From Proceedings of the Eighth International Workshop on Machine Learning

The Generality Of Overgenerality

William W. Cohen
AI Principles Research Dept. AT&T Bell Laboratories
600 Mountain Avenue Murray Hill, NJ 07974
wcohen@research.att.com

Abstract

Theory specialization is the process of using examples to specialize an overgeneral domain theory. This paper shows by example that many different types of background knowledge can be encoded as overgeneral theories, and further, that these theories can be specialized using existing theory specialization techniques. A consequence of this is that theory specialization techniques are a surprisingly general way of making use of background knowledge.

1 INTRODUCTION

Several recent papers [Cohen, 1990b; Flann and Dietterich, 1989] have presented learning systems that use as background knowledge an initial domain theory that is "overgeneral": that is, a theory that defines a concept that is a *superset* of the concept to be learned. In this paper, we will use the term *theory specialization* to describe the problem of specializing an overgeneral initial theory using examples.

This paper argues that *many different types of background knowledge can expressed as overgeneral theories*, albeit overgeneral theories of a somewhat artificial nature. A consequence of this is that theory specialization techniques are surprisingly general: in particular, we will argue that *many different types of learning tasks involving background knowledge can be solved using existing theory specialization techniques.* This argument will be made by example; first, however, we will give a quick review of a particular theory specialization technique, which will be used in the examples.

2 A NEW VARIANT OF A-EBL

The ANA-EBL algorithm [Cohen, 1990b] solves the *theory specialization problem*, defined as follows:

Given: 1) an over-general theory T_0,
2) examples of correct/incorrect predictions made using T_0,
Find: a theory T_s that specializes T_0, and only leads to correct predictions.

The specialization T_s of the initial theory T_0 will be called the *target theory.* ANA-EBL forms a target theory in two steps. First, it generates a set of *candidate rules* that are partial operationalizations[1] of T_0. Second, it uses set covering techniques to find a set of candidate rules that covers all of the positive examples (correct predictions) and none of the negative examples (incorrect predictions.) This set of rules becomes the new specialized theory T_s, and the original theory T_0 is discarded. Thus ANA-EBL can be thought of as learning by "throwing away" the incorrect parts of an overgeneral theory; in this respect, ANA-EBL is similar to Flann and Dietterich's IOE technique [Flann and Dietterich, 1989]. Importantly, ANA-EBL does *not* assume that there is a single explanation for each example; rather, ANA-EBL allows one to have multiple inconsistent explanations for each example. The ANA-EBL algorithm is described in more detail in Figure 1.

In generating candidate rules, since there are an exponential number of partial operationalizations of even a single proof, it is necessary to constrain the class of partial operationalizations that are considered. In the examples, we will use slightly different constraint from that adopted by ANA-EBL. Conceptually, the new strategy for generating rules is to select k nodes of an explanation structure of some positive examples, and mark as operational *the most general of the nodes that are not ancestors of the selected nodes.* (An example of this will be given in Section 3.1.) This process is repeated for all possible selections of up to k nodes and for all explanations structures of all positive examples; as in ANA-EBL, a greedy set cover is then used

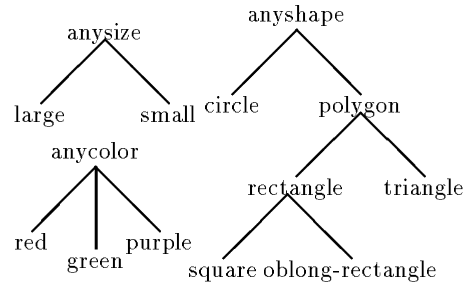
[1] In particular, rules formed by using EBG with a variant definition of operationality on the initial theory T_0 and the positive examples.

Algorithm ANA-EBL($S^+, S^-, T_0, \mathcal{O}, k$):

1. Compute the explanation structure of every proof in T_0 of every example in S^+.
2. For each explanation structure found in step 1, find the set of candidate rules that can be formed by
   (a) marking up to k internal nodes of the explanation structure as "operational"
   (b) applying the final stage of EBG to the resulting explanation structure.
3. Filter the set of candidate rules by removing any rule that covers an element of S^-.
4. Use a greedy set cover (for details, see [Cohen, 1990b]) to find a small set of the remaining candidate rules that covers all of the examples in S^+.
5. Return that set of rules as the theory T_s.

Figure 1: The ANA-EBL Learning Algorithm

Concept Hierarchies



Training Examples

+(small green square)
+(large red oblong-rectangle)
–(large red triangle)
–(small green triangle)
–(small purple square)
–(large purple oblong-rectangle)

Figure 2: The "christmas present" learning problem

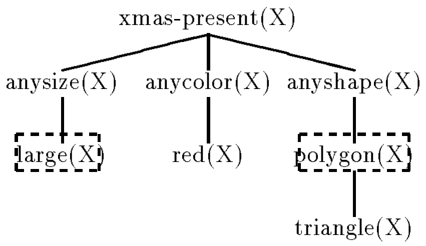


Figure 3: Sample explanation structure in the "christmas present" domain theory

to select a small subset of the candidate rules thus generated.[2] The new rule generation strategy shares with the old strategy the crucial property that at most $O(n^k)$ candidate rules are generated for each proof of size n; hence this variant of ANA-EBL still runs in time polynomial in the total size of the explanation structures for the positive examples.

To avoid possible confusion, we will refer to this variant algorithm as ELGIN[k-tip], or more briefly ELGIN.[3]

3 ENCODING LEARNING TASKS AS THEORY SPECIALIZATION

3.1 SIMPLE INDUCTIVE LEARNING

We will now describe how ELGIN can make use of various types of background knowledge. For pedagogical purposes, we will begin with the simple SBL problem of Figure 2. To encode this problem as a theory specialization task, we must first encode the background knowledge — the attribute hierarchies — as a theory. This can be done mechanically; the clauses below encode the heirarchies of Figure 2.

anysize(X) ← small(X).
anysize(X) ← large(X).
anycolor(X) ← red(X).
anycolor(X) ← green(X).
anycolor(X) ← purple(X).
anyshape(X) ← polygon(X).
anyshape(X) ← circle(X).
polygon(X) ← triangle(X).
polygon(X) ← rectangle(X).
rectangle(X) ← square(X).
rectangle(X) ← oblong_rectangle(X).

The encoding of the attribute hierarchies is most of what is needed to construct an appropriate overgeneral theory; to finish the job, it is sufficient to add to the theory the following overgeneral definition of the target concept *xmas_present*:

xmas_present(X) ← anysize(X), anycolor(X), anyshape(X)

Recall the candidate rule generation strategy is to select k nodes of an explanation structure and mark as operational the most general of the nodes that are not ancestors of the selected nodes. To illustrate this, suppose that $k = 2$, and the two framed nodes of the explanation structure of Figure 3 (for a "large red triangle") are selected. The nodes that are not ancestors of either of the selected nodes are the nodes labeled *large(X)*, *red(X)*, *anycolor(X)*, *polygon(X)*, and

[2] We have simplified for brievity; the actual implementation is somewhat more efficient that this description suggests. It repeatedly uses best-first search to generate the rule that would be chosen by the greedy set cover, rather than actually enumerating all the possible candidate rules.

[3] ELGIN is the name of a program that implements this and several other variations of the ANA-EBL algorithm; k-tips is the name of the new rule generation strategy [Cohen, 1990a].

drinking_vessel(X) ← stable(X),liftable(X),open_vessel(X).
stable(X) ← bottom(X,B),flat(B).
open_vessel(X) ← concavity(X,C),upward_pointing(C).
liftable(X) ← graspable(X),light(X).
graspable(X) ← handle(X,H).
graspable(X) ← small_width(X),insulating(X).

Figure 4: Theory for "Drinking Vessel"

triangle(X). Of these nodes, *triangle(X)* is less general than *polygon(X)*, and *red(X)* is less general than *anycolor(X)*; so the most general of the non-ancestor nodes are *large(X)*, *anycolor(X)*, and *polygon(X)*. These are marked as operational, leading to the rule
xmas_present(X) ← **large(X)**,anycolor(X),**polygon(X)**

In general, the candidate rules generated by ELGIN will have up to k meaningful conditions, and the remainder will be vacuous conditions like *anycolor(X)*. (For the remainder of the paper, we will write the non-vacuous conditions in boldface.) Since the final stage is to form a disjunction of these rules, for this domain theory, ELGIN[k-tip] operates much like Haussler's greedy algorithm for learning k-DNF [Haussler, 1988]. The actual implementation of ELGIN produces (in 1.2 seconds on a Sparc 1+) the following specialization from the examples of Figure 2.

xmas_present(A) ← anysize(A),**red(A),rectangle(A).**
xmas_present(A) ← anysize(A),**green(A),rectangle(A).**

With this domain theory, there is only one proof for each example, the size of the proof is bounded by the size of the attribute hierarchies, and the proof can be generated in linear time; it can be shown that these properties make generation of candidate rules by ELGIN computationally easy. Interestingly, the same technique can be used to learn relational concepts, rather than simply propositional ones [Cohen, 1990a, pages 127-136]. The greater difficulty of relational learning problems is reflected in the fact that there is no longer a single easily-generated proof for each example; instead there may be many proofs. This means that more candidate rules will be generated, and ELGIN will take more time to construct a hypothesis.

3.2 EMULATING THE IOU ALGORITHM

Now that we have seen how theory specialization can be used to perform simple inductive learning, it is fairly easy to show how theory specialization can be used to perform other tasks that involve learning from background knowledge. The first such task we will consider is *Induction Over the Unexplained (IOU)* [Mooney and Ourston, 1989]. IOU is also a theory specialization problem, but one to which ELGIN appears to be inapplicable: the difference is that the desired specialization of the theory is not in the deductive closure of the original theory, and hence cannot be constructed by ELGIN. However, to encode the problem of learning "cup" from "drinking vessel" (for example) as a ELGIN-style theory specialization problem, one can add to the theory above the following overgeneral definition of *cup*:

cup(X) ← drinking_vessel(X),
  anycolor(X),anyvolume(X),anyshape(X)

Negative examples that don't satisfy the definition of *drinking-vessel* will have no proofs, and hence are effectively discarded by ELGIN; again, other examples will have exactly one easily-generated proof. The theory that ELGIN actually finds (in 0.5 seconds) given the training data of [Mooney and Ourston, 1989] (shown in Table 1) is the same as the description generated by IOU:

cup(X)←**drinking_vessel(X)**,anycolor(X),
  small_volume(X),anyshape(X)

3.3 INCOMPLETE THEORIES

Another sort of background knowledge is an *incomplete theory*. Again, we will use the theory of Figure 4 and the examples of Table 1 to illustrate how theory specialization can be used to complete a theory.

As our first example, imagine that the rules defining *graspable* have been removed from the theory, and consider how this rule can be re-learned. One way to pose this as a theory specialization problem is to consider the attributes which might be relevant to graspability and write an overgeneral rule (in the style of the previous examples) for *graspable*. For example, we could add the rule

graspable(X) ← anyweight(X), anywidth(X), anyvolume(X),
  anyshape(X), maybeinsulating(X), maybehandle(X).

where *anyweight*, *anywidth* etc are the roots of the corresponding attribute hierarchies, and the remaining predicates are defined as (for example):

maybeinsulating(X) ← insulating(X).
maybeinsulating(X) ← not insulating(X).

Given the examples of Table 1 as training data (labeled as to whether or not they are "drinking vessels") we obtain (in 6.1 seconds) the following theory for *drinking vessel*:

drinking_vessel(A) ← **stable(A)**,anyweight(A),
  anywidth(A),anyvolume(A),anyshape(A),
  insulating(A), maybehandle(A),**light(A)**,
  open_vessel(A).
drinking_vessel(A) ← **stable(A)**,anyweight(A),
  anywidth(A),anyvolume(A),anyshape(A),
  maybeinsulating(A),**handle(A,B),light(A),**
  open_vessel(A).

From this theory we can extract the following definition of *graspable* (with conditions that are always true

Table 1: Examples for Learning "Cup"

vNAME(CLASS)	BOTTOM	FLAT	CONC	UP	LIGHT	HANDLE	WIDTH	INSUL	COLOR	VOLUME	SHAPE
CUP-1(+)	YES	YES	YES	YES	YES	YES	SMALL	NO	WHITE	SMALL	CYLINDER
CUP-2(+)	YES	YES	YES	YES	YES	NO	SMALL	YES	RED	SMALL	CYLINDER
SHOT-GLASS-1(-)	YES	YES	YES	YES	YES	NO	SMALL	YES	CLEAR	TINY	CYLINDER
MUG-1(-)	YES	YES	YES	YES	YES	YES	MED	NO	COPPER	LARGE	CYLINDER
CAN-1(-)	YES	YES	YES	YES	YES	NO	SMALL	NO	SILVER	SMALL	CYLINDER

discarded)

$$\text{graspable(A)} \leftarrow \text{insulating(A)} \vee \text{handle(A, B)}$$

This is a reasonable approximation, given the small amount of data. This example points out a weakness of ELGIN, when used for this sort of problem: the complexity of hypothesis (and hence the number of examples required to learn) depends on the size of the *operationalized* version of the completed theory, *not* on the size of the theory itself. The operationalized version of a theory can, of course, be much larger than the theory itself.

Theory completion is harder when the definitions of two or more predicates are missing. For example, suppose one added to the theory the clause *liftable(X)← sticky(X)*, along with an overgeneral definition of *sticky*. Now there are two possible proofs of each example X: one which assumes that *graspable(X)* is true, and one which assumes that *sticky(X)* is true. ELGIN can still be used, but as in learning relational concepts, the presence of multiple proofs means that more time will be spent: given the new theory, ELGIN requires 29.1 seconds to learn (as it turns out) a specialization identical to one shown above.

In the preceding examples, the missing rules were "at the bottom"; that is, the definition of *graspable* depends only on observed facts. Another interesting case is when the missing rule is "at the top". As an example of this, consider deleting the rule defining *drinking_vessel* from the theory. Again, one could replace this with an overgeneral rule such as the following.

drinking_vessel(X) ← maybestable(X), maybeliftable(X),
maybeopen_vessel(X), maybegraspable(X),
anybottom(X), anyconcavity(X), anyweight(X),
maybehandle(X), anywidth(X), maybeinsulating(X),
anycolor(X), anyvolume(X), anyshape(X).

This is a sort of constructive induction; we are trying to learn a concept based on intermediate-level concepts defined in the theory (like *liftable* and *graspable*) as well as low-level ones. In this case, ELGIN works much like the MIRO system [Drastal *et al.*, 1989]. Formal analysis suggests that the ELGIN algorithm is well-suited to this sort of problem since it is relatively insensitive to the number of intermediate-level attributes; in fact, the number of intermediate level attributes is reflected in only a logarithmic term of the worst-case bound on the sample complexity, and hence performance should degrade slowly as as the background theory grows. Unfortunately, the examples of Figure 1 do not provide enough information to re-learn a definition of *drinking-vessel*. Given these examples and the theory described above, ELGIN generates (in 5.0 seconds) the following rule:

drinking_vessel(X) ← maybestable(X), **liftable(X)**,
maybeopen_vessel(X), maybegraspable(X),
anybottom(X), anyconcavity(X), anyweight(X),
maybehandle(X), anywidth(X), maybeinsulating(X),
anycolor(X), anyvolume(X), anyshape(X).

By insisting that a drinking vessel be *liftable*, this separates the positive from the negative examples; however, since there are no negative examples indicating that the *stable* and *open_vessel* properties are necessary, they are not included in the learned concept.

safe_to_stack(X,Y) ← fragility(Y,low).
safe_to_stack(X,Y) ← lighter(X,Y).
lighter(X1,X2) ← weight(X1,W1),weight(X2,W2),W1<W2.
material(X,Y) ← madeof(X,Y).
construction(X,Y) ← body(X,Y).
material(X,M),construction(X,C) ≻ fragility(X,F)

+safe_to_stack(box2,box1). −safe_to_stack(box2,box3).
madeof(box1,lead). madeof(box2,steel). madeof(box3,lead).
weight(box1,100). weight(box2,10). weight(box3,10).
body(box1,rigid). body(box2,rigid). body(box3,nonrigid).

Figure 5: A determination-based theory and examples

3.4 DETERMINATIONS

The final sort of background knowledge that we will consider is *determinations*; the example that we will use to illustrate this is taken from [Mahadevan, 1989]; the theory and training examples are shown in Figure 5. Mahadevan used this as an example of a system which performed EBG using a theory composed of ordinary Horn clauses and determinations.

The crucial insight (for both Mahadevan's system and our own) is that each determination corresponds to a set of possible rules. For example, the determination of the theory of Figure 5 implies that some subset of the following set of rules are correct.

fragility(X,low) ← material(X,steel),construction(X,rigid).
fragility(X,high) ← material(X,steel),construction(X,rigid).
fragility(X,low) ← material(X,steel),construction(X,nonrigid).
fragility(X,high) ← material(X,steel),construction(X,nonrigid).
fragility(X,low) ← material(X,lead),construction(X,rigid).
fragility(X,high) ← material(X,lead),construction(X,rigid).

fragility(X,low) ← material(X,lead),construction(X,nonrigid).
fragility(X,high) ← material(X,lead),construction(X,nonrigid).

Adding all of these rules (rather than simply the correct ones) to the initial theory results in an overgeneral theory. Invoking ELGIN with the examples and theory of Figure 5, with the determination replaced manually[4] with the list of clauses above, produces (in 0.2 seconds) the following specialization.

safe_to_stack(A,B) ←
material(B,steel),construction(B,rigid).

This is the same result produced by Mahadevan's system. Unlike ELGIN, Mahadevan's system requires that only one determination be used in the proof of an example; however, Mahadevan's system produces its result from a single example.

Determinations are closely related to the "generalization rules" used for causal knowledge in [Pazzani, 1990]. We have also been able to translate some of Pazzani's examples, for example the one of [Pazzani, 1989], into over-general theories.

4 CONCLUSIONS

To summarize, we have argued (by example) that many sorts of background knowledge can be encoded as theory specialization problems, and furthermore, that existing theory specialization techniques can make effective use of these encodings. One purpose of this argument is to suggest a uniform way of discussing, comparing, and integrating various types of techniques for learning from background knowledge. The applications of theory specialization suggested by this paper raise several further possible topics for research. One interesting task is to develop appropriate *semantics* for overgeneral theories of the types presented here; the reader will note that these theories are an odd mix of statements with a clear meaning (such as the encoding of attribute hierarchies) with arcane programming tricks like

xmas_present(X) ← anysize(X),anycolor(X),anyshape(X)

Another topic is to expand the range of competence of existing theory specialization techniques. To pick an extremely specific example, one point raised by this research is that, while tree-structured and discrete attributes can be easily encoded in theory specialization problems, there is no efficient way of encoding real-valued attributes. A more general problem is to construct theory specialization algorithms more powerful than the k-DNF algorithm that ELGIN[k-tip] is based on. Some initial results on this problem are reported in [Cohen, 1991].

[4] A more elegant alternative to enumerating the above list of rules is to modify the theorem prover to supply these rules as needed to complete a proof; this also has the advantage that one does not need to know all of the possible materials, fragilities, and construction types in advance. Mahadevan's extension of EBG is roughly equivalent to applying EBG to explanation structures produced by such a modified theorem prover.

Acknowledgements

Discussions with Haym Hirsh helped to clarify some of the ideas in this paper.

References

(Cohen, 1990a) William W. Cohen. Explanation based generalization as an abstraction mechanism in concept learning. Technical Report DCS-TR-271, Rutgers University, 1990. (PhD thesis).

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

(Cohen, 1991) William W. Cohen. A decision tree approach to theory specialization. In preparation, 1991.

(Drastal *et al.*, 1989) George Drastal, Gabor Czako, and Stan Raatz. Induction in abstraction spaces: A form of constructive induction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989. Morgan Kaufmann.

(Flann and Dietterich, 1989) Nicholas Flann and Thomas Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4(2), 1989.

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

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

(Mooney and Ourston, 1989) Ray Mooney and Dirk Ourston. Induction over the unexplained. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York, 1989. Morgan Kaufmann.

(Pazzani, 1989) Mike Pazzani. Explanation-based learning with weak domain theories. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York, 1989. Morgan Kaufmann.

(Pazzani, 1990) Michael Pazzani. *Creating a Memory of Causal Relationships.* Lawrence Erlbaum, 1990.