UC Irvine
ICS Technical Reports

Title
The utility of knowledge in inductive learning

Permalink
https://escholarship.org/uc/item/2xd1k9bk

Authors
Pazzani, Michael Pazzani
Kibler, Dennis Kibler

Publication Date
1990-06-25

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Notice: This Material
may be protected
by Copyright Law
(Title 17 U.S.C.)

Z
699
C3
no. 90-18

The Utility of Knowledge in Inductive Learning

Michael Pazzani Dennis Kibler

Department of Information and Computer Science
University of California, Irvine, CA 92717

June 25, 1990

Technical Report 90-18

This research is partially supported by NSF Grant IRI-8908260. We would like to thank Ross Quinlan for his advice on FOIL and Tim Cain, Caroline Ehrlich, Ross Quinlan, Wendy Sarrett and Glenn Silverstein for reviewing a draft of this paper.

The Utility of Knowledge in Inductive Learning

Michael Pazzani
pazzani@ics.uci.edu

Dennis Kibler
kibler@ics.uci.edu

Department of Information & Computer Science
University of California, Irvine
Irvine, CA 92717 U.S.A.
(714) 856-5951

TOPIC- MACHINE LEARNING

Abstract

In this paper we demonstrate how different forms of background knowledge can be integrated with a top-down inductive method for generating constant-free Horn-clause theories. Furthermore, we evaluate, both theoretically and empirically, the value that these types of knowledge will have on the cost of generating a correct Horn-clause theory. The taxonomy of background knowledge and associated costs provides a useful guide in directing the acquisition of knowledge. Moreover, we demonstrate that a hybrid explanation-based and inductive learning method can advantageously use an approximate domain theory, even when this theory is incorrect and incomplete.

1 Introduction

Existing systems that combine empirical and explanation-based learning either severely restrict the domain theory (e.g. to attribute-value pairs (Lebowitz, 1986; Danyluk, 1989), or to unary predicates (Hirsh, 1989; Mooney & Ourston, 1989; Katz, 1989; Shavlik & Towell, 1989; Pazzani, 1989; Sarrett & Pazzani, 1989)) or they acquire general relational concepts but only under very restrictive circumstances (e.g., OCCAM (Pazzani, 1988), IOE (Flann & Dietterich, 1989), ML-SMART (Bergadano, Giordana, & Ponsero, 1989)). Such simplistic domain theories, necessitated in part by the empirical learning component, reduce the expressiveness of explanation-based learning and the applicability of the resulting integrated learning system.

A recent advance in Horn-clause concept learning, FOIL (Quinlan, 1989 & Quinlan, in press), eliminates the need for limiting the expressiveness of domain theories. In this paper, we analyze the complexity of FOIL in terms of the size of hypothesis space generated and tested during learning. We describe how FOIL can be extended to use a variety of background knowledge either to increase the class of problems that can be solved or to decrease the hypothesis space explored. We introduce a new learning system, called SaranWrap, that uses an information-based metric to evaluate extensions to (possibly null) hypotheses of a concept definition. The extensions may be proposed either by an inductive component or by an explanation-based component. We demonstrate that the resulting system can make use of domain knowledge when available to constrain the search for a hypothesis and can utilize incomplete and incorrect domain theories in a uniform manner.

A secondary goal of this paper is to create a taxonomy of various types of background knowledge and show the effect that each type of knowledge has on the size of the hypothesis space or the portion of the hypotheses space searched. It is hoped that this taxonomy can be used to illuminate the similarities and differences between the types of background knowledge used in a variety of systems including COBWEB (Fisher, 1988), CIGOL (Muggleton & Buntine, 1988) and OCCAM (Pazzani, 1988).

2 Background: FOIL

FOIL (Quinlan, in press) inductively generates constant-free Horn-clause theories in a manner similar to that used by ID3 (Quinlan, 1986) which generates decision trees with attribute-value tests. In particular, FOIL uses a divide-and-conquer approach guided by a heuristic based on information theory. In order to review Quinlan's approach we need to introduce some terminology. A Horn-clause definition for a concept $P_0(V_{0,1}, ..., V_{0,n_0})$ consists of a disjunctive set of clauses. Each *clause* consists of a head and a conjunction of *terms*:

$P_0(V_{0,1}, ..., V_{0,n_0}) : -P_1(V_{1,1}, ..., V_{1,n_1}), ..., P_m(V_{m,1}, ..., V_{m,n_m})$.

We will call $(V_{m,1}, ..., V_{m,n_m})$ a *variablization* of a predicate P_m. A variable $V_{m,i}$ of a term $P_m(V_{m,1}, ..., V_{m,n_m})$ will be called *old* if it appears either in the head or in

any unnegated term to the left of P_m. Otherwise, the variable will be called *new*.

Since FOIL is currently under development, we present here only its essential characteristics.[1] We assume that we are given a collection of labeled instances, a set of known predicates, and an unknown predicate P of known arity that we are to learn. The task is to determine a Horn-clause definition of P in terms of the given predicates as well as (in a limited manner) the predicate P.

Let Pos be the set of positive instances and let Neg be the set of negative instances. Actually the notion of an instance requires some elaboration. Similar to AQ (Michalski, 1980), FOIL has two main phases: adding clauses to the theory until every positive instance is covered and forming clauses that do not contain any negative instances. Positive instances are covered as follows:

```
Until Pos is empty
  Construct a clause that covers some positive instance and avoids all
   negative instances.
  Add clause to the theory.
  Remove those elements of Pos that are covered by the new clause.
```

Constructing a clause that misses all negative instances can also be described simply. Note that the clause $P_0(V_{0,1}, ..., V_{0,n_0}) \leftarrow true$ covers all positive instances. To avoid negative examples, this clause is specialized in the following manner:

```
Let P0(V0,1, ..., V0,n0) ← true be the initial clause
Let Old be (V0,1, ..., V0,n0).
Let Pos be the positive examples not satisfied by the current definition.
Let Neg be the negative examples.
Until Neg is empty
  Choose the predicate and variabilization with the maximum gain.
  Make the variablized predicate a term in the body of the clause.
  Add any new variables in the term to Old.
  Let Pos be all extensions of Pos that are satisfied by the term.
  Let Neg be all extensions of Neg that are satisfied by the term.
```

At this level of abstraction, FOIL is quite simple. It uses hill-climbing to add the term with the maximum information gain to a clause. For each variabilization of each predicate P, FOIL measures the information gain, which is defined differently from ID3. Without going into the exact computation, (see Quinlan, in press) we note that, in effect, the information metric checks every possible value of each variable in the term for whether the term is true. The old variables take on values

[1] All of the experiments we report were run on a rational reconstruction of the initial version of FOIL. In particular, later versions allow for inexact clauses, post-pruning, heuristic search control and various search optimizations.

from the training examples (or extensions to training examples). The new variables take on every possible value from the constants of the domain. When FOIL adds a term to a clause, it updates the set of old variables with the new variables of the term, and extends the set of positive and negative examples by adding all values for the new variables (provided the term is true with these values).

3.0 Analysis of FOIL

To begin, we estimate the cost of adding a single term to a clause. There are two reasonable measures we might use to estimate this cost. One measure, called the theory-cost, is independent of the number of examples and tells us how many different terms can be choosen to extend the body of the given clause. The second measure, called the evaluation-cost, depends on the particular examples that are given and measures the total number of examples that are checked to compute the term with the maximum information gain.

First, we give a worst-case analysis for the theory cost. For this analysis let Max be the maximum arity of any predicate, $Pred$ be the number of predicates, and Var be the number of variables in the current clause. To add a new predicate we may choose from one of $Pred$ predicates. If the predicate has arity Max (the worst case), then we must consider choosing Max variables from Var old variables and $Max - 1$ new variables (FOIL requires that at least one variable in a term be old). A simple upper bound on this is: $(Var + Max - 1)^{Max} - (Max - 1)^{Max}$. Consequently, the theory cost is:

$$TheoryCost = Pred * ((Var + Max - 1)^{Max} - (Max - 1)^{Max}) \quad (1)$$

A simpler approximation of this formula is:

$$TheoryCost = Pred * (Var + Max)^{Max} \quad (2)$$

One can make a number of qualitative inferences from this formula. In particular, it shows that additional predicates increase the cost by a linear amount, while increasing the maximum arity of the predicates increases the size of the search space exponentially.

In order to estimate the cost of evaluating a predicate, which is the main cost in running FOIL, we need to count the number of the examples being tested. At this stage in the algorithm, there are as many as Max variables which require instantiations. Let Const be the number of constants or values that variables could take on. Then, in the worst case, that could lead to $Const^{Max}$ examples. Consequently, the total cost of selecting the right predicate is bounded by:

$$EvaluationCost = TheoryCost * Const^{Max} \quad (3)$$

This shows that arity of predicates and the length of clauses, in terms of the number of distinct variables, and the number of constants are all going to play a strong role in limiting the size of theories that can be learned.

In the subsequent sections we will show how by adding knowledge, we can reduce, sometimes dramatically, these costs. Somewhat surprisingly, this analysis will also show that sometimes large amounts of knowledge will have very little effect on reducing the search space.

4.0 SaranWrap

SaranWrap extends FOIL in a variety of ways. Each of these extensions affect only how SaranWrap selects terms to test while extending a (possibly empty) clause under construction. One class of extensions allows SaranWrap to use semantic constraints to limit the search space. A second class of extensions allows SaranWrap to accept partial Horn-clause theories, containing both operational and non-operational predicates. These extensions allow SaranWrap to take advantage of incorrect and incomplete domain theories with surprising success.

In the following sections, we describe these extensions in more detail and evaluate the effect of each extension on the number of terms tested by SaranWrap. To illustrate these extensions we use two domains. The first domain, that of learning the *member* predicate, illustrates how a simple recursive concept can be learned. FOIL is provided with positive and negative examples of the *member* predicate (e.g., $member(b,[a,b,c])$ $notmember(a,[b,c])$ and the *component* predicate (e.g., $component(a,[b,c],[a,b,c])$ and learns a recursive definition for member:

$$member(X,Y) : -component(X,Z,Y).$$
$$member(X,Y) : -component(A,B,Y), member(X,B).$$

The second domain is much more complicated and was introduced by Muggleton *et al.* (1989). The concept description varies from four to eleven clauses, depending upon the predicates used. The predicate to be learned is $illegal(A,B,C,D,E,F)$ which is true if a chessboard containing a white king and rook and black king is in an illegal state. A state is illegal if either king is in check or more than one piece occupies the same space. A and B are the position of the white king (file and rank), C and D are the white rook's position, and E and F are the black king's position. In this example, the operational predicates used are $between(X,Y,Z)$ (the value of Y is between the values of X and Z), $adjacent(X,Y)$ (the value of X is either one greater or one less than the value of Y) and $equal(X,Y)$ (the values of X and Y are equal). This domain suggests that SaranWrap can handle realistically sized problems.

4.1 Semantic Constraints: Single arguments Semantic constraints may be placed on either the type of arguments or the combination of arguments. Type constraints provide a useful and inexpensive way of incorporating a simple form of background knowledge. SaranWrap can easily take advantage of typing information. Typing information reduces the search space in two manners.[2] First,

[2]Quinlan (in press) mentions how type constraints may be used (in combination with the

it is not necessary to test terms where the types of old variables conflict with the usage of these variables as arguments to a predicate. More precisely, let us assume that a domain has T types and in the best case these types are distributed equally among the variables. Then, with typing, theory-cost can reduce to $TheoryCost = Pred * (Var + Max)/T^{Max}$, an exponential reduction. Second, it is not necessary to test terms against examples where the type of the constant conflicts with the type of the variable. This reduces the evaluation cost to $TheoryCost * (Const/T)^{Max/T}$. This shows that in the best case typing can reduce the search space by an exponential amount. In practice, the reduction, though significant, is less the best case.

In the chess domain, typing information was used to ensure that the predicates *between*, *equal*, and *adjacent* were only applied to either all ranks or all files. The benefit of typing is illustrated by the fact that SaranWrap, using typing, tests 3240 terms and 242,982 examples as compared to 10,366 terms and 820,030 examples for SaranWrap without typing when learning *illegal* from 641 randomly selected positive and negative training examples.

4.2 Semantic Constraints: Multiple Arguments A second type of semantic constraint involves inter-argument constraints, the relationship between the arguments of a predicate. For example, $equal(X, X)$ is trivially true and $between(X, X, Y)$ is trivially false. Such expressions should not play a part in a concept definition and therefore it is wasteful to test hypotheses including these terms.

So far, we have only implemented the case in which it is necessary for all of the variables in a term to differ. Providing such inter-argument constraints on terms when SaranWrap learns *illegal* considerably reduces the size of the hypothesis space explored. Unlike typing, inter-argument constraints only reduce the number of variablized terms generated and do not reduce the number of examples on which a term is tested to compute the information gain.

The value of inter-argument constraints is illustrated by the fact that SaranWrap using typing and inter-argument constraints tests 1296 terms and 109,350 examples as compared to 3240 terms and 242,982 examples for SaranWrap using only typing.

4.3 Operational Domain theory In the next sections, we consider ways in which a domain theory can improve upon inductive learning. First we will consider the case where the domain theory is a (possibly incorrect) partial, operational Horn-clause definition of a concept, and then we will consider the non-operational case. The second case divides into two cases: one where we are given non-operational predicates and the other where we have a non-operational concept definition.

Incremental concept learning systems such as COBWEB (Fisher, 1987) build up a partial (operational) concept definition. This partial concept definition influences

closed-world assumption) to generate negative examples of the predicate to be learned from the positive examples. However, type constraints are not used to eliminate terms from consideration or to extend both positive and negative examples.

how further examples are processed. Langley (1989) has argued that a hypothesis maintained by an incremental learning system is analogous to the domain theory of an explanation-based learning system. The extension of SaranWrap to use a partial, operational Horn-clause theory is straightforward. SaranWrap processes each clause in the order they are given. Processing a clause consists of computing the positive and negative examples that satisfy the clause. If there are no positive examples, then the clause is deleted. If there are no negative examples, then the clause is accepted verbatim as part of the concept definition. If there are both positive and negative examples satisfied by the clause, SaranWrap specializes the clause by conjoining new terms in the same manner as FOIL extends a clause. The processing and possible extension of each clause insures that there is at least one positive example and no negative examples satisfied by each clause. At this stage, the partial definition may not correctly classify all positive examples. If there are positive examples not satisfied by the current Horn-clause theory, additional clauses are added in the same manner that FOIL creates new clauses.

The effect of providing a partial concept definition is quite surprising. In general, search in FOIL is dominated by the last term of the clause with the largest number of variables. This means that if a partial theory that is nearly complete, but omits the last conjunct of the clause with the largest number of distinct variables, reduces the search by only a negligible amount.

This is illustrated by the following experiment. We gave SaranWrap three partial definitions of the member function, namely:

1. $member(X, Y) : -component(X, Z, Y).$
2. $member(X, Y) : -component(X, Z, Y).$
   $member(X, Y) : -component(A, B, Y).$
3. $member(X, Y) : -component(X, Y, Z).$

The first two definitions are partial but correct. The last is incorrect. SaranWrap tests 268, 228 and 308 terms and 20140, 12167 and 23358 examples with the above partial concept definitions as compared to FOIL's 308 terms and 23,057 examples. Note that the correct partial definitions given do not significantly reduce the number of terms tested because the majority of the work is needed to add the last term to the last clause of member. The incorrect partial definition does not increase the number of terms tested but does increase the number of examples tested.

4.4 Non-Operational Predicates Partial theories involving operational predicates are not necessarily very valuable. Next, we consider domain theories using non-operational predicates, i.e., ones which are defined in terms of operational and other non-operational predicates. Systems such as CIGOL (Muggleton & Buntine, 1988) make use of (or invent) a background knowledge of this form. For example, if an operational definition of the predicate $between(X, Y, Z)$ is not provided, it could be defined in terms of the operational predicate $less\_than$ by:

$between(X, Y, Z) : -less\_than(X, Y), less\_than(Y, Z)$.

One advantage of the non-operational predicates is illustrated by the fact that $between(X, Y, Z)$ may have positive information gain, while $less\_than(X, Y)$ and $less\_than(Y, Z)$ may have negative gain. Therefore, FOIL's hill-climbing search may not learn a concept that involves $less\_than(X, Y), less\_than(Y, Z)$. Note that it would be computationally prohibitive to consider all conjunctions of length two of the operational predicates. In general, this would more than square the theory-cost and nearly square the evaluation-cost, both of which are usually large. Non-operational predicates provide information on what particular combinations of operational predicates may be useful and allow SaranWrap to simulate a selective look-ahead.

Non-operational predicates are evaluated in the same manner as operational predicates in SaranWrap. The information gain of all terms that variablize a non-operational predicate is computed. If the term with the most gain is non-operational, then the term is operationalized and the operational definition is added to the clause under construction.

The operationalization process in SaranWrap differs from that of EBL in that it is guided by an information gain metric over a set of both positive and negative examples rather than a single positive example. As in EBL, the operational definition for a predicate may specialize the predicate if the domain theory is disjunctive (i.e., if there are multiple clauses for any non-operational predicate). In EBL, the predicates that are the leaves of the proof tree of the single training example are used as the operational definition. In SaranWrap, the information gain metric is used to determine how to expand a proof tree, in the following manner:

```
operationalize(Predicate, Pos, Neg):
Initialize body to the empty set.
for each clause in the definition of Pred of Predicate
  compute_gain(clause, Pos, Neg)
for the clause with the maximum gain
  for each term T in the clause
    if T is operational add T to body
    else add operationalize(T, Pos, Neg) to body.
```

Due to its reliance on hill-climbing search, FOIL is unable to learn a completely correct definition of *illegal* using only *less_than*, *equal* and *adjacent*. When SaranWrap is also given a non-operational definition of *between* in terms of *less_than*, it finds a completely correct definition in terms of the operational predicates *less_than*, *equal* and *adjacent*.

A disadvantage of using non-operational predicates in this manner is that each additional non-operational predicate, particularly those with many arguments, increases the search space. This has the undesirable consequence that the more one

knows, the slower one learns. This became obvious when we added rules from a domain theory of chess to SaranWrap. These rules indicate facts such as a king is in check if there is an opposing rook in the same file as the king and there is not another piece between the rook and king. With this domain theory, SaranWrap tested 3063 terms and 283,602 examples to find an operational concept definition as opposed to 1296 terms and 109,350 examples when just the operational predicates are searched.

4.5 Non-Operational Concept Definitions In the previous section, we pointed out how adding background knowledge in the form of a domain theory can increase the ability of SaranWrap to find solutions. However, increasing the size of the domain theory may increase the search space explored by the learning program. In explanation-based learning, the search for a concept definition is facilitated by providing the learning system with a target concept (Mitchell, Keller, & Kedar-Cabelli, 1986). The target concept is assumed to be a correct, non-operational definition of the concept to be learned and the domain theory is assumed to be correct. In SaranWrap, we relax the assumptions that the target concept and the domain theory are correct. Because SaranWrap makes use of multiple training examples, it has the potential of learning a correct concept definition in spite of these inaccuracies.

When a non-operational target concept is provided to SaranWrap, it computes the information gain of the terms that are conjoined together to form the target concept. SaranWrap operationalizes the term in the target concept with the maximum gain (provided the gain is positive). Otherwise, the clause is extended inductively by incorporating the variablization of operational predicate with the maximum gain. In this manner, SaranWrap prefers to use an explanation-based method but can use an inductive method if the domain theory is incomplete or inaccurate (Pazzani, 1988)[3]. When SaranWrap is provided with a correct target concept and the domain theory of the previous section, it finds a correct definition of *illegal* by testing 72 terms and 9713 examples.

This extension will tolerate incomplete and incorrect theories because the terms used to extend a clause are tested by an information-based based metric to make sure they have positive gain. If there is no explanation-based extension with positive gain, then SaranWrap incorporates terms inductively. We have tested SaranWrap with an incorrect target concept (which leaves out the constraint that a chessboard is illegal if two pieces occupy the same square). In this case, SaranWrap tests 432 terms and 22,788 examples. We also tested SaranWrap with an incorrect domain theory that included an erroneous clause stating that a king could move like a knight (and therefore it is illegal for the opposing kings to be a knight's move apart). In

[3]We have also experimented with a version that compares the information gain of all operational terms to the information gain of the terms in the target concept. In all of the experiments reported in this paper, this alternative requires more search.

this case, SaranWrap tested 435 terms and 47,195 examples.

In the next section, we present experiments in which we systematically perturb the domain theory and indicate the effects on the search performed by SaranWrap.

5.0 Experiments

The following experiments demonstrate that SaranWrap, using a combination of explanation-based and empirical learning methods, can advantageously use partial domain theories, even when these theories contain severe errors. Incorrect domains theories for the definition of *illegal* are generated from a correct theory by four different perturbation operators, specifically randomly deleting or adding either a term or clause from/to the given theory. Note that deleting a term or adding a clause causes the theory to become overly general while adding a term or deleting a clause causes the theory to become overly specialized.

In the first set of experiments each perturbation operator was applied individually but possibly multiple times. Figure 1 plots the accuracy of the resulting domain theory (averaged over 10 trials on either the positive or the negative training examples as appropriate according to the type of modification) and the number of terms tested by SaranWrap for each operation as a function of the number of modifications to the domain theory. (There were fewer than 10 modifications possible for deleting clauses or terms.) Note that in every case SaranWrap learns a concept that is more than 99% accurate when tested on a set of 2,000 examples in spite of a domain theory that is quite inaccurate. In each case, the same 641 randomly selected training examples are used. Also note that SaranWrap is able to exploit extremely inaccurate domain theories to constrain the search for a concept definition.

Through a single mechanism, SaranWrap responds to each type of modified domain theory in a different manner. If these modifications result in negative gain, SaranWrap will not operationalize this rule but instead finds an accurate definition using as much of the domain theory as possible and fills in the remainder with its bottom-up inductive method.

5.1 Term Deletion: If the terms defining a clause have positive gain, then SaranWrap can operationalize the clauses that are not altered in a purely top-down fashion. If the clause has positive gain with the term deleted, then SaranWrap can operationalize this clause and then use bottom-up inductive methods to complete the definition. Quite often, using bottom-up methods, the operational definition contains the term that was deleted.

5.2 Clause Deletion: If the terms relying on a rule have positive gain, then SaranWrap can operationalize the clauses that are not deleted. Operational descriptions equivalent to the remaining clauses (i.e., they cover the positive training examples not covered by the remaining clauses and do not cover any negative examples) are added in a bottom-up fashion.

5.3 Term Addition: Often, the clauses unaffected will have greater positive gain

than the affected clause and will be operationalized first. At this point, the terms relying on a modified clause will not have positive gain, and an operational description equivalent to the clause before modification is added inductively. Occasionally, the clause has positive gain, and it is included in a concept definition anyway. To cover additional positive examples, SaranWrap inductively adds clauses. SaranWrap does not delete terms from operationalized descriptions and does not simplify Horn-clauses after learning. Therefore, when a clause with an additional term definition is operationalized, it is possible that an additional, more general term is learned inductively. This does not affect the accuracy of the resulting concept, but merely results in the induced theory having redundant clauses. A simple post-processor could be added to detect and simplify the definition (Quinlan, personal communication).

5.4 Clause Addition: SaranWrap operationalize clauses with maximum gain and it is unlikely that randomly added clauses will have more gain. In effect, SaranWrap uses information gain as a method to find a subset of the domain theory that is accurate. Cohen (1990) also presents a technique to deal with this problem by finding all possible explanations and using a greedy set covering technique to find a useful subset of the domain theory. In contrast, SaranWrap serially finds an operational specialization with the highest positive gain on all training examples and makes this the first clause. It removes those positive training examples covered by that clause, and finds the next specialized operational definition.

It is important to stress that SaranWrap does not contain any special code to deal with each type of modification. Rather, the above behavior falls out of using a uniform, information-based heuristic to judge the usefulness of operationalizing the target concept or using bottom-up methods to extend or add a clause. With these modifications, SaranWrap with a domain theory performed sufficiently better than SaranWrap without a domain theory even using domain theories that classified fewer than 50% of the training examples correctly.

We also ran experiments in which all of the above modifications were performed simultaneously on the domain theory, yielding an domain theory that misclassifies both positive and negative examples. Figure 2 plots the accuracy of the domain theory and the number of terms expanded by SaranWrap mode when the domain theory was modified by adding or deleting clauses and terms as a function of the number of modifications to the domain theory (averaged over 12 trails). As in the previous examples, the concept learned by SaranWrap is greater than 99% accurate when tested on 2000 test examples.

The results of adding and deleting clauses and terms indicate that SaranWrap with an incorrect and incomplete domain theory generally explores a smaller portion of the search space than SaranWrap without a domain theory until the domain theory falls below 70% accuracy.

6 Conclusions

In this paper, we have evaluated a number of types of knowledge with regard to aiding the inductive task of theory formation. We demonstrated that semantic constraints, whether on single arguments as in typing or on multiple arguments, are easy to add and yield great benefit. We also considered background knowledge in the form of partial theories. Somewhat surprisingly, we saw that an operational partial definition will reduce the search very little, while a non-operational partial definition and a supporting domain theory can reduce search by an exponential amount. A domain theory without a non-operational partial definition (i.e., a target concept) can increase the number of problems solved by hill-climbing and the cost of increasing the size of the search space for all problems. By classifying the various types of knowledge and knowing the value of each piece of knowledge, knowledge acquisition can be better directed. In addition, we demonstrated a means for uniformly combining theory and inductive learning into a method which can naturally and advantageously use and correct inconsistent and incomplete theories.

Acknowledgements

This research is partially supported by NSF grant IRI-8908260. We would like to thank Ross Quinlan for his advice on FOIL and Tim Cain, Caroline Ehrlich, Ross Quinlan, Wendy Sarrett and Glenn Silverstein for reviewing a draft of this paper.

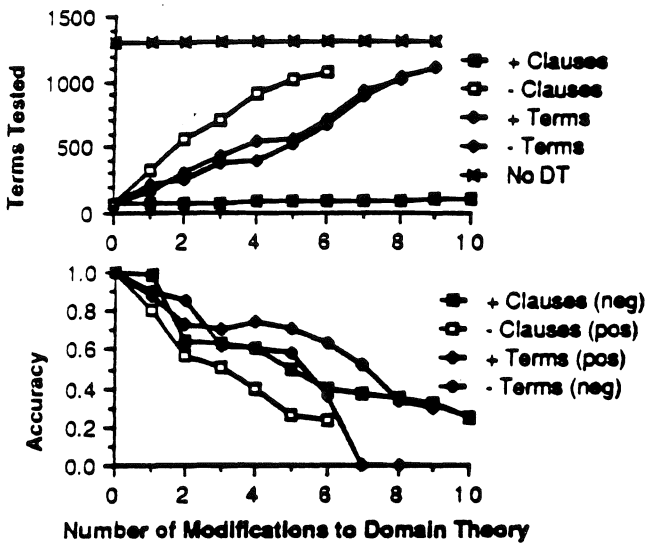

Figure 1. Individual modifications to the Domain Theory. + Terms indicates deleting terms from the domain theory, - Clauses indicates deleting clauses, etc. No DT indicates the domain theory was not used.

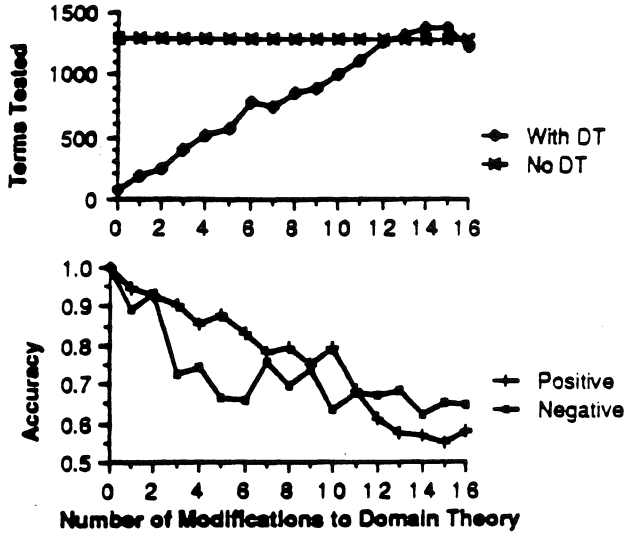

Figure 2. Combined Modification to the Domain Theory

References

Bergadano, F., Giordana, A., & Ponsero, S. (1989). Deduction in top-down inductive learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 23–25). Ithaca, NY: Morgan Kaufmann.

Cohen, William (1990). Abductive explanation-based learning: A solution to the multiple explanation-problem (ML-TR-29). New Brunswick, NJ: Rutgers University.

Danyluk, A. (1989). Finding new rules for incomplete theories: Explicit biases for induction with contextual information. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 34–36). Ithaca, NY: Morgan Kaufmann.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

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

Hirsh, H. (1989). Combining empirical and analytical learning with version spaces. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 29-33). Ithaca, NY: Morgan Kaufmann.

Katz, B. (1989). Integrating learning in a neural network. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 69–71). Ithaca, NY: Morgan Kaufmann.

Langley, P. (1989). Unifying themes in empirical and explanation-based learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 2–4). Ithaca, NY: Morgan Kaufmann.

Lebowitz, M. (1986). Integrated learning: Controlling explanation. *Cognitive Science, 10.*

Michalski, R. (1980). Pattern recognition as rule-guided inductive inference,*IEEE Transactions on Pattern Analysis and Machine Intelligence, 2*, 349–361.

Mitchell, T., Keller, R., & Kedar-Cabelli, S., (1986). Explanation-based learning: A unifying view. *Machine Learning, 1*, 47–80.

Mooney, R., & Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 5–7). Ithaca, NY: Morgan Kaufmann.

Muggleton, S., Bain, M., Hayes-Michie, J., & Michie, D. (1989). An experimental comparison of human and machine learning formalisms. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 115–118). Ithaca, NY: Morgan Kaufmann.

Muggleton, S. & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution *Proceedings of the Fifth International Workshop on Machine Learning* (pp. 339–352). Ann Arbor, MI: Morgan Kaufmann.

Pazzani, M. J. (1988). *Learning causal relationships: An integration of empirical and explanation-based learning methods.* Doctoral dissertation, University of California, Los Angeles.

Pazzani, M. (1989). Explanation-based learning with weak domain theories. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 72-74). Ithaca, NY: Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Quinlan, J. R. (1989). Learning relations: Comparison of a symbolic and a connectionist approach. Technical Report, University of Sidney, Syndey.

Quinlan, J. R. (in press) Learning logical definitions from relations.

Sarrett, W., & Pazzani, M. (1989). One-sided algorithms for integrating empirical and explanation-based learning *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 26–28). Ithaca, NY: Morgan Kaufman.

Shavlik, J. & Towell, G. (1989). Combining explanation-based learning and artificial neural networks. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 90–93). Ithaca, NY: Morgan Kaufmann.

3 1970 00802 9180