG(T, \mathcal{O}, p_x)$
$H \leftarrow H \cup \{R\}$
endif
endfor
return rename(N, H)
end

subroutine $EBG(T, \mathcal{O}, p_x)$:

Inputs:
a domain theory T and an operationality predicate $\mathcal{O}$
a proof p_x of an instance x
Output:
a description of a set G such that $\{x\} \subseteq G \subseteq C_T$,
where C_T is the concept defined by the domain theory
begin
$a_x \leftarrow$ the abstract explanation structure for p_x
$LEAVES \leftarrow \{L : L$ is a leaf of $a_x\}$
return the rule whose consequent is the root of a_x and whose antecedent is $\bigwedge_{L \in LEAVES} L$
end

subroutine rename(N, H):

begin
return a copy of R in which the principle functor of the consequent of each rule
has been replaced with the symbol N
end

Figure 3. Algorithms for EBL/TS and EBG.

Table 1. Example of correct bids.

	Hand	Correct Bid	Features
1	♠ KQJ74 ♡ K2 ♢ AQJ987 ♣ –	diamond	strong-opener
2	♠ AQ987 ♡ KQJ742 ♢ 32 ♣ –	heart	weak-opener
3	♠ 32 ♡ AQ98 ♢ KQ9742 ♣ 9	diamond	weak-opener
4	♠ 3 ♡ AQ97 ♢ J98 ♣ KQ963	heart	weak-opener

C=correct-bid(H,S)

strong-opener(H) ∧ 5cards(H,S1) ∧ 4cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
strong-opener(H) ∧ 6cards(H,S1) ∧ 4cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
strong-opener(H) ∧ 6cards(H,S1) ∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 6cards(H,S1) ∧ 4cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 6cards(H,S1) ∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)

strong-opener(H) ∧ 4cards(H,S1) ∧ 4cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)
strong-opener(H) ∧ 5cards(H,S1) ∧ 5cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)
strong-opener(H) ∧ 6cards(H,S1) ∧ 6cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 4cards(H,S1) ∧ 4cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 5cards(H,S1) ∧ 5cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 6cards(H,S1) ∧ 6cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 4cards(H,S1) ∧ 5cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)

Figure 4. Theory for opening bids converged to by EBL/TS.

In contrast to the definition of EBL presented in Mitchell et al., (1986), EBL/TS relaxes the assumption that the domain theory is complete and correct. Instead, it is assumed that the theory is incorrect in a particular way. The theory must encode necessary but not sufficient conditions for membership in the concept to be learned; additionally, the rules formed by EBG must be specific enough so that they do not over-generalize an instance *relative to the unknown concept C.* The latter condition may not be met if the level of operationality is too high, or if there are properties relevant to membership in *C* that are not reflected in the domain theory. Thus, EBL/TS *cannot* be used with an arbitrary domain theory and operationality predicate; since some engineering of the domain theory and operationality predicate is usually needed, EBL/TS should perhaps be viewed as a partial solution to the theory specialization problem.

The question of precisely when EBL/TS and similar algorithms can be used to solve a theory specialization problem is discussed in more detail in Section 3.1.

2.3. A domain theory generating multiple explanations

One problem with EBL/TS is illustrated by the following bridge bid.

	Hand	Bid	Features
5	♠ AQJ987 ♡ K2 ◇ KQJ74 ♣ –	spade	strong-opener

There are *two* possible explanations (shown in Figure 5) as to why this is a plausible bid: the spade suit might have been preferred because it is the longer suit or because it is the higher suit. However, only one of these explanations (the first one) is correct with respect to the intended specialization of the theory shown in Figure 2. Since EBL/TS has no way of choosing between these two explanations, it must either discard this example or run a risk of over-generalizing by making an arbitrary choice.

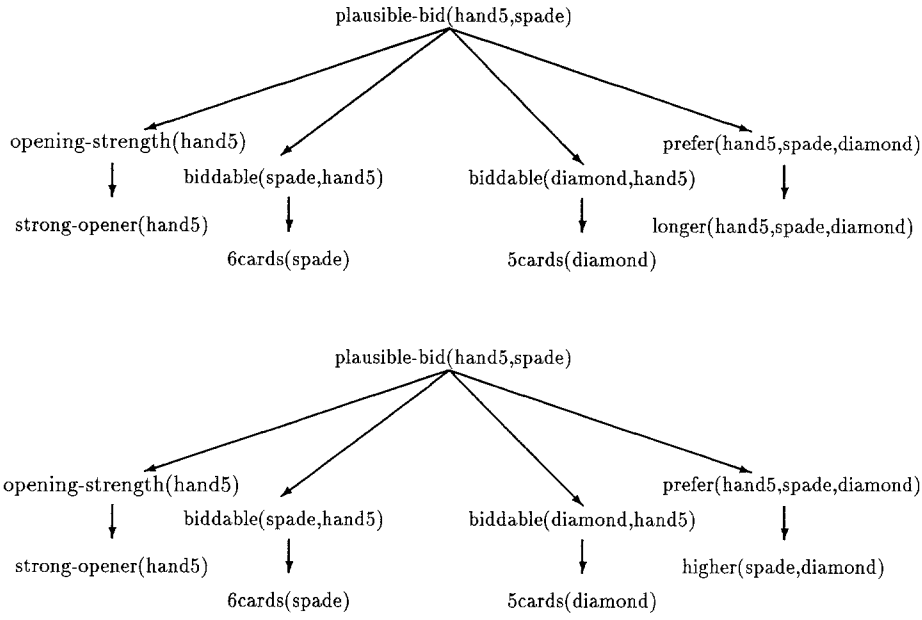

Figure 5. Possible explanations for hand 5.

The strategy of discarding examples that have multiple explanations is not always feasible: with some domain theories *all* examples will have multiple explanations. Consider the domain theory of Figure 6. This theory is a partial definition of the term *weak-opener*, which is used in the theory of Figure 1. Again, this theory is a simplification of one manually extracted from a textbook.

The first rule of the theory states that a hand is a *weak-opener* if it has ten or eleven high card points, two or more quick tricks and "good length in the major suits." In the textbook, high card points and quick tricks are precisely defined, but "good length in the major suits" is not defined: instead, interpretation of this phrase is left to the reader. One reasonable interpretation of this phrase is "having length in the major suits greater than

C=weak-opener(H)

high-card-points(H,HCP) $\wedge$ (HCP $\geq$ 10) $\wedge$ (HCP $\leq$ 11) $\wedge$ quick-tricks(H,QT) $\wedge$ (QT $\geq$ 2)
$\wedge$ good-length-in-major-suits(H) $\Rightarrow$ weak-opener(H)

length-in-major-suits(H,L) $\wedge$ minimum-good-length(Th) $\wedge$ (L $\geq$ Th) $\Rightarrow$ good-length-in-major-suits(H)

minimum-good-length(0)
minimum-good-length(1)
$\vdots$
minimum-good-length(13)

Figure 6. Domain theory for *weak-opener*.

some threshold length *Th*;" the second rule in the theory reflects this interpretation. These two rules are an incomplete definition of *weak-opener*; to complete the definition, we need to find a correct definition of the predicate *minimum-good-length*.

Notice, however, that the threshold length *Th* is bounded by thirteen (the number of cards in a bridge hand) and is integral; hence there are a finite number of possible definitions of the undefined predicate. In Figure 6, the theory has been completed by adding *all* of the possible definitions of the predicate *minimum-good-length*. The motivation for adding all possible completions to the theory is that now using this theory with a standard backchaining theorem prover has a similar effect to using the incomplete version of the theory with an *abductive theorem prover*. Abductive reasoning is, intuitively, reasoning from effects to possible causes (in contrast to deductive reasoning, which can be thought of as reasoning from causes to effects.) Abductive reasoning is often performed by using a backchaining theorem prover that has been augmented with the ability to make assumptions in order to complete a proof; often the assumptions are taken from some predetermined class.

For example, given the observation that *weak-opener (♠32 ♡AQ98 ◇KQ9742 ♣9)* is true, an abductive theorem prover might find the proof shown in Figure 7 by assuming that the *minimal-good-length* for a hand is three; this is of course only one of several possible assumptions that could be made to complete the proof. If all possible assumptions are added as axioms, as in Figure 6, then a standard backchaining theorem prover will produce the same set of proofs that would be produced by an abductive theorem prover.[5] Again, notice that only one of these many proofs (the one that assumes the correct threshold value) will be correct.

Since different assumptions will lead to different proofs, an abductive theorem prover will normally generate many different proofs. In general, if there are k points where the theory is incomplete, and for each of these points there are n possible assumptions that might complete the theory, there can be up to n^k different proofs for an example. Since all of the proofs generated by the abductive theorem prover will also be generated by a standard backchaining theorem prover using the theory of Figure 6, for such theories, *every example will have many alternative proofs*. Some mechanism is clearly needed to choose among these alternative inconsistent explanations. Such abductive theories have been described several times in the literature on EBL (Pazzani, 1988; O'Rorke et al., 1989), and are an important motivation for studying the multiple explanation problem.

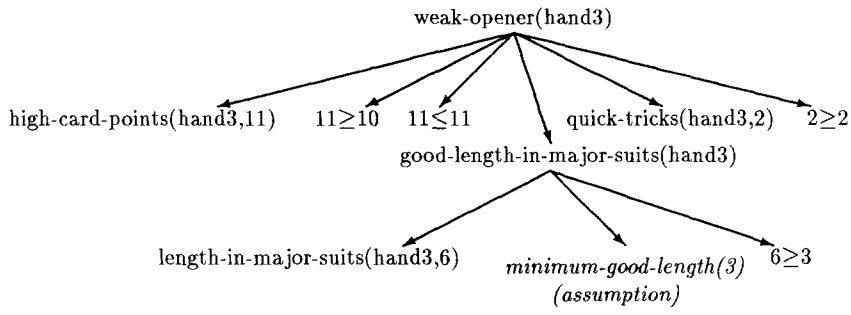

Figure 7. One possible proof for *weak-opener (♠32 ♡AQ98 ◇KQ9742 ♣9)*.

2.4. The abductive-EBL algorithm

How can the multiple inconsistent explanation problem be solved? We will constrain the extended learning algorithm to return as output a disjunction of sets, each of which is the generalization of some proof of a training example. In other words, we require the output to be of the form

$$C(x) = R_1(x) \cup R_2(x) \cup \ldots \cup R_n(x)$$

where each R_i is found by applying the *EBG* algorithm to some proof of some training example x. Since we assume that all the possible proofs for a training example can be generated in a reasonable amount of time—in other words, that all the possible candidate generalizations R_i can be easily generated—the main question seems to be: which of the many possible generalizations R_i should be included in the concept C?

One possible choice is to pick the set of generalizations that accounts for all of the training examples, but which has the simplest description. If S^+ is the set of all positive examples of the concept C, A-EBL tries to find the smallest set of rules $\{R_1, \ldots, R_n\}$ such that $\cup R_i \supseteq S^+$, where "smallest" means having the simplest description. Biases towards simple representations are common in machine learning systems; it will be shown in Section 3.2 that in this case, this bias leads to a system that learns simple concepts relatively quickly.

To satisfy this bias (i.e., to find the smallest set $\{R_1, \ldots, R_n\}$ such that $\cup R_i \supseteq S^+$) one must solve the *minimum set cover problem*, which is known to be NP-complete. Fortunately, however, good heuristic solutions to the set cover problem exist. One such heuristic solution is the *greedy set cover algorithm*, first described in Johnson (1974) and extended in Chvátal (1979). This algorithm simply repeatedly adds to the set cover that rule R_i which maximizes the ratio of the number of as-yet-uncovered examples in R_i to the size of the description of R_i. The greedy set cover is guaranteed to find a cover of size no greater than $n \log m$, where n is the size of the description of the smallest cover, and m is the number of examples to be covered. In this problem, n is the size of the description of the unknown concept, and m is the number of examples.

The size measure may be defined arbitrarily: the only constraint is that the size of a description of a cover be the sum of the sizes of the descriptions of the component sets. In A-EBL, the size measured for generalizations is simply the number of nodes in the associated abstract explanation structure. This is a heuristic estimate of the complexity of a generalization. It was chosen partially because of its analytic tractability, and partially for its ease of implementation; in particular, using this measure makes the arguments of Section 3.2 much cleaner, and also makes possible the program optimizations discussed in Section 2.6.

The greedy set cover algorithm is a reasonable approach to generating a hypothesis for the unknown concept C. However, our motivating example of bridge bidding, and also formal analysis (see Section 3) suggests that some *negative* information is also required to ensure convergence. Fortunately, negative information is easy to incorporate into our learning algorithm. If we are given a negative example—i.e., some element $x \notin C$—then we know that no R_i containing x is a disjunct of C. So these R_i's can be eliminated from consideration by the set cover algorithm.

Algorithm *A-EBL*$(T, \mathcal{O}, S^+, S^-, N)$:

Inputs:
a domain theory T defining a concept C_T
an operationality predicate $\mathcal{O}$
a set S^+ of positive examples of an unknown concept C
a set S^- of negative examples of an unknown concept C
a name N to use for the unknown concept
Output:
a hypothesis H for the unknown concept C
begin
$RS \leftarrow \{R_i : consistent(R_i, S^-) \wedge R_i = EBG(T, \mathcal{O}, p_x)$
for some proof p_x of some positive example $x \in S^+\}$
$H \leftarrow$ greedy-set-cover(RS, S^+)
return rename(N, H)
end

Algorithm greedy-set-cover(RS, S^+):

begin
$RS' \leftarrow$ a minimal subset of RS such that $\bigcup_{R_i \in G'} R_i \supseteq S^+$
return RS'
end

Algorithm consistent(R_i, S^-):

begin
return true iff $\neg\exists$ a negative example $x \in R_i$
end

Figure 8. High-level description of the A-EBL algorithm.

Putting this all together, we arrive at the algorithm of Figure 8. This algorithm is the main result of this paper. It starts out by explictly computing all the possible generalizations of every positive example. It then finds a hypothesis for the unknown concept C by using the greedy set cover algorithm to find a cover of the set of positive examples, using the set of consistent generalizations of positive examples as candidate elements of the cover; a generalization is *consistent* if it does not also cover some negative example.

2.5. An example

This section will present a simple example of the operation of A-EBL. The inputs are the theory and operationality predicate shown in Figure 1, the examples shown in Table 2, and the name *correct-bid* for the unknown concept. Notice that in addition to examples of correct bids, some examples of plausible but incorrect bids are also given; these incorrect bids are negative examples.

Table 2. Examples of correct and incorrect bids.

	Hand	Correct Bid	Features
(a)	♠ AQJ987 ♡ K2 ◇ KQJ74 ♣ –	spade	strong-opener
(b)	♠ KQJ74 ♡ K2 ◇ AQJ987 ♣ –	diamond	strong-opener
(c)	♠ 3 ♡ AQ97 ◇ J98 ♣ KQ963	heart	weak-opener
(d)	♠ 3 ♡ KQ963 ◇ J98 ♣ AQ97	heart	weak-opener

	Hand	Incorrect Bid	Features
(c)	♠ 3 ♡ AQ97 ◇ J98 ♣ KQ963	club	weak-opener
(d)	♠ 3 ♡ KQ963 ◇ J98 ♣ AQ97	club	weak-opener

The first step of A-EBL is to compute the set of possible generalizations of each positive example. Hands (a) and (d) have two possible explanations each, so there are a total of six generalizations of the four positive examples; these generalizations are shown below. Of these rules, rule (d-2) is the only inconsistent rule; it covers the negative example associated with hand (c).

(a-1) strong-opener(H) ∧ 6cards(H,S1)
∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ plausible-bid(H,S1)

(a-2) strong-opener(H) ∧ 6cards(H,S1)
∧ 5cards(H,S2) ∧ higher(S1,S2) ⇒ plausible-bid(H,S1)

(b) strong-opener(H) ∧ 6cards(H,S1)
∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ plausible-bid(H,S1)

(c) weak-opener(H) ∧ 4cards(H,S1)
∧ 5cards(H,S2) ∧ higher(S1,S2) ⇒ plausible-bid(H,S1)

(d-1) weak-opener(H) ∧ 5cards(H,S1)
∧ 4cards(H,S2) ∧ higher(S1,S2) ⇒ plausible-bid(H,S1)

(d-2) weak-opener(H) ∧ 5cards(H,S1)
∧ 4cards(H,S2) ∧ longer(H,S1,S2) ⇒ plausible-bid(H,S1)

The next step is to find a small subset of the consistent rules that cover the set of positive examples S^+ using the greedy set cover algorithm. The first rule that is added to the cover is the rule R_1 that maximizes the ratio

$$\frac{|R_1 \cap (\text{uncovered examples in } S^+)|}{size(R_1)}$$

As it turns out, all of the rules in this example are of size nine. Rules (a-1), (b), (c), and (d-1) cover two examples each, and rule (a-2) covers one example. Therefore the greedy set cover will add one of the rules (a-1), (b), (c), or (d-1) to the cover; let us assume that the rule added is rule (a-1).

Now the positive examples associated with hands (a) and (b) have been covered. Rules (a-1), (a-2) and (b) thus cover no as-yet-uncovered examples, and hence the ratio of

$$\frac{|R \cap (\text{uncovered examples in } S^+)|}{size(R_1)}$$

for these rules is 0/9 = 0. The rules (d-1) and (c) both cover the remaining two examples, and hence one of them will be added to the set cover. After renaming the rules, the final output of A-EBL is the theory below.

strong-opener(H) ∧ 6cards(H,S1) ∧ 5cards(H,S2) ∧ longer(H,S1,S2) ⇒ correct-bid(H,S1)
weak-opener(H) ∧ 5cards(H,S1) ∧ 4cards(H,S2) ∧ higher(S1,S2) ⇒ correct-bid(H,S1)

Notice that the greedy set cover algorithm, and hence A-EBL, incorporates several heuristics for choosing between explanations. First, it chooses explanations that (like the explanation associated with rule (a-1)) explain several positive examples over explanations that explain a single positive example. Second, it rejects explanations that (like the explanation associated with rule (d-2)) are shown to be inconsistent by negative data. Finally, it prefers simple explanations to more complex ones; this preference is not demonstrated by this simple example.

2.6. Optimizations of the algorithm

Figure 8 is a high-level description of the A-EBL algorithm. In fact, our implementation of A-EBL incorporates some optimizations which improve the algorithm's run-time without affecting its correctness. The goal of these optimizations is to replace membership tests in the sets R_i, which tend to be slow, with comparison operations between abstract explanation structures, which can be done efficiently.

- First, we explicitly compute, not the set of generalizations of each example, but the set of abstract explanation structures for the proofs of each example. These abstract explanation structures are stored as trees whose nodes are labeled with the *names* of domain theory clauses,[6] rather than the clauses themselves. This makes comparison operations relatively fast; it also means that we can delay the operation of converting the abstract explanation structures to rules (an operation which we call *peval*, for partial evaluation), and perform it only on those rules which are actually part of our hypothesis.
- Second, we represent each example x internally as the set of abstract explanation structures associated with the explanations of x, and represent each generalization internally as the abstract explanation structure from which that generalization is derived. To test if $x \in R_i$, it is now sufficient to test whether the abstract explanation structure representing R_i is in the set of abstract explanation structures representing x.

Code incorporating these changes, and also giving the details of the greedy set cover algorithm, is given in Figure 9.

Algorithm *A-EBL*$(T, \mathcal{O}, S^+, S^-, N)$:

Inputs:
a domain theory T defining concept C_T
an operationality predicate $\mathcal{O}$
a set S^+ of positive examples of an unknown concept C
a set S^- of negative examples of an unknown concept C
a name N to use for the unknown concept
Output:
a hypothesis H for the unknown concept
begin
$H \leftarrow \emptyset$
(POS and NEG are each sets of sets of abstract explanation structures)
$POS \leftarrow \emptyset$
$NEG \leftarrow \emptyset$
(precompute the explanation structures for all the examples)
for each positive example x **do**
add to POS the set of all abstract explanation structures for x
for each negative example x **do**
add to NEG the set of all abstract explanation structures for x
(find the minimal set cover)
while $POS \neq \emptyset$ **do**
$a_x \leftarrow$ optimal-consistent-explanation-structure(POS, NEG)
$H \leftarrow H \cup \{\text{peval}(a_x)\}$
$POS \leftarrow POS - \{p \in POS : a_x \in p\}$
endwhile
return rename(N, H)
end

Algorithm optimal-consistent-explanation-structure(POS, NEG):

begin
return the abstract explanation structure a_x such that
$\neg\exists n \in NEG : a_x \in n$ and $\frac{|\{p \in POS : a_x \in p\}|}{size(a_x)}$ is maximal
end

Algorithm peval(a_x):

see Appendix B

Figure 9. Implemented algorithm for A-EBL.

3. Formal analysis of A-EBL

In Section 4 we will return to the domain of bridge bidding and investigate the performance of A-EBL on a larger and more realistic version of the bridge bidding problem. Questions will doubtless remain, however, regarding the generality of the technique.

A-EBL learns an unknown concept using a particular sort of domain-theory induced bias: under what circumstances is this bias appropriate? A second question concerns how well this technique will scale up: can A-EBL be applied equally well to larger learning tasks, with more data?

To answer the first question, it is necessary to characterize the types of concepts for which EBL/TS and A-EBL are "competent"—that is, to characterize the class of concepts for which A-EBL's inductive bias is not too strong. This sort of analysis is usually not necessary for an inductive learner, because usually it is clear what sorts of hypotheses can be formed, and hence when the bias is not too strong; however, since A-EBL's inductive bias is derived in part from the domain theory, the class of concepts that it can hypothesize is not immediately obvious. A complete characterization is beyond the scope of this paper; however, we do give a relative characterization of the competence of A-EBL and EBL/TS. The characterization shows that A-EBL is applicable in a broader range of circumstances than EBL/TS.

To answer the second question, we will adopt the pac-learnability model, a formal model of learnability. Our calculations will show that in the worst case, the number of examples needed by A-EBL to learn a theory specialization which has size n is only polynomial in n. Moreover, the number of samples needed by A-EBL is only worse by a logarithmic factor from the corresponding figure for EBL/TS. We also analyze the run-time of the algorithm, to verify that it is efficient. The results show that, excluding the time spent in theorem-proving, our implementation runs in time $O(n \log m \cdot P[N + P])$ where n is the size of the unknown concept, m is the number of training examples, P is the number of proofs of positive examples, and N is the number of proofs of negative examples.

3.1. Competence results

3.1.1. Analysis of EBL/TS and A-EBL

Let a *sample* of an unknown concept C be a pair of sets S^+, S^-, where $S^+ \subseteq C$ and $S^- \subseteq \overline{C}$. We define "competence" as follows.

Definition 3 *An algorithm L is* competent *for C if and only if for every sample of C, L can output a hypothesis that is consistent with the sample.*

Intuitively, an algorithm L is "competent for C" unless its inductive bias is too strong to learn C. The motivation for this particular definition of competence is that normally one detects that the bias of an inductive learner is too strong only when the learner fails to find a hypothesis consistent with some sample.

If L is competent for C, then it immediately follows that L must be able to hypothesize C (or the description of some equivalent set.) If C is of finite cardinality, the competence of L for C also implies that L will eventually converge to C as the sample size is increased; hence competence is closely related to Gold's criterion of *learnability in the limit* (Gold, 1967). For infinite sets C, competence implies that L has the somewhat weaker property of *AE-convergence* to C (Kelly, 1988).

Of course, just because the bias of a learner is not too strong does not imply that the bias is appropriate; the bias might be so weak that learning is impractically slow. Competence does *not* imply that L will learn C efficiently; the efficiency of learning is considered in the next section.

Given this formalization of competence, it is now possible to characterize the concepts for which EBL/TS and A-EBL are competent. Since the competence of these algorithms clearly depends on the domain theory T and the operationality predicate $\mathcal{O}$, we will consider the competence of versions of these algorithms parameterized by particular domain theories and operationality predicates: for example EBL/TS$_{T,\mathcal{O}}$ denotes the EBL/TS algorithm using the theory T and the operationality predicate $\mathcal{O}$.

One final observation should be made at this point. We have not, up to this point, specified what action EBL/TS should take if there is more than one proof of some example x. There are two obvious possibilities.

1. One could require that there be only one explanation for each positive instance.
2. One could assume that EBL/TS makes an arbitrary choice among the possible alternatives.

In this section, we will make the assumption that an arbitrary choice is made. This choice is more favorable to EBL/TS, since (as we will see) it leads to competence in a somewhat broader range of circumstances than requiring only a single explanation for each example. However, this choice requires extending the definition of competence. We extend the definition of competence for algorithms L that make arbitrary choices as follows: L is defined to be *competent* for C if for every sample of C and for every possible sequence of choices made by L, L outputs a hypothesis that is consistent with the sample.

Theorem 1 *EBL/TS$_{T,\mathcal{O}}$ is competent for an unknown concept C if and only if*

1. $C = R_1 \cup \ldots \cup R_k$, *where* $\forall 1 \leq i \leq k$, *there is some proof* p_{x_i} *of some* $x_i \in C$ *such that* R_i *is the output of* $EBG(T, \mathcal{O}, p_{x_i})$, *and*
2. $\forall x \in C$, $\forall p_x : p_x$ *is a proof of* x, $EBG(T, \mathcal{O}, p_x) \subseteq C$.

In other words, EBL/TS is competent when the concept to be learned can be described as a disjunction of EBG rules, and when every rule formed by applying EBG to a positive example does not over-generalize that example *relative to the unknown concept C*. Notice that these conditions do *not* imply that the theory is complete and correct for C; in particular, there can be proofs of negative instances. Also note that this condition is somewhat less restrictive than saying that there is exactly one proof for each positive example.

Proof: First we show the "only if" part. If the first condition does not hold, then EBL/TS cannot hypothesize C, so clearly for some sufficiently large sample it must be unable to produce a consistent hypothesis. If the second condition does not hold, then there is some $x^+ \in C$ with a proof p_{x^+} such that EBG over-generalizes given that proof. If EBL/TS is given a sample containing x^+ and selects the proof p_{x^+} to generalize, it will output a

hypothesis $H \supset C$; since a sample can contain negative examples in $H - C$, this hypothesis will be inconsistent with some samples containing negative examples.

The "if" part is straightforward. First, notice that by the first condition, if $x \in C$ then $x \in R_i$, and hence x has some proof. Thus EBL/TS always outputs a hypothesis that covers all the positive examples. If the second condition holds, then EBL/TS will never include an over-general rule in its hypothesis; thus its hypothesis H is a subset of C and hence consistent with the negative data as well.

Theorem 2 *A-EBL$_{T,\Theta}$ is competent for an unknown concept C if and only if*

1. $C = R_1 \cup \ldots \cup R_k$, *where* $\forall 1 \leq i \leq k$, *there is some proof* p_{x_i} *of some* $x_i \in C$ *such that* R_i *is the output of EBG*(T, Θ, p_{x_i}), *and*
2. $\forall x \in C, \exists p_x : p_x$ *is a proof of* x *and EBG*$(T, \Theta, p_x) \subseteq C$.

The difference between these conditions and the conditions of Theorem 1 is that it is not longer necessary that *every* rule formed by applying EBG to a positive example does not over-generalize relative to the unknown concept C; instead it is sufficient that *one* of the explanations of each example is not over-generalized.

Proof: First we show the "only if" part. Again, if the first condition does not hold, then A-EBL cannot hypothesize C, so clearly for some sufficiently large sample it must be unable to produce a consistent hypothesis. If the second condition does not hold, then there is some $x^+ \in C$ such that EBG over-generalizes every proof of x^+. Let $p_1, \ldots, p_n$ be the proofs of x^+, $R_1, \ldots, R_n$ be the rules obtained from applying EBG to these proofs, and let $x_1^-, \ldots, x_n^-$ be negative examples included in $R_1, \ldots, R_n$ respectively. Notice that the R_i's are the only rules that include x^+, since if some other rule R' formed from proof p' included x^+ then p' would also be a proof of x^+. Hence if A-EBL is given a sample containing the positive example X^+ and the negative examples $x_1^-, \ldots, x_n^-$ it will be unable to find any consistent rule that includes x^+, and therefore will be unable to produce a consistent hypothesis.

As before, the "if" part is straightforward: if the conditions above hold, A-EBL will always be able to find a consistent rule that covers each positive example, and hence will always form a consistent hypothesis. ■

3.1.2. Discussion

In introducing EBL/TS, we gave an informal characterization of when EBL/TS was applicable. Intuitively, there are two prerequisites for using EBL/TS: the domain theory must be *complete*, in the sense that explanations can always be generated, and the rules formed by applying EBG must be specific enough so that they do not over-generalize an instance relative to the unknown concept C. In this section, that intuition was confirmed; the condition necessary for EBL/TS to work includes the condition that

$$\forall x \in C, \forall p_x : p_x \text{ is a proof of } x, \; EBG(T, \Theta, p_x) \subseteq C.$$

This can be interpreted as a sort of "granularity condition" on the generalizations produced by EBG; the generalizations cannot be so general that they extend past the boundary of the unknown concept C. This condition may not be met if the level of operationality is too high, or if there are properties relevant to membership in C that are not reflected in the domain theory.

For A-EBL, the corresponding condition is

$$\forall x \in C,\ \exists p_x : p_x \textit{ is a proof of } x \textit{ and } EBG(T, \mathcal{O}, p_x) \subseteq C$$

The change of the quantification on the proofs of x from "$\forall$" *to* "$\exists$" is consistent with the intuition that A-EBL can tolerate incorrect explanations, while EBL/TS cannot; A-EBL only requires that some explanations not do over-generalize, rather than requiring that all explanations do not over-generalize.

Notice that the conditions for competence for EBL/TS also imply that for every positive example x, $x \in R_i$ for some rule R_i produced by EBG, and hence that x has some proof. (The proof of x will be some instantiation of the abstract explanation structure from which R_i is derived.) Thus, an immediate consequence of the results above is that A-EBL is competent for a strictly larger class of concepts than EBL/TS.

3.2. Convergence results

3.2.1. Preliminary definitions

Our definition of learnability is the usual one of "distribution-free" or "probably approximately correct" learnability (Blumer et al., 1986; Valiant, 1984). We start by formalizing the notion of a concept class. A concept is a familiar notion; a concept class simply formalizes the notion of the "inductive bias" of a learning system. We define a *concept class* $\mathcal{C}$ to be a set of concepts, where a *concept* is simply a subset of some base set X, called the *domain*. We will consider concept classes which are parameterized by the "size" of the concepts relative to some *size measure*, and use $\mathcal{C}(n)$ to denote $\{C \in \mathcal{C} : size(C) \leq n\}$.

What we would like is some way of measuring the likely error of a learning program, given samples of a reasonable size. Computing the likely error of a hypothesis is easier if we assume that examples are drawn randomly, according to some fixed probability distribution D. The error of the hypothesis can then be measured relative to this probability distribution. Informally, we would like to say that the hypothesis is "good" if its error is below some arbitrarily set threshold; that is, if the hypothesis is "approximately correct," To formalize these notions, define a *sample of C drawn according to D* to be a sample S^+, S^- of C such that the elements of $S^+ \cup S^-$ are drawn randomly according to the probability distribution function D. The *error* of a hypothesis H is just $D(H \Delta C)$, where C is the unknown concept, and Δ denotes symmetric difference. A hypothesis is ϵ-*good* if its error is less than ϵ.

Since the samples are drawn randomly, there will be some small chance of getting an unrepresentative sample; in this case, it is unreasonable to require the learning algorithm to generate a good hypothesis. Therefore, we must require that the learning algorithm return

an ϵ-good hypothesis only with high probability, not in every case. To account for this, let us introduce another parameter δ, and allow the learning system to be wrong—that is, to produce a hypothesis that is not ϵ-good—with probability at most δ. It is also desirable to allow the learning algorithm to require more examples as one increases the complexity of the concept to be learned, or as one decreases ϵ or δ. Finally, it is unreasonable to expect the learning system to work if its inductive bias is too strong. We thus arrive at the following definition:

Definition 4 *A* pac-learning algorithm *for a parameterized concept class* $\mathcal{C}(n)$ *is an algorithm LEARN with an associated polynomial function* $m(1/\epsilon, 1/\delta, n)$ *so that for every* $n > 0$, *every* $C \in \mathcal{C}(n)$, *every* $0 < \epsilon < 1$, *every* $0 < \delta < 1$, *and every probability distribution D, given a sample of C drawn according to D such that* $|S^+| + |S^-| > m(1/\epsilon, 1/\delta, n)$, *the output of LEARN is a hypothesis H such that*

$$Prob(D(H \Delta C) > \epsilon) < \delta$$

In other words, *LEARN* will probably produce an approximately correct hypothesis, given that the unknown concept C is in the concept class which represents *LEARN*'s inductive bias. Furthermore, *LEARN* will do this using a sample size polynomial in the size of C and the inverse of the error parameters.

The function $m(1/\epsilon, 1/\delta, n)$ is called the *sample complexity* of the algorithm. It gives an upper bound on the number of examples needed to produce a probably approximately correct description of an unknown concept of size n, analogous to the way that the time complexity of an algorithm gives an upper bound on the number of steps before termination. A concept class with an associated pac-learning algorithm is said to be *pac-learnable.*

3.2.2. Analysis of EBL/TS and A-EBL

Let $\mathcal{C}^{\forall}_{T,\Theta}(n)$ denote the class of concepts C such that

1. $C = R_1 \cup \ldots \cup R_k$, where $\forall 1 \leq i \leq k$, there is some proof p_{x_i} of some $x_i \in C$ such that R_i is the output of $EBG(T, \Theta, p_{x_i})$, and
2. $\Sigma_{i=1}^{k} size(R_i) \leq n$, and
3. $\forall x \in C$, $\forall p_x : p_x$ *is a proof of* x, $EBG(T, \Theta, p_x) \subseteq C$.

In the definition, $size(R_i)$ denotes the size measure used in the A-EBL algorithm: namely, the number of nodes in the associated abstract explanation structure. The class $\mathcal{C}^{\exists}_{T,\Theta}(n)$ is defined identically except that the final condition is changed to

$$\forall x \in C, \exists p_x : p_x \text{ is a proof of } x \text{ and } EBG(T, \Theta, p_x) \subseteq C$$

The class $\mathcal{C}^{1}_{T,\Theta}(n)$ is defined identically to $\mathcal{C}^{\forall}_{T,\Theta}(n)$ except that it is also required that there be only one abstract explanation structure for each positive example x. Notice that the concept classes $\mathcal{C}^{\forall}_{T,\Theta}(n)$ and $\mathcal{C}^{\exists}_{T,\Theta}(n)$ are (except for the restrictions on size) precisely

the classes of concepts for which EBL/TS and A-EBL were shown to be competent in the previous section. Also, $\mathcal{C}^{1}_{T,\mathcal{O}}(n)$ is the class of concepts for which EBL/TS can be used without making arbitrary choices; choices are avoided by requiring that each example have only a single explanation. Notice also that, since the condition that $C = R_1 \cup \ldots \cup R_k$ (where the R's are rules produced by EBG) implies that every positive example has at least one explanation, the following inclusions hold:

$$\mathcal{C}^{1}_{T,\mathcal{O}}(n) \subseteq \mathcal{C}^{\forall}_{T,\mathcal{O}}(n) \subseteq \mathcal{C}^{\exists}_{T,\mathcal{O}}(n)$$

The sample complexities of EBL/TS and A-EBL as pac-learning algorithms can now be bounded for these concept classes.

Theorem 3 *For all theories T and all operationality predicates $\mathcal{O}$, EBL/TS$_{T,\mathcal{O}}$ is a pac-learning algorithm for $\mathcal{C}^{1}_{T,\mathcal{O}}(n)$ with sample complexity*

$$m\left(\frac{1}{\epsilon}, \frac{1}{\delta}, n\right) = O\left(\left(\frac{1}{\epsilon}\log\frac{1}{\delta}, \frac{n \log r}{\epsilon}\log\frac{1}{\epsilon}\right)\right)$$

where r is the number of rules (i.e., Horn clauses) in T.

Proof: Any set of outputs of EBG is determined by a set of abstract explanation structures, which in turn is uniquely defined by the sequence of clause labelings generated by the following procedure: arrange the abstract explanation structures in any arbitrary order, and traverse the nodes of each tree in postfix order, reading off the names of the clauses which were used to prove each term. The symbol **nil** is used to indicate when an operational leaf is reached. The result of this is a string of length n whose letters are either the names of one of the r clauses in the theory, or **nil**. (To see that this uniquely determines the set of abstract explanation structures, notice that since the number of sons of each node can be determined from the name of the clause labeling the node, the tree of clause names which is used internally in the A-EBL to represent an abstract explanation structure can be reconstructed; the abstract explanation structure itself can then be generated using the algorithm in Appendix B.) There are of course, only $(r + 1)^n$ such strings; hence the number of sets of abstract explanation structures with total size n is at most $(r + 1)^n$.

It is clear that EBL/TS will produce a concept of minimize size; it uses the minimum number of disjuncts to cover a set of examples, and since there is only one generalization for each example, there is no flexibility in choosing the size of these disjuncts. The remainder of the theorem follows from the observation that the VC-dimension (Blumer et al., 1986) of a concept class $\mathcal{C}$ is bounded by $\log_2|\mathcal{C}|$ and Theorem 2 of Blumer et al., (1986). ■

The result above *cannot* be extended to the broader class $\mathcal{C}^{\forall}_{T,\mathcal{O}}$. If an adversary is allowed to pick T, $\mathcal{O}$, and the choices made by EBL/TS, there are concepts in $\mathcal{C}^{1}_{T,\mathcal{O}}$ that EBL/TS cannot pac-learn. An example is the theory of Figure 10. This theory contains two alternate definitions of a predicate for list membership, one which is operational and one which is not. The desired specialization of the theory is the theory consisting of the single rule below, defining the new concept mem′:

C_T=*mem(X,L)*

mem1(X,L) ⇒ mem(X,L)
mem2(X,L) ⇒ mem(X,L)

true ⇒ mem2(X,[X|Y])
mem2(X,Y) ⇒ mem2(X,[Z|Y])

mem1 is an operational predicate with the same definition as mem2

Figure 10. Theory with a specialization that EBL/TS cannot pac-learn.

mem1(X,L) ⇒ mem′(x,L)

However, EBL/TS can output hypotheses of arbitrary size by consistently choosing to form rules using the non-operational theory of membership *mem2*. For example, EBL/TS could output a hypothesis of the form

true ⇒ mem′(A,[A|B])
true ⇒ mem′(A,[B,A|C])
true ⇒ mem′(A,[B,C,A|D])
⋮

Each of these rules is formed by applying EBG to a proof constructed using the *mem2* predicate. Notice that the number of examples needed for convergence does not depend only on ϵ, δ and n, as required by the definition of pac-learnability; instead it depends on the number of special case *mem2* rules needed to make the approximate theory above accurate. Thus the desired specialization will not be pac-learned.

In contrast, A-EBL pac-learns the following class of concepts.

Theorem 4 *Let r denote the number of rules in T. Then for all theories T and all operationality predicates* $\mathcal{O}$, *A-EBL is a pac-learning algorithm for* $\mathcal{C}^{\exists}_{T,\mathcal{O}}(n)$ *with sample complexity*

$$m\left(\frac{1}{\epsilon}, \frac{1}{\delta}, n\right) = O\left(\left(\frac{1}{\epsilon}\log\frac{1}{\delta}, \frac{n\log r}{\epsilon}\left(\log\frac{n\log r}{\epsilon}\right)^2\right)\right)$$

Proof: The learnability result and the sample complexity follows directly from the counting argument above, the fact that the greedy set cover will produce a concept which is within n log m of the minimal size, and Theorem 2 of Blumer et al., (1986). ■

The final theorem of this section gives a lower bound on the covergence rate of any pac-learning algorithm for this concept class.

Lemma 1 *There exist theories T such that, if* $\epsilon \leq 1/8$, $\delta \leq 1/100$, and $d(n) \geq 2$, *every pac-learning algorithm for* $\mathcal{C}^{1}_{T}$ *must have a sample complexity of at least*

$$m\left(\frac{1}{\epsilon}, \frac{1}{\delta}, n\right) = \Omega\left(\frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{n}{\epsilon}\right)$$

Proof: Recall that the Ω notation is analogous to the big O notation, but used for lower bounds; more precisely, $g(n) = \Omega(f(n))$ if $\exists c : cf(n) \geq g(n)$ for all but finitely many n. Consider the following theory T, defining the concept $C_T = p(X)$. T contains exactly n Horn clauses of the form *true* $\Rightarrow p(b_i)$, where each b_i is a distinct constant symbol. No predicates of T are marked as operational. It can easily be verified that for every subset C of the set $\{p(b_1), \ldots, p(b_n)\}$, there is a specialization $H \in \mathcal{C}^{1}_{T,\Theta}(n)$ of size no greater than n such that $H = C$. Hence the VC-dimension (Blumer et al, 1986) of $\mathcal{C}^{1}_{T,\Theta}(n)$ is at least n; the theorem is now immediate from Theorem 1 of Ehrenfeucht et al., (1988). ■

3.3. Complexity results

The only problem in performing a time analysis of A-EBL is that the cost of computing all the proofs of each training example cannot be bounded, since it is well known that Horn clause theorem-proving can be undecidable. However, a relativized result can be obtained: if we ignore the cost of theorem-proving, we have the following complexity result for our implementation.

Theorem 5 *Consider a learning problem in which*

- *n is the size of the unknown concept,*
- *m is the number of training examples,*
- *P is the number of proofs of positive examples, and*
- *N is the number of proofs of negative examples.*

If we neglect the time required to generate and abstract all the proofs, A-EBL runs in time

$$O(n \log m \cdot P[N + P])$$

Proof: (Please refer to Figure 9). The majority of the time is spent in the **while** loop in the main program. Since the set cover algorithm produces a cover a size at most $n \log m$, and each element of the cover is of size at least 1, this loop can be traversed at most $n \log m$ times. Inside the loop are three operations; it can easily be shown that finding the optimal consistent explanation structure dominates the other two in cost. This operation involves looping over each positive explanation structure and a) testing its consistency and then b) computing the ratio of the number of examples covered to the size of the explanation structure. The former operation takes time $O(N)$ using the simplest test—simply looping through the negative examples and testing for equivalence. The latter operation takes time $O(P)$, again by using the simple operation of comparing the explanation structure to every other explanation structure for a positive example. ■

It is easy to show that, by improving our naive consistency tests, the run-time can be improved to $O(n \log m \cdot P[\log P + \log N])$.

3.4. Discussion

To summarize, we have presented three substantial formal results. The first result concerns the domains of applicability of EBL/TS and A-EBL. In Section 3.1, we characterized the concepts C for which A-EBL and EBL/TS are *competent*; the notion of competence that was used was the ability to produce a consistent hypothesis for every sample of a concept. As an absolute characterization of the competence of A-EBL, the results of Section 3.1 were somewhat unsatisfying, since they are based on difficult-to-verify conditions regarding the behavior of EBG on proofs in the domain theory. However, the results are informative as a relative characterization of the competence of A-EBL and EBL/TS; they show that A-EBL is applicable in a wider range of circumstances.

The second result concerns the sample complexity of EBL/TS and A-EBL. If we adopt the assumption that examples are presented stochastically according to some fixed probability distribution D, then the worst-case sample size needed for learning of EBL/TS and A-EBL are comparable: the sample complexities are within a factor that is logarithmic in the complexity of the concept being learned. These worst-case results are fairly tight; the bound for EBL/TS is within an additional logarithmic factor of a known lower bound on the number of examples needed for learnability.

To get a better feel for how the sample complexity of EBL and A-EBL compare, let us make some assumptions to simplify the expressions for the sample complexities. If we assume that the size of the domain theory is fixed, and that the second term of the *max* expressions dominate (which they will, unless δ is exponentially much smaller than ϵ), we get the following functions. The final line of the table uses the lower bound theorem. We see that according to this worst-case analysis, A-EBL requires only a logarithmic number of additional examples over the number needed by EBL/TS.

$$\text{For A-EBL: } \frac{n}{\epsilon}\left(\log\frac{n}{\epsilon}\right)^2$$

$$\text{For EBL/TS: } \frac{n}{\epsilon}\log\frac{1}{\epsilon}$$

$$\text{For any pac-learner: } \frac{n}{\epsilon}$$

The third result concerns the *time complexity* of the algorithm. It is not possible in general to restrict the time spent in theorem proving; however, if we neglect the time spent in theorem proving, and consider the proofs of each example to be an input to the algorithm, our implementation runs in time polynomial in its inputs. Although our implementation is not optimally efficient, it is possible to bring this complexity down to a very low-degree polynomial: linear in the complexity of the concept being learned, and $O(n \log n)$ in the total number of explanations.

It is important to realize, however, that these results are worst case results; in particular, the analysis of sample complexity is a worst case result over all possible probability

distributions. It is possible that more efficient learning algorithms could be designed for some particular probability distribution, for instance for uniform distributions. The constants associated with the bounds on sample complexity are also quite high, and hence it is not clear from the analysis how fast A-EBL would converge given small sets of representative examples of some unknown concept. A final drawback of the analysis is that it does not help us assess certain other attributes of the algorithm of practical interest, such as the ease of implementation, and the dependence of the sample complexity on the syntactic form of the domain theory.

In the next section, we will use experimental techniques to answer some of these questions.

4. Experimental validation of A-EBL

In this section, we will first evaluate the behavior of the algorithm when given a small set of examples provided by a friendly and informative teacher who does *not* know the internal workings of the learning algorithm. The goal of these experiments is to determine if the learning system is "easily instructable": whether it can be taught a concept, or a good approximation to a concept, given a small number of examples. The constraint that the teacher does not know the internal workings of the algorithm raises our confidence that learning would also be successful if some other representative set of examples were chosen. As an additional verification of the formal results, we will also evaluate the algorithm on randomly generated data.

To enforce the constraint that the teacher does not know the internal workings of the algorithm, our learning problems are taken from a well-known introductory book on playing bridge (Sheinwold, 1964). The domain theory is transcribed, as directly as possible, from information presented in the book; the examples used for learning are the examples used in the book to illustrate application of the bidding system. Learning is necessary because the rules presented in the book are not a complete, correct, consistent bidding system; some of the bidding rules are heuristics that should not always be followed.

The following experiments are conducted.

Experiment 1. A domain theory for the concept *plausible opening bid* is specialized to a domain theory for the concept *correct opening bid* using A-EBL, as in our example of Section 2.2; the difference between this experiment and the example of Section 2.2 is that this theory is much larger and more realistic. The correctness of the specialization learned by A-EBL is evaluated by testing it on a sample test from Sheinwold (1964). This experiment demonstrates that A-EBL can be applied to domain theories of reasonable size and complexity, and that it can obtain good results not only asymptotically, but also on samples of moderate size.

Experiment 2. Evaluation of the concept learned in Experiment 1 shows that there is some room for improvement. In particular, some of the problem hands in the sample test are not bid correctly by the specialization learned by A-EBL. Closer investigation of these sample problems shows that the strategy used by A-EBL to choose between multiple explanations is not the cause of these errors: in fact, *no* strategy for choosing between multiple explanations would work, because no explanation of any of the training examples is also an explanation of the unsolved sample problems. In short, the reason for the failure on these problems is that the bias of the learning system is too strong.

To solve this problem, we weaken the bias of the learning system by considering as possible explanations of an example both the actual explanation and a set of possible *abstractions* of that explanation, where an explanation is *abstracted* by marking some internal nodes of the explanation structure as operational. This has the effect of expanding the set of possible specializations of a domain theory. The extended version of A-EBL is then evaluated, as before, by testing the specialization it produces on the sample test.

This experiment has two purposes. First, it demonstrates another application of A-EBL. Second, the performance of this extension of A-EBL is indicative of the performance of A-EBL on domain theories which produce a large number of explanations for each example. This experiment thus tests (somewhat indirectly) the performance of A-EBL on domain theories which are much "weaker" than the one considered in Experiment 1: that is, which produce many more explanations (on the order of several dozen explanations for each example).

Experiment 3. Sheinwold does not define precisely the meaning of every intermediate concept used in his rules for choosing an opening bid. In Experiments 1 and 2, this problem was circumvented by hand-coding an appropriate definition for each of these intermediate concepts, using knowledge of bridge bidding not derived from Sheinwold (1964). (See Appendix E for a listing of these hand-coded predicates.) This is undesirable, not only because it introduces an element of subjectivity into the transcription of the domain theory, but also because it is presumably unnecessary: after all, thousands of people have learned to open bridge hands from reading Sheinwold (1964) alone.

To address this problem, we construct a very weak domain theory for the concept *hand of opening strength*, a concept which contains as subconcepts most of these undefined intermediate concepts. The weak theory is drawn only from information in Sheinwold (1964) and commonsense knowledge, and is a more realistic version of the example of Section 2.3. Using the examples presented in the book, and weak "partial definitions" of the undefined concepts, A-EBL is able to learn a correct specialization of this weak theory. The specialization is tested in two ways: first, against the sample test from Sheinwold (1964), and second, by repeating Experiments 1 and 2 using the learned concept in place of the hand-coded concept.

This experiment has three purposes. First, it tests the performance of A-EBL on domain theories which are extremely weak, yielding as many as hundreds of different explanations for each example. Second, repeating Experiments 1 and 2 with a domain theory which is syntactically different, but semantically the same (at least over the training examples and test cases) tests the sensitivity of A-EBL to the way in which the domain theory is encoded. Third, it eliminates a source of subjectivity in transcribing the domain theory; hence the repetitions of Experiments 1 and 2 give a more reliable indication of the performance of A-EBL than the original versions of the experiments.

Experiment 4. The examples and test data taken from Sheinwold (1964) are a fair test of the learner in two ways; they are representatives of a naturally-occurring concept of some complexity, and they were chosen without knowledge of the learning algorithm. It is not an ideal test, however, because of its small size. To circumvent this problem, a program was written that randomly generates bridge hands and then opens them using hand-coded bidding rules; this allows larger sets of training and test data to be generated. In Experiment 4, the learning tasks of Experiments 1, 2 and 3 are revisited using larger sets of training and test data.

4.1. The domain

4.1.1. Source of knowledge and examples

The domain theories discussed in this paper formalize the first three chapters of Sheinwold (1964). The first chapter covers evaluation of the hand in terms of high card points and so on; this is a very simple procedure which can be easily coded. This chapter is mostly used in the definitions of the operational predicates of our domain theory. Chapters two and three cover opening bids of one of a suit and no-trump opening bids, respectively. Most of these chapters are devoted to presenting heuristic rules for bidding; these rules are primarily used in the definitions of the non-operational parts of our domain theory.

These three chapters are also the part of the book from which our examples are drawn. Our database of examples is comprised of the forty-eight examples used by Sheinwold in the first three chapters, with no additions, and only five omissions. The omitted examples dealt with opening in third- or fourth-hand position; they were omitted because eliminating position information greatly simplified our representation of the bidding problem.

4.1.2. Motivations for choosing the domain

The domain of bridge bidding is suited to our purposes for several reasons.

- It is uncontrived, in the sense that is was not constructed for the purpose of demonstrating a learning algorithm.
- Most of the rules required to understand the examples of the first two chapters are clearly and explicitly presented, which makes transcription into logic easy and direct, and reduces unconscious biases introduced in definition of the domain theory.
- Finally and most importantly, the information presented is primarily rules and examples that clarify their use, which is exactly the type of input needed by A-EBL.

A disadvantage of the choice of bridge is that in general, games and other artificial domains are simpler and cleaner then most natural domains; results obtained in such domains are sometimes difficult to extend to broader classes of problems.

It is clear from the test that the bidding rules presented are over-general, and that the examples are an important part of the author's presentation. Often Sheinwold explicitly states that a rule is merely a heuristic, and should not always be followed. In most of these situations, a series of clarifying examples follow. For instance, in discussing rules for bidding two-suited hands (Sheinwold, 1964, page 16), Sheinwold says "the general rule is, if your suits are *unequal* in length, bid the *longer* one; if your suits are *equal* in length, bid the *higher* one" but immediately adds that "you have to disregard this general rule on some hands." After a short digression into what defines a "biddable suit," he presents fifteen examples which clarify this general rule. Again, discussing three-suited hands (Sheinwold, 1964, page 23), Sheinwold says "as a rule, start with lowest or middle suit ... the examples show how easy it is to keep the bidding low." Four examples are then presented of three-suited hands and their appropriate bids.

In spite of the general clarify of the presentation of the bidding rules, some judgment was required in formalizing the domain. In particular, Sheinwold accompanies each example with an informal discussion. Sometimes this discussion gives a specific bidding rule followed; sometimes this discussion explains how the bid satisfied some broad objective, such as keeping the bidding low; in four places, a new rule which is an exception to a previously given general rule is cited to explain a bid which is not what the reader would otherwise expect. In transcribing the domain theory, we did not attempt to formalize the discussions following each example. No doubt in doing so, some information was missed; however, we felt it was safer to omit information than to run the risk of inserting information which was not explicitly present in the process of transcribing an informal discussion into logic. The only exception to the policy of not formalizing these discussions are the four "exceptional" rules mentioned above; these were of course needed to explain their associated examples. In two of these exceptional rules, it was also necessary to provide an interpretation of some undefined subconcepts. These rules are discussed in Appendix E.

4.2. Experiment 1: Learning the concept "Correct Opening Bid"

4.2.1. The domain theory and training data

The first experiment uses a domain theory similar to the theory shown in Figure 1. The theory considered here is much more substantial, though. It contains 124 clauses and 539 lines of code; of this code, about a quarter, 29 clauses and 130 lines of code, is non-operational. By way of comparison, the A-EBL learning system itself contains only 243 clauses and 862 lines of code.[7]

The top-level predicate *plausible-bid* contains a series of ten clauses that describe when to make various types of bids. These rules are straightforward encodings of rules given in Sheinwold (1964); typically, they check to see if certain conditions are true and then recommend a bid. For example, the rule for the two no-trump bid is

$$\text{hcp}(H,HCP) \wedge \text{between}(HCP,22,24) \wedge \text{balanced-distribution}(H) \wedge \text{all-suits-stopped}(H) \Rightarrow \text{plausible-bid}(H,\text{bid}(2,\text{no-trump}))$$

which states "bid 2 no-trump on any hand H with between 22 and 24 high card points, balanced distribution, and stoppers in all suits." Most of the conditions used in these rules (for example, "balanced distribution") are clearly defined in the text; two notable exceptions are the concepts *comfortable-rebid* and *length-in-majors*, which are used in defining the predicate *opening-strength*. In this experiment, definitions for these concepts were hand-coded, using knowledge from other sources. Experiment 3 will consider how these concepts can be learned from examples.

As suggested by the example of Section 2.2, the parts of the theory that are most problematic are the rules used to choose between two or more biddable suits. The theory contains a preference predicate, similar to that used in the previous example: the predicate encodes seven heuristic rules for choosing between two biddable suits, and two rules for choosing among three biddable suits. Most of these rules were over-general and required

specialization. As in the example, it was possible to use A-EBL to specialize the examples because the conditions missing from the over-general rules were used elsewhere in the proof, and because an appropriate choice of operationality was made.

The theory is presented in detail in Appendix E.

Negative examples, which are required by A-EBL, were obtained by assuming that the list of bids recommended for each example hand is a complete and exhaustive list of correct bids; that is, all bids for a hand which were *not* recommended were assumed to be negative examples.

For most of the example hands, only one bid was recommended. For three of the 43 example hands, two bids were recommended. Each of these examples was treated as two separate positive examples; in other words, each recommended $\langle Hand, Bid \rangle$ pair was used as a positive example. Thus there were a total of 46 positive examples. There were 15 non-trivial negative examples (a negative example is considered non-trivial if there is a proof that the example is in the concept C_T associated with the domain theory—in this case, if there is a plausible but incorrect bid.) The maximal number of proofs for an example was 3, and the average number of proofs was 1.4.

4.2.2. Experimental results

The biggest disadvantage of using A-EBL (relative to standard EBL) is that more examples may be needed by the learning program to converge. However, on this learning problem, A-EBL learned a set of 30 rules which explain all 46 examples. Notice that even *without* multiple explanations, the standard algorithm EBL/TS would have required at least 30 examples to learn this set of rules. This gives some experimental support to the hypothesis that the number of additional examples needed by A-EBL to choose between multiple explanations is reasonable.

To evaluate the correctness of the learned concept, we tested it on the relevant questions from the sample test in Sheinwold (1964). This test is quite comprehensive, covering all aspects of bridge discussed in his book. Unfortunately for us, this means that only a small portion of it, 16 out of 200 questions, is devoted to opening bids. For purposes of comparison, we also tested the original imperfect domain theory. The original domain theory makes errors on four out of the sixteen test problems, where an error is defined as either returning some bid which is not recommended, or not returning any recommended bid. The specialized theory corrects two of those errors. One of the two remaining mistakes is on the hand

♠KJ642 ♡A5 ♢3 ♣AQ732

The correct bid for this hand is one club; however, the theory learned by A-EBL does not recommend any bid. The learned theory is incorrect because the rule learned for bidding clubs over spades is specialized to a specific range of points, rather than all minimum hands. The other mistake is made on the hand

♠AKJ10876 ♡A762 ♢K2 ♣–

for which the correct bid is one spade. Again, no bid is recommended for this hand by the learned theory; this is because none of the examples had a seven-card suit, and the program is unable to generalize the length of suits given our definition of operationality.

The results of this experiment are summarized at the end of the next section in Table 3.

4.3. Experiment 2: Weakening the bias of the learner

It might be thought that the two mistakes made by the specialization of *plausible-bid* learned by A-EBL are a result of incorrect choices made in the set-covering routine, which is heuristic. However, close examination of these two anomalous test problems shows that this is not the case: for both of the anomalous test problems, *none* of the possible explanations for the test problem was also an explanation of *any* of the training examples. In other words, no learning strategy which learns simply by examining the training examples, selecting one or more explanations of each training example as valid, and then generalizing these selected explanations to form rules would be able to correctly bid the two anomalous test problems, when given the training examples, domain theory, and operationality criterion of Experiment 1.

In short, the anomalous test problems indicate that the bias imposed by the domain theory and operationality predicate is too strong to learn a correct concept from the examples given. This is unfortunate, because clearly Sheinwold expects an intelligent reader to be able to answer these questions correctly, given similar information.

Further analysis shows that, for each anomalous test case, although the correct explanation is not *identical* to the explanation of any of the training examples, the correct explanation is very *similar* to the explanation of some training example. For example, the explanation of the correct bid of one club for the test case ♠KJ642 ♡A5 ◇3 ♣AQ732 differs in only one subproof (the subproof for *opening-strength*) from the explanation used to justify the bid of one club on the training example ♠KQJ75 ♡5 ◇62 ♣AJ963. This particular problem could be handled by marking the predicate *opening-strength* as operational. Unfortunately, if this were done, then in some other cases, every rule learnable from a training example would be over-general.

Several extensions to Experiment 1 which would correctly handle the anomalous test problems are possible. For instance, the anomalous test problems could be fixed by modifying the operationality predicate; one could make the predicate *opening-strength* operational when processing the training example ♠KQJ75 ♡5 ◇62 ♣AJ963. However, there are other cases in which it is necessary for the *opening-strength* predicate to be non-operational, in order to distinguish between weak and strong hands. In principle, it is possible to make the operationality of predicates dependent on context, so this obstacle is not insurmountable; however, it is not clear how this new, complex definition of the operationality of *opening-strength* can be determined without prior knowledge of the concept to be learned.

Another possible modification is to weaken the bias of the learning system by considering as possible explanations of an example both the actual explanation, and a set of possible *abstractions* of that explanation. An *abstraction* of an explanation can be constructed by simply taking some non-operational node and marking it as operational. For instance, the explanation of the training example ♠KQJ75 ♡5 ◇A2 ♣AK963 could be abstracted

by marking the *opening-strength* predicate as operational; this would produce a more abstract explanation which also explains the first anomalous test problem.

This technique will greatly increase the number of explanations which are considered by the set covering algorithm. If explanations are abstracted by picking a single non-operational node and marking it operational, then in addition to the unabstracted explanations, each explanation will generate $O(n)$ abstracted explanations, where n is the size of the original explanation structure. If explanations are abstracted by picking k non-operational nodes and marking each of them operational, then each explanation will generate $O(n^k)$ abstracted explanations.

This strategy is reminiscent of explanation-based analogical reasoning techniques such as those described in Huhns and Acosta (1987) and Kedar-Cabelli (1987). These analogical reasoning techniques use EBG with an artifically high level of operationality to produce rules that match any potential analogies. The difference is that instead of generating a single very general rule, a large number of slightly more general rules are produced, each corresponding to a specific class of analogical instances. A-EBL's mechanism for choosing among multiple explanations can then be used to pick general rules that *only* match instances that should be treated the same as the training example. One can view such a rule as an *explicit* representation of the generalization that would be *implicitly* made by an analogical reasoner.

As an experiment, A-EBL was modified to consider in its set cover phase all extensions of an explanation which are obtained by picking k non-operational nodes, for $k = 1$ and $k = 2$, and marking them operational. We call this extension of A-EBL *analogical A-EBL*, or ANA-EBL for short, because of its similarities to the work of Huhns and Acosta (1987) and Kedar-Cabelli (1987). The time required for learning was greater for ANA-EBL; however, the concept learned was more accurate on the test cases, getting fifteen of the sixteen examples right. This experiment also indirectly indicates the performance of A-EBL with an even weaker domain theory (for $k = 1$, there are an average of 30 abstract explanations for each example.)

The results of Experiments 1 and 2 are summarized in Table 3. The table shows that A-EBL and its extension do well (although not perfectly) on this learning problem, in spite of the presence of multiple explanations; the CPU times indicate that the current implementation of the algorithm is reasonably efficient. In Table 3, the augmented version of A-EBL is listed as ANA-EBL (notice that the unaugmented version of A-EBL is equivalent to ANA-EBL with $k = 0$). Times are in CPU seconds on a SparcStation 1+. The abstractions of each explanation are generated at learning time; the times given in the column headed "Explain" are simply the times needed to generate and store the initial set of unabstracted explanations. The theories learned by ANA-EBL with $k = 1$ and $k = 2$ each contain 25 clauses.

Table 3. Learning *plausible-bid* with hand-coded intermediate concepts.

Theory	Accuracy	Time	
		Explain	Learn
Initial domain theory	12/16	—	—
Optimal choice	14/16	—	—
output of A-EBL	14/16	44.0	58.8
output of ANA-EBL ($k = 1$)	15/16	44.0	173.6
output of ANA-EBL ($k = 2$)	15/16	44.0	860.3

4.4. Experiment 3: Learning the concept "Hand of Opening Strength"

4.4.1. The domain theory

In Sheinwold (1964, page s 13–14), the following rules are given for deciding when a hand is strong enough to open (i.e., make a bid other than "pass"):

- Any hand with 14 or more high card points can be opened.
- Any hand with 12–13 high card points, 2 or more quick tricks, and a comfortable rebid can be opened.
- Any hand with 10–11 high card points, 2 or more quick tricks, a comfortable rebid and good length in the major suits can be opened.

These rules can be easily encoded into logic; Figure 11 shows one such encoding. However, these rules cannot be easily extended into a complete logical theory, because Sheinwold gives no rules for deciding when a hand has a "comfortable rebid" or for when it has "good length in majors": instead, he seems content to leave the interpretation of these terms to the reader. This is a common situation in natural language presentations. It is reasonable to assume that some learning mechanism, in conjunction with background knowledge, is responsible for inferring the definitions of these terms.

In fact, by applying some commonsense knowledge, it is possible to come up with many constraints on the meanings of these terms. For instance, if a hand has two biddable suits, then there is certainly a "comfortable" second bid.

biddable(Suit1,Hand) $\wedge$ biddable(Suit2,Hand) $\wedge$ (Suit1 $\neq$ Suit2) $\Rightarrow$ comfortable-rebid(Hand)

A second situation in which there is a "comfortable rebid" is if the hand contains a suit which is somewhat longer, or somewhat stronger, than is required to merely be biddable. In this case, bidding this strong suit a second time is reasonable. Given that the shortest biddable suit contains four cards, we can formalize this situation as follows.

rebiddable(Suit,Hand) $\Rightarrow$ comfortable-rebid(Hand)

suit(Suit) $\wedge$ length(Suit,Hand,N) $\wedge$ somewhat-larger(N,3)
$\wedge$ hcp(Suit,Hand,HCP) $\wedge$ somewhat-large(HCP) $\Rightarrow$ rebiddable(Hand)

high-card-points(Hand,HCP) $\wedge$ HCP$\geq$14 $\Rightarrow$ opening-strength(Hand)

quick-tricks(Hand,QT) $\wedge$ (QT$\geq$2) $\wedge$ comfortable-rebid(Hand)
$\wedge$ high-card-points(Hand,HCP) $\wedge$ between(HCP,12,13) $\Rightarrow$ opening-strength(Hand)

quick-tricks(Hand,QT) $\wedge$ (QT$\geq$2) $\wedge$ comfortable-rebid(Hand)
$\wedge$ high-card-points(Hand,HCP) $\wedge$ between(HCP,12,13)
$\wedge$ length-in-majors(Hand) $\Rightarrow$ opening-strength(Hand)

Figure 11. Sheinwold's rules for *opening-strength*.

Similarly, we can infer that a hand has "good length in the major suits" if it has either one long major suit, or (perhaps) two shorter ones.

length(spade,Hand,NS) ∧ length(heart,Hand,NH)
∧ somewhat-large(NS+NH) ⇒ length-in-majors(Hand)
major-suit(Suit) ∧ length(Suit,Hand,N) ∧ somewhat-large(N) ⇒ length-in-majors(Hand)

The only thing we are missing now is a definition of the predicates *somewhat-large* and *somewhat-larger*. Some facts about them can, however, be easily deduced. For instance, since all numbers dealt with here are natural numbers, then a number that is "somewhat large" will be at least as large as zero, the smallest natural number. Also, if $N - 1$ is "somewhat larger" than M, then N is also "somewhat larger" than M. These observations can be formalized as the following definitions.

somewhat-larger(N,−1) ⇒ somewhat-large(N)
somewhat-larger(N,M) ⇒ somewhat-larger(N+1,M)

Finally, we still need some mechanism to pick out those pairs of numbers N and M such that N is "somewhat larger" than M. A natural way of doing this with over-general rules is to add the following clause to our theory.[8]

N > M ⇒ somewhat-larger(N,M)

It is possible to interpret this rule as a possible assumption, which can be made in order to construct a proof in the theory above (which would otherwise be incomplete). Since the assumption may be false, the rules above are an over-general subtheory for the predicates *somewhat-large* and *somewhat-larger*. Hence, the domain theory for *opening-strength* is over-general. In a moment, we will see how A-EBL fares in specializing this over-general theory.[9]

4.4.2. Discussion of the domain theory

At first glance, it might seem that there will be an enormous number of proofs for each example: if the subgoal *somewhat-large*(N) is generated, then the rules for *somewhat-larger* will in effect propose a different proof for each possible cutoff point between N and 0. Worse, several such subgoals can be generated by a single training example, and the number of total proofs is the product of the number of proofs of each subgoal. Since A-EBL has to compute all of these proofs, it might seem that the technique is not applicable.

However, examination of the theory reveals that no single proof will have more than three subgoals of the form *somewhat-larger*(N,M): the worst case is two subgoals from the *rebiddable* predicate, and one from the *length-in-majors* predicate. Moreover, N will be fairly small: at most 13 if it represents the length of a suit or pair of suits, or 11 if it represents the number of high card points in a suit. So not knowing the cutoffs multiplies the number of proofs for an example by at most a factor of $13 \cdot 11 \cdot 11$ in the worst case.

The efficiency of our implementation is such that this number of proofs can be handled comfortably.

4.4.3. *Experiment results*

Our experimental results for this domain theory are summarized in Table 4. A hand was interpreted as a positive example of the concept *opening-strength* if any opening bid other than "pass" was recommended, and a negative example otherwise. All but three of the negative examples were trivial. Using approximately nine minutes of CPU time on a Sparc-Station 1+, the A-EBL algorithm learned a set of three rules which explain all 43 positive examples and three non-trivial negative examples. The maximal number of proofs of an example was 289, and the average number of proofs was 96. The correctness of the learned concept was tested by applying it to the questions in the sample test, and seeing if the concept *opening-strength* corresponded with the hands which Sheinwold recommended by opened.

As the table shows, the learned concept is perfect on the test cases. However, this is not a strong test of the correctness of the concept: since the primary goal of the sample test is to test selection of the correct suit to bid, and not whether to bid or not, there is only a single test question for which the correct bid is to pass. Since the original domain theory was over-general—that is, it classifies hands as being of opening strength strictly too often—even the original weak domain theory does well on the test cases. (The original domain theory is not, however, a correct theory of opening strength, as is indicated by its imperfect performance on the training set.)

As a second test of the concept learned for *opening-strength*, the definition of the learned concept was spliced into the domain theory for *plausible-bid* used in Experiments 1 and 2, and Experiments 1 and 2 were repeated. This is a stronger test because the opening bid subtheory, in addition to deciding if a hand is strong enough to open, is used indirectly in learning preference rules. The results of these experiments are summarized in Table 5; they indicate that the learned subtheory for *opening-strength* is useful in this secondary task of supporting learning, as well as in determining if a hand is of opening strength.[10]

In addition to providing another test of the correctness of the *opening-strength* concept, repeating these experiments tests the sensitivity of A-EBL to the syntactic expression of the domain theory, since the learned definition of *opening-strength* is quite different from the hand-coded definition. In this case at least, A-EBL performed comparably on the two versions of the domain theory: although the concepts learned were different, they performed identically on the test cases.

Table 4. Learning *opening-strength*.

	Accuracy		Time	
Theory	Training Examples	Test Set	Explain	Learn
Domain theory	41/43	16/16	—	—
output of A-EBL	43/43	16/16	78.8	449.9

Table 5. Learning *plausible-bid* with learned intermediate concepts.

Theory	Accuracy	Time	
		Explain	Learn
Domain theory	12/16	—	—
Domain theory w/learned opening-strength	12/16	—	—
output of A-EBL	14/16	54.5	100.2
output of ANA-EBL ($k = 1$)	15/16	54.5	364.3
output of ANA-EBL ($k = 2$)	15/16	54.5	***

4.5. Experiment 4: Learning from random data

The examples and test data taken from Sheinwold (1964) are a fair test of the learner in two ways; they are representatives of a naturally-occurring concept of some complexity, and they were chosen without knowledge of the learning algorithm. It is not an ideal test, however, because of its small size. To circumvent this problem, a program was written that randomly generated bridge hands and then opened them using hand-coded bidding rules. The hand-coded rules are a reasonable implementation of the bidding system presented in Sheinwold (1964); for instance, they bid all the problem hands in the sample test correctly. By using this program as a classifier, an unlimited amount of training and test data can be generated. However, it should be noted that this introduces another source of potential bias, because the data no longer consists of true representatives of Sheinwold's *opening-bid* concept, but of representatives of our own interpretation of that concept.

Two experiments were performed using this source of data. First, the behavior of A-EBL and ANA-EBL in learning *correct-bid* using the domain theory of Experiments 1 and 2 was studied. A test set of 1000 hands was generated and classified by the generation program, as well as a separate training set of 300 hands. A-EBL and ANA-EBL with $k = 1$ and $k = 2$ were then given progressively larger subsets of the training set; the accuracy of each hypothesis was measured by using it to classify the hands in the test set, and comparing the classifications to the correct ones. This experiment was repeated 10 times and the error rates were averaged, using the same test set in each trial. The result is the "learning curve" shown in Figure 12 that plots the accuracy of the hypothesis against the number of training examples. For comparison, the accuracy of the original over-general domain theory is also known.

The experiments confirm that increasing k can improve the sample complexity of learning; going from A-EBL (which is equivalent to ANA-EBL with $k = 0$) to ANA-EBL with $k = 1$ substantially improves the learning rate, and going from $k = 1$ to $k = 2$ also leads to an additional slight but statistically significant[11] improvement.

Randomly generated data was also used to learn the *opening-strength* predicate using the domain theory of Experiment 3. Again, a test set of 1000 hands was generated and classified by the hand-coded bidding program. Then a separate training set of 50 hands was generated and classified by the hand-coded program. A-EBL was then given the first 10 examples from the training set, the first 20 examples, the first 30, the first 40, and finally the entire set. The accuracy of each hypothesis produced was measured by using it to classify

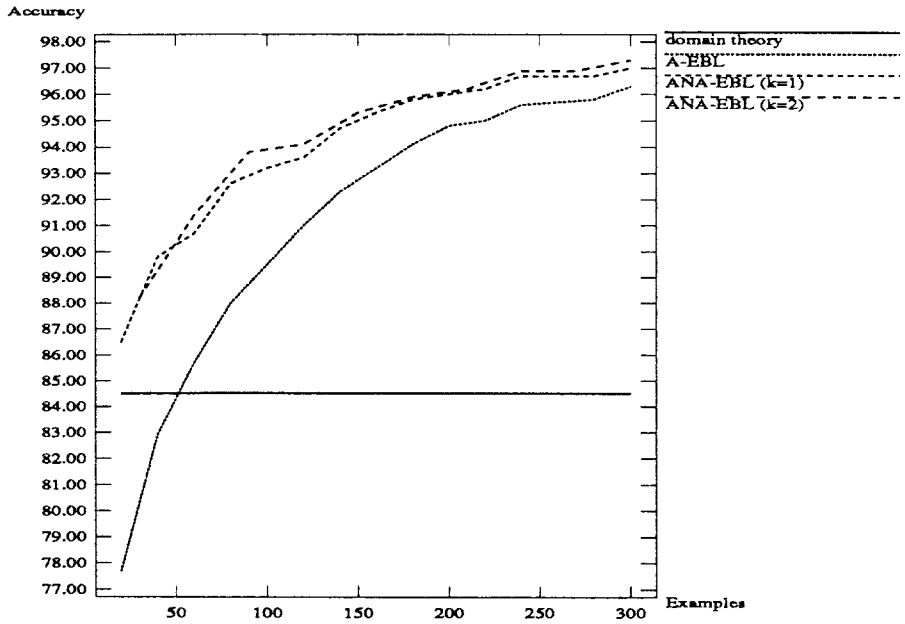

Figure 12. Learning curves for *correct-bid*.

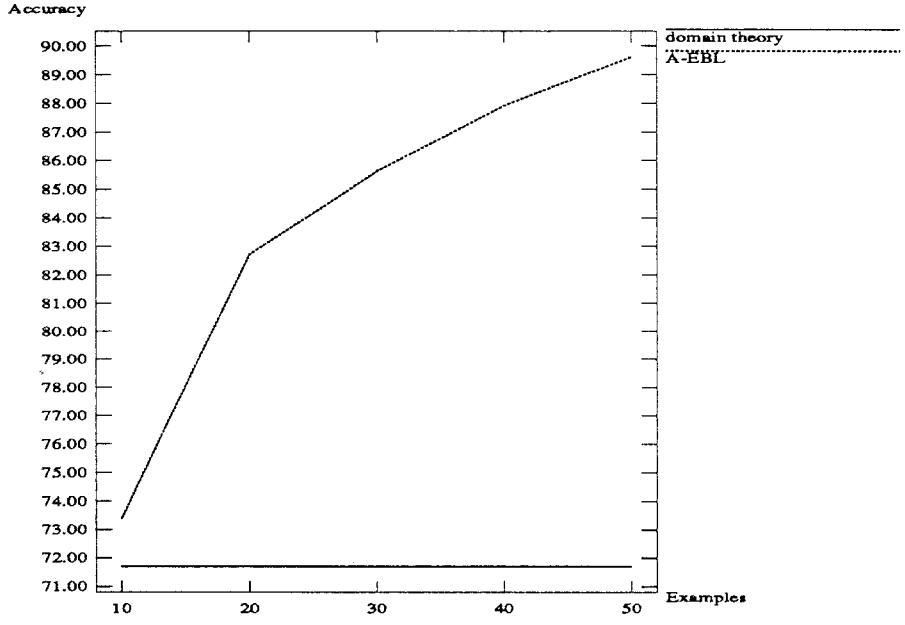

Figure 13. Learning curves for *opening-strength*.

the hands in the test set and comparing the classifications to the correct ones. These experiments were repeated 10 times and the error rates were averaged. The resulting learning curve is shown in Figure 13.

5. Related work

5.1. Other research in EBL

In Pazzani (1988), mechanisms are described that choose between alternative explanations in an imperfect domain theory for plan recognization, which is an abductive task. Pazzani identifies five heuristics for selecting explanations. Some of these were specific to plan recognition domains. One of these, the heuristic of preferring explanations which account for a larger number of observed changes, bears some similarity to the basic method of A-EBL. Similar heuristics for choosing between multiple explanations in the context of completing an incomplete theory are proposed in Fawcett (1989). Our work extends the evaluation heuristics proposed by Pazzani and Fawcett by giving a precise way of weighting the complexity of an explanation and the number of observations that it covers, and justifying this heuristic with a pac-learning analysis.

An advantage of the heuristics used by Pazzani and Fawcett is that they refer to a single explanation and the observations that it explains, and hence applying these heuristics to a set of explanations can be done in time linear in the total size of the set of explanations. In contrast, the heuristics used by A-EBL in the greedy set cover do not refer to a single explanation, and hence A-EBL runs in time that is superlinear in the number of explanations. This suggests that A-EBL's time complexity could be improved by using heuristics such as those suggested by Pazzani and Fawcett to filter the set of candidate explanations and eliminate the least plausible ones.

Pazzani also considers collecting more data to rule out alternative hypothesis via experimentation. In a similar vein, Rajamoney and DeJong (1988) describe a system for choosing between multiple explanations based on experimentation. Their approach is based on a technique called *active explanation reduction.* One can think of active explanation reduction as a way of collecting informative examples; one would expect such a technique to have a lower time and sample complexity of learning than A-EBL. The advantage of A-EBL is that experimentation is not always possible.

Hirsh has used the incremental version-space merging (IVSM) method (Hirsh, 1990) to choose between multiple explanations. Like A-EBL, IVSM is a general technique, with strong formal justifications; it is also incremental. However, the IVSM method requires an additional source of information in the form of a *concept description language* which provides an additional bias to the learning system. The bias toward disjunctive explanations used in A-EBL could be used as a concept description language; however, if this were done, the space requirements of the IVSM algorithm would grow exponentially with the number of explanations per example. IVSM is thus effectively limited to domain theories which generate a small number of explanations per example or to conjunctive concept description languages.

The initial domain theories used by A-EBL and EBL/TS are similar to "plausible low-belief domain theories" described in Rosenbloom and Aasman (1990). However, our mechanisms for making use of these theories are quite different; Rosenbloom and Aasman encode an inductive learning algorithm directly in the domain theory, rather than using inductive postprocessing of the rules created by EBG (as is done in A-EBL.) The learning algorithm they use is an extended version-space algorithm that has many of the same limitations as the algorithm of Hirsh (1990).

At a high level, A-EBL is similar to Flann and Dietterich's *induction over explanations* (IOE) algorithm (Flann & Dietterich, 1989). Both A-EBL and IOE begin with an over-general domain theory and learn a specialization of it. However, IOE learns a specialization that is defined by a single rule, and A-EBL learns a specialization that is defined by multiple rules. IOE also assumes that the domain theory generates only a single explanation for each example, while A-EBL does not. An advantage of IOE is that it can be used to learn co-reference constraints and other constraints on variable values that are not learnable by A-EBL.

5.2. Research in similarity-based learning

The basic procedure that A-EBL uses to specialize a theory is to form a set cover of the positive examples. Set covering techniques have been used in several similarity-based learning system, such as the AQ family of algorithms (Kietterich & Michalski, 1983; Michalski et al., 1986), the CN2 learning algorithm (Clark & Niblett, 1989), and Haussler's algorithm for learning pure disjunctive concepts (Haussler, 1988). In all of these algorithms, the basic procedure is to start with an empty cover and repeatedly add to it some conjunctive set or "complex"; often this complex is one that is chosen to cover as many positive examples as possible. A-EBL differs from the algorithms above in the nature of "complexes" that are added to the set cover: A-EBL adds rules derived by applying EBG to some explanation, rather than rules derived using similarity-based methods on the data. A-EBL also uses slightly different heuristics for choosing a rule to add to the set cover than previous algorithms; A-EBL adds the rule that maximizes the ratio of coverage to size. This makes A-EBL relatively insensitive to the number of rules that could be potentially generated by EBL, but more sensitive to the total size of the concept to be learned.

5.3. Research in abduction

A-EBL bears some similarity to set-covering based implementations of *abduction*, for example, Allemang et al., (1987) and Reggia (1983). The resemblance is especially strong to those abductive systems which use explicit logical theories to define the space of possible abductive explanations of a set of phenomena (Poole, 1988; Reiter, 1987). In fact, learning and abduction are very similar tasks. In both cases, the goal is to come up with a hypothesis which explains a particular set of data points; and in both cases, a simpler hypothesis is desired.

However, the goal of abductive reasoning systems is different from the goal of A-EBL. Set-cover based abductive reasoning systems produce as output the set of *all possible* hypothesis which satisfy some relatively weak definition of minimality (for instance, minimality under the partial order of set inclusion). The criterion of success is whether this set contains all of the most likely hypotheses. A-EBL, in contrast, produces a *single* hypothesis explaining a set of phenomena, uses a relatively strong definition of minimality (small syntactic size, relative to a particular encoding) and incorporates a completely different formal model of correctness—the pac-learnability model. This model guarantees the correctness of *future classifications* made by the hypothesis in a probabilistic sense.

This difference in aims is very important. Put another way, A-EBL differs from abductive reasoning systems in that it has a weaker goal. The goal of abduction is to find a set of *most plausible* explanations for a set of phenomena, given a set of possible explanations. A-EBL finds instead a set of explanations which are *probabilistically guaranteed to make good predictions,* if the phenomena that A-EBL are asked to predict are drawn from the same population from which training instances were taken. In satisfying this goal, A-EBL does quite well; its sample complexity is within a logarithmic factor of a known lower bound.

There are two other less important differences between A-EBL and abductive reasoning systems. First, most abductive reasoning systems confirm that the assumptions made in completing a proof are mutually consistent; in A-EBL, however, no such "consistency check" is performed. A-EBL can be thought of as using the examples to probabilistically check that the set of explanations is "consistent" in the (somewhat weaker) sense that they lead to correct predictions; one justification for this weak consistency check is that sound and complete consistency checks are usually computationally intractable. A second difference is that no parts of the theory used by A-EBL are explicitly marked as "assumptions": instead, every explanation is viewed as potentially incorrect. This simplification reflects our own research emphases, and could probably be relaxed by slight modifications to the set cover algorithm without disturbing our results.

A second connection between EBL with multiple explanations and abduction is that a sufficiently powerful abductive reasoning system would be a solution to the multiple explanation problem, since it could be used to determine directly which explanation of an example is correct. In most domains, however, abductive reasoning does not produce a single explanation of a set of phenomena, but a fairly large set. Our research can be viewed as an effort to sidestep the difficulties imposed by the existence of only weak abductive reasoning systems: by relaxing the problem of abduction to the problem of ensuring accurate prediction on later problems, we are able to learn effectively from a weak, explanatory domain theory *without* solving the problem of abduction.

6. Further work

The experiments reported in this paper used a fairly simple and somewhat artificial domain. Another disadvantage of the domain is that it was used as a test case in developing A-EBL, and hence is not a good prospective test of the learning system. Further experimentation in larger and more realistic domains would be desirable. One obstacle in applying

A-EBL to real-world domains is that A-EBL is sensitive to noise; extending A-EBL to deal with noise is an important issue for future research.

Another drawback of our algorithm is that it is non-incremental. Finding an incremental version of the set-covering algorithm on which A-EBL is based would make the algorithm applicable in a wider range of circumstances, and is another topic for further research. Some initial results on this problem have been reported in Cohen (1990b, Chapter 5).

Another topic for later research is learning rules with *exceptions*, that is rules which are applicable only if some other set of rules are not applicable. Our algorithm cannot learn rules which have exceptions unless the domain theory explicitly encodes the exceptions to every rule. An interesting research topic would be to extend the set cover algorithm to discover this information, and thus automatically learn rules with exceptions from a simpler domain theory.

Another set of topics to investigate are further applications of the A-EBL technique. We would like to investigate the usefulness of A-EBL is using a weak theory to complete an incomplete domain theory, as in Hall (1988); Mahadevan (1989); and Roy and Mostow (1988). Since multiple explanations can easily arise in these situations, this seems like a natural application of A-EBL. A second application is weakening the bias of explanation-based learning system by considering "abstracted" explanations, as was done in Experiment 2. A final application of A-EBL (of special interest to us because it was one of the original motivation for pursuing this research) is learning control rules for search programs. Some initial results in this area have been reported in Cohen (1990c).

7. Conclusion

A much-investigated research topic in machine learning is using prior knowledge of a learning problem to improve the performance of similarity-based learning techniques. One approach to this problem is to attempt to extend explanation-based learning (EBL) methods to *imperfect* domain theories: that is, domain theories that are not a complete and correct description of the concept to be learned. One problem that many types of imperfect theories demonstrate is the *multiple inconsistent explanation problem*. This problem occurs when a domain theory produces multiple explanations for a training instance, only some of which are correct. This paper has described an extention of EBL called A-EBL that handles the multiple explanation problem. A-EBL makes use of additional positive examples and negative examples to choose among the multiple explanations.

We have evaluated the behavior of A-EBL in two ways. A *formal* evaluation was used to show that the technique will scale well in efficiency and required sample size. An *empirical* evaluation was used to demonstrate that it can succeed in some learning tasks of interest using a reasonable number of representative examples and a modest amount of CPU time.

The formal evaluation is based on Valiant's definition of probably approximately correct learnability; this theory is applicable since both A-EBL and EBL/TS (the formalization of "standard" explanation-based learning with which we compare A-EBL) can be viewed as inductive learning systems which learn a particular specialization of an initial domain theory. We compute the sample complexities of A-EBL and EBL/TS, and show that they

are within a logarithmic factor of one another; in other words, that the presence of multiple explanations requires only a modest increase in the number of examples necessary to learn with high probability an approximately correct description of an unknown concept. These worst case sample complexities are shown to be tight, within a logarithmic bound. We also show that A-EBL runs in time polynomial in the complexity of the concept to be learned and the total number of explanations, and that A-EBL is "competent" in a strictly broader range of circumstances.

The empirical evaluation is in the domain of bridge opening bids, and is based on a database of forty-three well-chosen, representative examples, as well as on larger randomly-generated databases, and several different non-trivial domain theories. The domain theories and examples used by A-EBL closely parallel the information given in the text of an introductory book on playing bridge (Sheinwold, 1964). All the theories suffer from the multiple explanation problem. To use these theories, A-EBL has to make many of the same inferences a person would, including inferring the meaning of undefined terms from background information and examples, and inferring the exact conditions under which heuristic bidding rules should be applied from knowledge of these rules and examples. In these experiments, A-EBL is successful in improving the accuracy of the original domain theories when given either randomly selected examples or "textbook examples" taken from Sheinwold (1964).

8. Acknowledgments

This paper benefitted greatly from discussions with Alex Borgida, Tom Dietterich, Haym Hirsh, Thorne McCarty, Jack Mostow, Shane Bruce, Phil Franklin, Russell Greiner, Chun Liew, Mike Pazzani, Cullen Schaffer, and many other colleagues. Comments by Paul Rosenbloom and three anonymous reviewers were extremely helpful; I would also like to thank Susan Cohen for proofreading a draft of this paper. The early stages of this research were done while the author was attending Rutgers University and receiving a Marion Johnson Fellowship; later stages were done while the author was attending Rutgers University and receiving an AT&T Bell Laboratories Fellowship.

Notes

1. This terminology was adopted when EBL was used for the purpose of finding a more *efficiently useable* representation of the concept defined by the original domain theory; therefore, many of the terms have unfortunate connotations for inductive versions of EBL. However, they are so widely used that we hesitate to change them substantially. In particular, the term "operational" is misleading; the subgoals that the $\mathcal{O}$ predicate should succeed on are those subgoals which are supported by subproofs whose exact structure is irrelevant to the concept being learned. Also, the term "target concept" or "goal concept" is frequently used for the concept C_T defined by the domain theory; in the inductive learning literature, however, target concept refers to the unknown concept C that the learner is trying to discover. To avoid confusion, we avoid all use of the term "target concept."
2. The main difference in the two definitions is that in the theory specialization problem negative information may be used. The term theory specialization was also chosen to emphasize the fact that the unknown concept C might be defined by a set of rules, rather than a single rule.

3. Readers unfamiliar with bridge are referred to Appendix A.
4. This definition assumes that if a node is operational, every descendant of that node is also operational.
5. Or nearly the same set of proofs; one important difference between using abduction on the incomplete theory and using standard theorem proving on the completed theory is that many abductive reasoning systems confirm that the assumptions made in completing a proof are mutually consistent.
6. The name of a clause is just some unique identifier associated with a clause. Use of this representation is possible because variable binding information (and hence the generalization produced by EBG) can be recovered from a tree which is labeled only with clause names; one technique for doing this is described in Appendix B.
7. Actually, the A-EBL algorithm described in this paper is embedded in ELGIN, a larger and more flexible learning system (Cohen, 1990b); the statistics above reflect the code in ELGIN used directly in running A-EBL.
8. The theory actually used in the experiments was modified slightly to make it possible to use a simple backchaining theorem prover; the modification is discussed in Appendix E.
9. Notice that this encoding takes advantage of the fact that A-EBL can easily learn a different specialization of a predicate for each context in which that predicate is used; in this case, the specialization of the predicate *somewhat-large* as used in *length-in-majors* will be different form the specialization of the predicate *somewhat-large* as used in *rebiddable*. This is a side effect of the fact that A-EBL learns an operationalized theory.
10. CPU times for ANA-EBL with $k = 2$ are not given because these experiments were performed on a different version of Prolog and a different machine.
11. Using a z-test on the differences of the error rates of ANA-EBL using $k = 1$ and $k = 2$ yields $p > 99.95\%$ ($z = 3.53$) that the difference is real.
12. Ideally, they would be learned by another application of A-EBL (as the concept *opening-strength* will be learned, as described in Section 4.4), but, because they are only used in exceptional circumstances, only one or two examples were available for each; hence we were forced to hand-code these definitions.

Appendices

A. Summary of the game of bridge

Bridge is a card game, somewhat similar to whist, spades, or hearts, and is played by two partnerships of two persons each. The object in play is to *take tricks*, i.e., to capture the other players' cards. Tricks are usually taken with a high card (like the ace, king or queen) in the suit led or a card in the suit designated as the *trump suit*. Play is preceded by an *auction* in which partnerships compete for the right to name the trump suit. Each *bid* in the auction is a number and a suit name, or a number and the word *no-trump*, which means that there will be no trump suit. The heart and spade suits are called *major suits* and the club and diamond suits are *minor suits*. More points are awarded for tricks taken with a major suit as trump.

The first bid is called an *opening bid*. Subsequent bids must be "higher": either they name a higher suit, or a higher number, or both. Suits are ordered (from lowest to highest) as follows: clubs, diamonds, hearts, spades and no-trump. Players bid in order going around the table; if a player does not want to bid, he may *pass*. The auction ends when there have been three consecutive passes. The partnership making the highest bid is then obligated to take a certain number of tricks, or else they will be penalized. The trump suit will be the trump suit named by the highest bid; the number of tricks that must be taken is related to the number of the highest bid.

To decide what to bid, players compute certain evaluation metrics on their hand. One of these is *high card points*, which is related to the number of face cards in the hand: an

ace counts for four high-card points, a king three, a queen two, and a jack one. Another measure is *quick tricks,* which counts certain configurations of face cards: for instance, the ace and king of the same suit count for two quick tricks.

Bidding is a complicated problem. Typically, a partnership can only make two or three bids before the auction gets too high; based on this tiny amount of information, they must decide what the collective long suit of the partnership is (to name the trump suit appropriately) and how strong the two hands are together. To maximize the amount of information conveyed by each bid, partnerships always adopt some sort of *bidding system*, in which each bid is given some conventional meaning. Needless to say, making a bid which is incorrect according to the agreed-on bidding system is a grievous error. These bidding systems are usually quite complex. The one used in our examples is a variant of one of the simplest, called *Standard American*, as presented in Sheinwold (1964). Sheinwold devotes 230 pages to bidding, and 40 pages to normal (non-forcing and non-preemptive) opening bids alone.

B. Algorithm for recovering binding information

Our implementation of A-EBL does not represent abstract explanation structures directly, but uses a tree whose nodes are labeled with clause names, where the *name* of a clause is just some unique identifier associated with a clause. This appendix describes how an abstract explanation structure can be recovered from this data structure, which we will call a "name tree."

Let a *naming relation* be some function N from a finite set of symbols Σ onto the clauses of domain theory T. Let τ be a name tree with labels drawn from the set $\Sigma \cup \{\textbf{nil}\}$, where **nil** indicates an operational leaf of the tree. Code for constructing an abstract explanation structure from τ is given below, in the form of a function *convert-name*. This function returns two values, a tree and its associated substitution. The top level invocation should use a variabilized version of the predicate being specialized (e.g., *plausible-bid*(x, y)) as the second argument.

Algorithm convert-name(τ, G):

if τ is labeled **nil then**
    return a tree with a single node labeled G,
        and the empty subsitution { }

else
    let $\tau_1, \ldots, \tau_l$ be the sons of τ
    let "$A \leftarrow B_1, \ldots, B_k$" be $N(l)$, where l is the label of the root of τ
    let θ_0 be the mgu of G and A
    let $\theta \leftarrow \theta_0$
    for $i = 1, \ldots, k$ **do**
        $\eta_i, \theta_i \leftarrow$ convert-name(τ_i, $B_i\theta$)

let $\theta \leftarrow \theta \circ \theta_i$
endfor
return a tree with root labeled $G\theta$ and subtrees $\eta_1\theta, \ldots, \eta_k\theta$,
and the substitution θ
endif

To make the definitions and algorithm of the preceding section more concrete, we reproduce below Prolog code for the two operations actually performed in our system on name trees. The predicate *aes* is a meta-interpreter that proves a goal G and returns an abstract explanation structure for G (in name tree form). The predicate *peval* "partially evaluates" an abstract explanation structure in name tree form, producting a Horn clause.

aes(+*G*, −*Ag*) ← *Ag is an abstract explanation structure for G*

aes(G,nil) ← operational(G),!,call(G).
aes((G,H),(Ag,Ah)) ← !,aes(G,Ag),aes(H,Ah).
aes(G,(Id ← Ah)) ← !,theory-clause(G,H,Id),aes(H,Ah).

peval(+*AES*, −*Clause*) ← *partially evaluate an abstract explanation structure, producing a clause for a logic program*

peval(AES,(A ← B)) ← goal-formula(A),peval(A,AES,B).

peval(G,nil,G) ← !.
peval((G,H),(Q,R),Body) ← !,
peval(G,Q,Body1), peval(H,R,Body2),combine(Body1,Body2,Body).
peval(G,(Id ← Q),Body) ← !,
theory-clause(G,H,Id), peval(H,Q,Body).

theory-clause(*G,H,Id*) ← (*G* ← *H*) *is a clause of the domain theory with name Id*
combine(*T1,T2,T3*) ← *T3 is the conjunction of T1 and T2*

C. Training data

	Hand	Recommended Bids
1	♠ J86532 ♡ A ◇ KQ9643 ♣ –	1 spade
2	♠ KQ874 ♡ KQJ653 ◇ A ♣ 3	1 heart
3	♠ KQ874 ♡ QJ8653 ◇ A ♣ 3	1 spade
4	♠ KJ63 ♡ AKJ63 ◇ AK2 ♣ 4	1 heart
5	♠ KQ1085 ♡ KQ2 ◇ Q76 ♣ 43	1 spade
6	♠ KQ108 ♡ KQ2 ◇ Q765 ♣ 43	pass
7	♠ AQJ85 ♡ K10974 ◇ 43 ♣ 6	1 spade
8	♠ KQ73 ♡ KJ75 ◇ AJ84 ♣ 6	1 heart
9	♠ AQJ85 ♡ A974 ◇ 43 ♣ 62	1 spade
10	♠ KQJ75 ♡ 5 ◇ AJ963 ♣ 62	1 spade
11	♠ J852 ♡ AQ2 ◇ AK8 ♣ 762	1 diamond
12	♠ KJ63 ♡ AKJ63 ◇ 52 ♣ 84	1 heart
13	♠ AK63 ♡ KQ532 ◇ 52 ♣ 84	1 spade
14	♠ KQ63 ♡ 52 ◇ AK863 ♣ 84	1 diamond
15	♠ AK87 ♡ AQJ9 ◇ 62 ♣ 975	1 spade
16	♠ A2 ♡ KQ87 ◇ AJ94 ♣ 975	1 heart
17	♠ KQ73 ♡ KJ75 ◇ 6 ♣ AJ84	1 club
18	♠ KQ73 ♡ 6 ◇ KJ75 ♣ AJ84	1 club or 1 diamond
19	♠ 6 ♡ KQ73 ◇ KJ75 ♣ AJ84	1 club or 1 diamond
20	♠ KQ87 ♡ A2 ◇ 975 ♣ AJ94	1 club
21	♠ A2 ♡ KQ87 ◇ 975 ♣ AJ94	1 club
22	♠ KQJ75 ♡ 5 ◇ 62 ♣ AJ963	1 club
23	♠ J852 ♡ AQ2 ◇ 762 ♣ AK8	1 club
24	♠ KQ63 ♡ 52 ◇ 84 ♣ AK863	1 club
25	♠ KQJ75 ♡ 5 ◇ A2 ♣ AK963	1 spade
26	♠ 43 ♡ 6 ◇ AQJ85 ♣ K10974	pass
27	♠ QJ852 ♡ QJ7 ◇ QJ6 ♣ KJ	pass
28	♠ KJ3 ♡ KJ4 ◇ A3 ♣ Q6432	1 club
29	♠ KJ3 ♡ KJ4 ◇ QJ3 ♣ QJ64	1 club
30	♠ KQ8 ♡ AQ74 ◇ KQ92 ♣ 54	1 heart
31	♠ KQ8 ♡ AQ74 ◇ KQ2 ♣ 954	1 no-trump
32	♠ K7 ♡ KQ873 ◇ AJ94 ♣ A5	1 heart
33	♠ AJ8 ♡ K4 ◇ AJ932 ♣ AJ5	1 diamond
34	♠ KJ5 ♡ AQ9 ◇ AKJ4 ♣ AJ6	2 no-trump
35	♠ AQ965 ♡ K104 ◇ AQ8 ♣ AK	2 no-trump
36	♠ KQ5 ♡ AQ9 ◇ AKJ4 ♣ AQ6	3 no-trump
37	♠ J52 ♡ AKJ ◇ AKJ ♣ AQ82	1 club
38	♠ AQ852 ♡ KJ5 ◇ AJ ♣ K74	1 spade
39	♠ AQ85 ♡ KJ5 ◇ AQ3 ♣ K74	1 club
40	♠ QJ8 ♡ AJ93 ◇ K92 ♣ KJ5	1 club or 1 heart
41	♠ KJ8 ♡ QJ74 ◇ A62 ♣ KQ5	1 no-trump
42	♠ KJ82 ♡ QJ74 ◇ AQ ♣ KQ5	1 no-trump
43	♠ AQ6 ♡ K62 ◇ K5 ♣ KQ1042	1 no-trump

D. Test data

	Hand	Recommended Bids
1	♠ AKJ10876 ♡ A762 ◇ K2 ♣ –	1 spade
2	♠ J9642 ♡ AKQ85 ◇ A5 ♣ 3	1 spade
3	♠ KQJ1094 ♡ K93 ◇ 54 ♣ 63	pass
4	♠ 108643 ♡ AKJ ◇ AQ6 ♣ 63	1 spade
5	♠ AJ1084 ♡ K93 ◇ AQ6 ♣ 63	1 spade
6	♠ AKQ ♡ 107652 ◇ AKQJ ♣ A	1 heart
7	♠ KJ642 ♡ A5 ◇ 3 ♣ AQ732	1 club
8	♠ KQJ64 ♡ A5 ◇ 3 ♣ AQ732	1 spade
9	♠ AJ4 ♡ 9632 ◇ AK10 ♣ AQJ	1 club or 1 diamond
10	♠ K10942 ♡ AKJ ◇ 85 ♣ K109	1 spade
11	♠ K1094 ♡ AKJ ◇ Q52 ♣ K109	1 no-trump
12	♠ K1094 ♡ AKJ ◇ 85 ♣ K1094	1 club
13	♠ Q64 ♡ A52 ◇ AQ5 ♣ KJ32	1 no-trump
14	♠ 64 ♡ AQJ2 ◇ AQ5 ♣ KJ32	1 club or 1 heart
15	♠ AQ ♡ AQJ2 ◇ AQ5 ♣ KJ32	2 no-trump
16	♠ AQJ ♡ AQJ2 ◇ AQ5 ♣ KQ3	3 no-trump

E. Domain theories

E.1. Theory for plausible opening bids

This appendix presents the non-operational portions of the domain theory for opening bids used in the experiments. Low-level routines (such as the routines to count a hand, etc.) are not included. The theory is given in Prolog syntax.

The top-level predicate *plausible-bid* contains a series of ten clauses that explain how to open several types of hands. The first eight are straightforward translations of rules from the text.

```
plausible-bid(Hand,bid(pass)) ←
        not opening-strength(Hand).

plausible-bid(Hand,bid(1,Suitl)) ←
        not one-suited(Hand),
        not notrump(Hand),
        opening-strength(Hand),
        short-minor(Suitl,Hand).

plausible-bid(Hand,bid(1,Suitl)) ←
        not two-suited(Hand),
        not notrump(Hand),
        opening-strength(Hand),
        biddable(Suitl,Hand).
```

```
plausible-bid(Hand,bid(1,c)) ←
        hcp(Hand,15),
        balanced-distribution(Hand),
        almost-all-suits-stopped(Hand).

plausible-bid(Hand,bid(1,notrump)) ←
        not too-strong-for-1nt(Hand),
        hcp(Hand,HCP),
        between(HCP,16,18),
        balanced-distribution(Hand),
        almost-all-suits-stopped(Hand).

plausible-bid(Hand,bid(1,Suit)) ←
        hcp(Hand,HCP),
        between(HCP,19,21),
        balanced-distribution(Hand),
        almost-all-suits-stopped(Hand),
        short-minor(Suit,Hand).

plausible-bid(Hand,bid(2,notrump)) ←
        hcp(Hand,HCP),
        between(HCP,22,24),
        balanced-distribution(Hand),
        all-suits-stopped(Hand).

plausible-bid(Hand,bid(3,notrump)) ←
        hcp(Hand,HCP),
        between(HCP,25,27),
        balanced-distribution(Hand),
        all-suits-stopped(Hand).
```

The top-level rules for choosing between two and three biddable suits are given below.

```
plausible-bid(Hand,bid(1,Suit)) ←
        not three-suited(Hand),
        not notrump(Hand),
        opening-strength(Hand),
        biddable(Suit1,Hand),
        biddable(Suit2,Hand),
        Suit1 ≠ Suit2,
        prefer(Hand,Suit,Suit1,Suit2).
```

```
plausible-bid(Hand,bid(1,Suit)) ←
        opening-strength(Hand),
        biddable(Suit1,Hand),
        biddable(Suit2,Hand),
        biddable(Suit3,Hand),
        Suit1 ≠ Suit2,
        Suit1 ≠ Suit3,
        Suit2 ≠ Suit3,
        prefer(Hand,Suit,Suit1,Suit2,Suit3).
```

There are two rules for choosing among three possible suits: to prefer the *middle* suit, or to prefer the *lower* suit.

```
prefer(Hand,Suit,Suit1,Suit2,Suit3) ←
        middle-suit(Suit,Suit1,Suit2,Suit3).
prefer(Hand,Suit,Suit1,Suit2,Suit3) ←
        lowest-suit(Suit,Suit1,Suit2,Suit3).
```

The rules for choosing among two possible suits are more complicated than in our simple example of Section 2.2. First, there are two "exceptional rules," introduced on pages of 20 and 21 of Sheinwold (1964): to prefer a strong 5-card suit to a weaker 4-card suit, and to prefer clubs to spades given a weak hand. If neither of these exceptions hold, then any of five general rules can be used: to prefer the longer suit, the lower suit, the higher suit, the higher of two touching suits, or the lower of two non-touching suits.

```
prefer(Hand,Suit,Suit1,Suit2) ←
        prefer-exception(Hand,Suit,Suit1,Suit2).
prefer(Hand,Suit,Suit1,Suit2) ←
        not prefer-exception(Hand,Suit,Suit1,Suit2),
        prefer-default(Hand,Suit,Suit1,Suit2).

prefer-exception(Hand,Suit1,Suit1,Suit2) ←
        strong-5-over-weak-4-card-suit(Hand,Suit,Suit1,Suit2).
prefer-exception(Hand,Suit,Suit1,Suit2) ←
        weak(Hand),
        clubs-over-spades((suit,Suit1,Suit2).

prefer-default(Hand,Suit,Suit1,Suit2) ·
        longer(Hand,Suit,Suit1,Suit2).
prefer-default(Hand,Suit,Suit1,Suit2) ←
        higher(Suit,Suit1,Suit2).
prefer-default(Hand,Suit,Suit1,Suit2) ←
        lower(Suit,Suit1,Suit2).
```

```
prefer-default(Hand,Suit,Suit1,Suit2) ←
        higher-and-touching(Suit,Suit1,Suit2).
prefer-default(Hand,Suit,Suit1,Suit2) ←
        lower-and-not-touching(Suit,Suit1,Suit2).
```

The rules *strong-5-over-weak-4-card-suit* and *weak* required some judgment to define.[12] The former predicate was defined to succeed when the five-card suit has more high card points than the four-card suit. The latter predicate was defined to succeed on hands with less than or equal to 14 high card points; this cutoff was selected by choosing the halfway point between the high-card count of the single positive example and the single negative example.

The top-level rules for opening strength are as given in Figure 11; the following hand-coded rules for *comfortable-rebid* and *length-in-majors* were used.

```
length-in-majors(Hand) ←
        length(s,Hand,NS),
        length(h,Hand,NH),
        N is NS+NH,
        N ≥ 8.
length-in-majors(Hand) ←
        major-suit(Suit),
        length(Suit,Hand,N),
        N ≥ 5.
comfortable-rebid(Hand) ←
        rebiddable(_,Hand).
comfortable-rebid(Hand) ←
        biddable(Suit1,Hand),
        biddable(Suit2,Hand),
        Suit1 ≠ Suit2.

rebiddable(Suit,Hand) ←
        suit(Suit),
        length(Suit,Hand,N),
        N ≥ 5.
```

Finally, the rules for deciding when a suit is *biddable* are as follows. The second rule simplifies the corresponding rule in Sheinwold (1964): while he states that a four-card suit must be "QJxx or better," the rule simply checks that there are three or more high card points.

```
biddable(Suit,Hand) ←
        suit(Suit),
        length(Suit,Hand,N),
        greater-than-or-equal(N,5).
```

```
biddable(Suit,Hand) ←
        suit(Suit),
        length(Suit,Hand,4),
        high-card-points(Suit,Hand,HCP),
        HCP ≥ 3.

greater-than-or-equal(N,N).
greater-than-or-equal(N,M) ←
        N1 is N−1,
        greater-than-or-equal(N1,M).
```

The decision to use the logical predicate *greater-than-or-equal-to* to make the comparison in the first clause of the *biddable* predicate, rather than the extralogical predicate "≥," represents a fairly subtle choice of the operationality predicte. In general, we classify the predicate ≥ as operational; however, this is inappropriate in this case because we would like the proofs of biddability for suits of different lengths to be distinct. (For example, in most certain circumstances a six-card suit will be treated differently from a five-card suit.) Hence the non-operational comparison predicate *greater-than-or-equal-to* is used.

Making such subtle choices in operationality is undesirable, since it gives a hint to the learning system. In fact, if one's goal is to generate a learning problem which contains exactly the information present in the textbook and no additional information, introduction of any operationality predicate is somewhat problematic, since this information is not given explicitly in the textbook. However, in this use of EBL, an operationality predicate represents knowledge about what features are relevant to the concept to be learned; it can be plausibly argued that this knowledge could be inferred by a reader. For instance, in this case, marking *greater-than-or-equal-to* operational here would require the learning algorithm (either EBL/TS or A-EBL) to treat all suits of length greater than four identically; this would essentially suppress the feature of suit-length. However, in Sheinwold's informal discussions, the length of the biddable suits is sometimes mentioned; this suggests that this feature is relevant to learning. Another hint to the reader that suit length is relevant in bidding is that Sheinwold tends to group his examples by the length of their suits.

The number of high card points needed to decide if a suit is biddable as a short minor in the final rule was also a subjective evaluation, based on knowledge not found in Sheinwold (1964).

```
short-minor(Suit,Hand) ←
        minor-suit(Suit),
        hcp(Suit,Hand,HCP),
        HCP ≥ 3,
        length(Suit,Hand,N),
        N ≥ 3.
```

The remaining predicates are marked as operational.

E.2. Theory for opening strength

The theory for opening strength is as presented in Section 4.4 with one minor change. Recall the rules defining the predicate *somewhat-larger*.

somewhat-larger(N,M) ⇒ somewhat-larger(N+1,M)
N > M ⇒ somewhat-larger(N,M)

Given any two numbers N and M, there are only a finite number of proofs that *somewhat-larger*(N,M) holds: however, the simple Prolog meta-interpreter that we use as a theorem-prover loops when it is given this theory. To prevent looping, it was necessary to add the additional conjunct "N > M" to the recursive clause of *somewhat-larger*. The actual definition of *somewhat-larger* used in the experiments was the following.

somewhat-larger(N,M) ← N > M.
somewhat-larger(N,M) ← N > M, N1 is N−1, somewhat-larger(N1,M).

References

Allemang, D., Tanner, M., Bylander, T. & Josephson, J. (1987). On the computational complexity of hypothesis assembly. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence,* Milan, Italy. Morgan Kaufmann.

Blumer, A., Ehrenfeucht, A., Haussler, D. & Warmuth, M. (1986). Classifying learnable concepts with the Vapnik-Chervonenkis dimension. In *Eighteenth Annual Symposium of the Theory of Computing.* ACM Press.

Carbonell, J., Mason, M., & Mitchell, T. (1987). Machine learning and planning in a reactive environment. In *Proceedings of the NASA AI Forum.*

Chvátal, V. (1979). A greedy heuristic for the set covering problem. *Mathematics of Operations Research,* 4.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning,* 3.

Cohen, W. (1990a). An analysis of representation shift in concept learning. In *Proceedings of the Seventh International Conference on Machine Learning,* Austin, Texas. Morgan Kaufmann.

Cohen, W. (1990b). Concept learning using explanation based generalization as an abstraction mechanism. (Technical Report DCS-TR-271). Rutgers University.

Cohen, W. (1990c). Learning approximate control rules of high utility. In *Proceedings of the Seventh International Conference of Machine Learning,* Austin, Texas. Morgan Kaufmann.

Cohen, W. (1990d). Learning from textbook knowledge: A case study. In *Proceedings of the Eighth National Conference on Artificial Intelligence,* Boston, Massachusetts. MIT Press.

DeJong, G. & Mooney, R. (1986). EBL: An alternative view. *Machine Learning,* 1.

Dietterich, T. & Michalski, R. (1983). A comparative review of selected methods for learning from examples. In *Machine learning: An artificial intelligence approach.* Morgan Kaufmann.

Dietterich, T. (1986). Learning at the knowledge level. *Machine Learning,* 1.

Drastal, G., Czako, G. & Raatz, S. (1989). Induction in abstraction spaces: A form of constructive induction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, Michigan. Morgan Kaufmann.

Ehrenfeucht, A., Haussler, D., Kearns, M. & Valiant, L. (1988). A general lower bound on the number of examples needed for learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory,* Boston, Massachusetts.

Fawcett, T. (1989). Learning from plausible explanations. In *Proceedings of the Sixth International Workshop on Machine Learning,* Ithaca, New York. Morgan Kaufmann.

Flann, N. & Dietterich, T. (1989). A study of explanation-based methods for inductive learning. *Machine Learning*, 4.

Ginsberg, A. (1988). Theory revision via prior operationalization. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, Minnesota. Morgan Kaufmann.

Gold, M. (1967). Language identification in the limit. *Information and Control*, 10.

Hall, R. (1988). Learning by failing to explain: Using partial explanation to learn in incomplete or intractible domains. *Machine Learning*, 1.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36.

Hirsh, H. (1988). Empirical techniques for repairing imperfect theories. In *Proceedings of the 1988 Spring Symposium on EBL*, Palo Alto, California. AAAI.

Hirst, H. (1989). Combining empirical and analytic learning with version spaces. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York. Morgan Kaufmann.

Hirst, H. (1990). *Incremental version space merging: A general framework for concept learning.* Kluwer Academic Publishers.

Huhns, M. & Acosta, R. (1987). ARGO: An analogical reasoning system for solving design problems. (Technical Report AI/CAD-092-87). MCC.

Johnson, D.S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9.

Kedar-Cabelli, S. (1987). Formulating concepts according to purpose. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, Washington. Morgan Kaufmann.

Kelly, K. (1988). Theory discovery and the hypothesis language. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, Michigan. Morgan Kaufmann.

Laird, J. (1988). Recovery from incorrect knowledge in SOAR. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, Minnesota. Morgan Kaufmann.

Mahadevan, S. (1985). Verification-based learning: A generalization strategy for inferring problem-reduction methods. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California. Morgan Kaufmann.

Mahadevan, S. (1989). Using determinations in EBL: A solution to the incomplete theory problem. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York. Morgan Kaufmann.

Michalski, R.S., Mozetic, I., Hong, J. & Lavrac, N. (1986). The multipurpose incremental learning system AQ15 and its application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania. Morgan Kaufmann.

Mitchell, T., Keller, R. & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1.

O'Rorke, P., Morris, S. & Schulenburg, D. (1989). Theory formation by adbuction: Initial results of a case study based on the chemical revolution. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York. Morgan Kaufmann.

Pazzani, M. (1988). Selecting the best explanation in explanation-based learning. In *Proceedings of the 1988 Spring Symposium on Explanation Based Learning*, Palo Alto, California. AAAI.

Poole, D. (1988). A logical framework for default reasoning. *Artificial Intelligence*, 36, 27–47.

Rajamoney, S. & Dejong, G. (1988). Active explanation reduction: An approach to the multiple explanation problem. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, Michigan. Morgan Kaufmann.

Reggia, J. (1983). Diagnostic expert systems based on a set-covering model. *International Journal of Man-Machine Studies*, 5.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32, 57–95.

Rosenbloom, P. & Aasman, J. (1990). Knowledge level and inductive uses of chunking (EBL). In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, Massachusetts. MIT Press.

Roy, S. & Mostow, J. (1988). Parsing to learn fine grained rules. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, Minnesota. Morgan Kaufmann.

Sheinwold, A. (1964). *5 weeks to winning bridge.* Simon & Schuster.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM*, 27.