From Proceedings of the Ninth International Conference on Machine Learning

Compiling Prior Knowledge Into an Explicit Bias

William W. Cohen
AT&T Bell Laboratories
AI Principles Research Department
Murray Hill, NJ 07974 USA
wcohen@research.att.com

Abstract

Current theory-guided learning systems are inflexible, in that they are committed to performing one particular class of theory corrections; this is problematic because in some cases special-purpose theory-guided learning systems can dramatically outperform general-purpose ones. To address this problem, we describe a new system in which theory-guided learning is separated into two phases. The first phase is "theory intrepretation", in which the initial theory is translated into an explicit description of the bias for an inductive learning system; we introduce antecedent description grammars as a language for explicitly representing this bias. The second phase is "grammatically biased learning", in which this bias is used to search for a hypothesis. We demonstrate empirically that this approach leads to a flexible learning system which can, by use of suitable translators, emulate several useful learning systems; we also argue that this architecture makes it easier for a user unfamiliar with the details of the learning system to predict the impact that an initial theory will have on learning.

1 Problems Facing Theory-Guided Learning

Using background knowledge to improve the performance of an inductive learner is a long-standing research problem. Recent progress on this issue has led to the development of several "theory-guided learning systems": systems that take as input an initial buggy domain theory in addition to examples of the concept to be learned, and output a corrected domain theory, sometimes in operational form. These systems have had some experimental success: for example, the EITHER system was able to significantly improve the performance of an ID3-type learner given an initial approximate theory for recognizing DNA promoter sequences [Ourston and Mooney, 1990]; and the FOCL system has made effective use of initial theories containing a wide variety of different kinds of errors on a chess endgame problem [Pazzani and Kibler, 1990]. Currently, improving and extending these learning systems is an active area of research [Towell *et al.*, 1990; Richards and Mooney, 1991; Wogulis, 1991].

The following learning problem illustrates some of the more difficult issues that remain for theory-guided learning.

The problem is to learn when white can attack black's king along a file in a chess position containing the two kings and one white rook. A chess position is represented as six numbers between one and eight representing the rank and file of the white king, white rook, and black king respectively: representative examples of the concept might be +*file-attack(1,5,3,6,5,6)*,−*file-attack(2,6,1,6,5,6)*. A correct theory for this problem is shown in Figure 1; we will consider giving various theory-guided learners a mutilated version of this theory in which the last clause is deleted.

file-attack(A,B,C,D,E,F) :- same-file(C,D,E,F) ∧ unblocked-king(A,B,C,D,E,F).
unblocked-king(A,B,C,D,E,F) :- ¬A=C.
unblocked-king(A,B,C,D,E,F) :- ¬between(D,B,F).
same-file(C,D,E,F) :- D=F. *% deleted*

Figure 1: Theory for file attacks in a KRK position

This problem was suggested by the results reported by Pazzani and Kibler [1990]. To summarize briefly, this paper described the theory-guided learning system FOCL and its application to the problem of learning when a KRK position is *illegal*; although arguments can also be made for other systems, in this paper I will take FOCL as representative of the state of the art in theory-guided learning. The *file-attack* problem is from the same domain as the *illegal* problem studied in [Pazzani and Kibler, 1990], uses the same feature predicates, and in fact is a subcase of *illegal*.[1] Further, FOCL was shown to improve performance given initial theories for *illegal* that contained exactly the type of mutilation (clause deletion) used in the *file-attack* problem. Thus one would expect that results for the *file-attack* problem would be similar to results on the *illegal* problem. Surprisingly, however, this is not the case. Table 1 shows that giving this initial theory to FOCL degrades rather than improves performance.[2]

Learner	%Error		Search	
	(no theory)	theory	(no theory)	theory
FOCL	(6.8)	6.8	(193.1)	197.1
SIMPLE	(6.8)	0.0	(193.1)	11.7
# examples			25 train/1000 test	

Table 1: Performance on *file-attack* problem

We note that FOCL did *not* fail in this case because the initial theory is intrinsically useless or difficult; in fact it is quite easy to construct a learning algorithm that performs well on this problem. To illustrate this, we wrote a new theory-guided learner called SIMPLE that uses a few very simple heuristics to complete an incomplete theory.[3] Our implementation of SIMPLE consists of only 21 lines of Prolog code; however, as Table 1 shows, SIMPLE is quite effective at using the initial theory for *file-attacks*.

This example illustrates two important points.

Generality comes at the expense of power. The reason for this apparent paradox is that SIMPLE gets a great deal of leverage from knowing that the *only* errors in the initial theory are missing predicates. Since the theory is assumed to otherwise be correct, it is fairly straightforward to take examples of *file-attack*,

[1] Pazzani and Kibler assume white to move.

[2] The increase in search space is small but statistically significant. In the table, the parenthesized numbers are for the knowledge-free (no initial theory) versions of the learners; and Search space measures the number of clauses tested against the set of examples; for problems in this paper this is a good comparative test of CPU time. All results are averages over 20 trials. *File-attack* problems are chosen from a non-uniform distribution in which file-attacks occur about two-thirds of the time; 25 training and 1000 test cases were used in each trial. Measures for FOCL are for our implementation, which does not include some later extensions to the algorithm such as predicate deletion [Pazzani *et al.*, 1991].

[3] In more detail, SIMPLE first scans the theory for predicates that are used, but not defined (such as *same-file* in this example) and tags these as "missing". Next, it examines each example in turn. If the example can be proved by the initial theory, it is discarded; otherwise SIMPLE backchains through the theory to see if there is some instance of a missing predicate that would imply that example. If there is, the example is replaced by the implying instance. Then, SIMPLE invokes an inductive learner (a knowledge-free version of FOCL) on the new example set to find a definition of the missing predicates, and adds this definition to the theory.

backchain through the incomplete theory, and derive examples of the missing predicate *same-file*. (To take a concrete example, SIMPLE can easily derive the fact that since *(1,5,3,6,5,6)* is a positive example of *file-attack*, *(3,6,5,6)* must be a positive example of *same-file.*) SIMPLE can then learn the definition of *same-file* by standard inductive logic programming techniques.

FOCL, on the other hand, makes no strong assumptions about the correctness of the initial theory; thus from FOCL's point of view deriving examples of "missing predicates" using SIMPLE's strategy is extremely chancy. (In the example above, for instance, while it might be that the domain theory should be repaired by adding some clause implying *same-file(3,6,5,6)*, it is equally likely that the *same-file* condition should be deleted from the first clause.) In general, FOCL must be much more careful in using the initial theory precisely *because* it is trying to guard against a broad class of errors. In short, *FOCL's failure on this narrow class of problems is an inevitable result of its success on the larger class.* Simply put, a generality-power tradeoff exists in theory-guided learning.

In retrospect, this tradeoff is not surprising; in fact, it would have been more surprising to find that no such tradeoff existed. However, a corollary to this observation is that the recent shift in research away from special-purpose theory-guided learners [Cohen, 1990; Mooney and Ourston, 1989; Flann and Dietterich, 1989; Whitehall, 1990] and toward general-purpose theory-guided learners like FOCL [Ourston and Mooney, 1990; Pazzani and Kibler, 1990; Towell *et al.*, 1990; Richards and Mooney, 1991; Wogulis, 1991] will probably *not* lead to a single learning system that will handle all theory-guided learning tasks. The end product of research in theory-guided learning will more likely be a *suite* of learning systems using different types of initial theories and at different points on the generality-power tradeoff curve.

It is difficult to predict the impact of adding knowledge to a system. This point is again emphasized by the *file-attack* problem, on which one would have expected FOCL to do well; a more natural illustration is given by the promoter recognition problem we consider later in the paper. The problem is that *the effect a theory will have is not solely a function of the theory itself, but also of the way in which the theory is treated by the learner*; again, this is illustrated by the *file-attack* problem, for which SIMPLE and FOCL get quite different results. Generally speaking, in *all* theory-guided learners that we know of, the learning algorithm used imposes some specific "interpretation" on the initial theory. (For example, SIMPLE interprets the initial theory as rules for reducing one class of examples to a simpler class; on the other hand, FOCL interprets the theory as a hint about what sort of clauses should be added to the hypothesis — roughly speaking, a hint that clauses in some sense "close" to clauses in the deductive closure of the initial theory are more likely to be correct.) Because the the impact of a theory depends on this "interpretation", and because the "interpretation" of a theory is determined by the learning algorithms, *the impact of the initial theory on a learning system is intricately and inseparably connected with the learning algorithms used in that system*, and thus difficult to predict unless the user understands the learning algorithms completely.

In practice, this is an extremely heavy burden to lay upon a user — an impossibly heavy one if a suite of learning programs is involved. It is hard to believe that theory-guided learning programs will ever be used (except by other machine learning researchers) unless one can either give a succinct and natural description of the initial theories that are useful to the learner, or, at the very least, provide some way of informing the user what the impact of his or her knowledge is going to be.

2 Explicitly Biased Learning

These problems can be addressed by adopting the alternative architecture shown in Figure 2. The learner itself accepts two inputs: a set of examples of the concept to be learned, and an explicit description of the hypothesis space to be searched — that is, a description of the *bias* of the learning system. A set of translators are used to derive a bias from an initial theory:[4] the user decides which of these translators should be used for his problem. For example, there might be one translator that converts an initial theory to the bias implicit in SIMPLE, and another that converts the initial theory to the bias implicit in FOCL. Notice that this architecture modularizes the theory-guided learning process in a new way: the process of theory interpretation has been neatly separated from learning. It is now the *translators* that impose an interpretation on the initial theory, not the learning algorithm itself.

We have implemented a learning system of this sort called GRENDEL,[5] which offers the following advantages over conventional approaches.

Flexibility. Since the ability to work with general classes of initial theories (as FOCL does) is not a panacea, it is important for a learning system to work well with many different types of initial theories. In this architecture, in order to investigate a new way of interpreting a background theory, it is only necessary to add an additional translator. This makes it easy to explore different special-purpose theory-guided learning systems, or new points on the generality-power

[4] Or more generally, from some source of background knowledge.

[5] Grammatically REstricted NonDeductive Learner

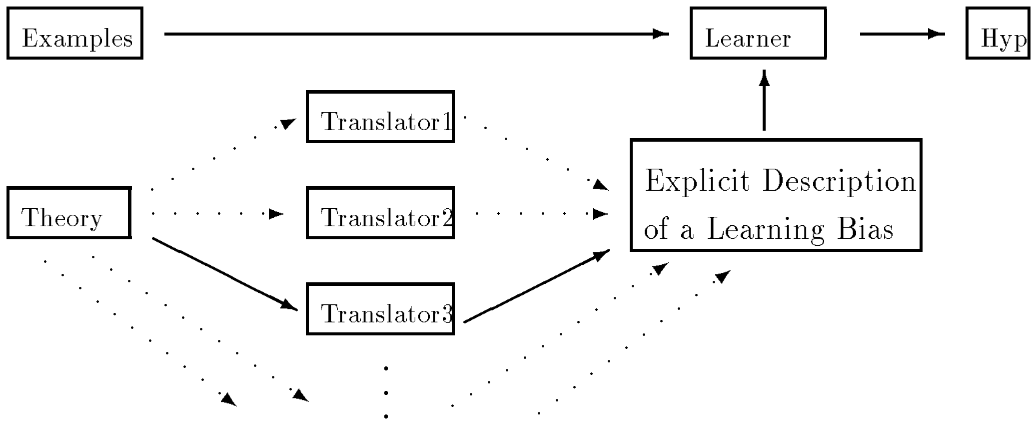

Figure 2: An explicitly biased learning system

tradeoff curve. One would expect a mature learning system of this type to include translators for several special-purpose learners like SIMPLE in addition to general-purpose ones like FOCL.

Predictability. If the output of the translators is intelligible to a user, then one can get immediate feedback on the likely effect of an initial theory. This gives some hope of providing a way for the user to predict the impact of adding an initial theory, and to determine if an interpretation of a theory is correct.

The challenge in designing such a learning system, of course, is in finding an appropriate means of representing the output of the translators: to coin a term, the difficulty is in finding an appropriate *bias representation language.* Ideally, this language will be perspicuous, expressive enough to encode a wide variety of types of knowledge, and easy to translate into, as well as being an efficient constraint language for the learner. In the remainder of the paper, we will first describe our bias representation language, and give several examples that indicate how it can be used to express some common biases. We will then give experimental results demonstrating the flexibility of the system, which is the more quantifiable of the benefits described above.

In this paper, we will *not* discuss the learning algorithm itself; indeed, one of the points of this paper is that to achieve "predictability" it must be possible to talk meaningfully about how initial theories are used in a learning system *without* detailed knowledge of the learning algorithms themselves. The learning algorithm is briefly summarized in an appendix, and discussed in detail in [Cohen, 1991b].

3 Antecedent Description Grammars

Learners like GOLEM [Bratko *et al.*, 1991], FOIL [Quinlan, 1990], and FOCL [Pazzani and Kibler, 1990] generate hypotheses of the form

$$\begin{array}{l} \mathrm{p}(X_1,\ldots,X_n) \text{ :-body}_1 \\ \mathrm{p}(X_1,\ldots,X_n) \text{ :-body}_2 \\ \vdots \\ \mathrm{p}(X_1,\ldots,X_n) \text{ :-body}_k \end{array}$$

where each body$_i$ is a conjunction of predicates from some predetermined set of "operational" predicates. We will assume that the learning problem is to generate a hypothesis of this form; let us consider how this space of hypotheses can be succinctly represented. As the consequents of the clauses in a hypothesis are always the same, the difficulty is clearly in describing the space of possible clause antecedents.

Since the antecedent of each clause is essentially a string of literals, a compact and general way of representing a set of possible antecedents is with a grammar.[6] Since each sentence generated by this grammar will be a possible antecedent of a clause, we call such a grammar an *antecedent description grammar (ADG)*; a learning system guided by such a grammar is called a *grammatically biased learner.* Several examples of ADG's will be given shortly.

Together, the declaration of the goal formula and the ADG define a "hard constraint" on the hypotheses that the learner can generate: any hypothesis must consist only of clauses such that the antecedent is a sentence of the grammar. A second type of bias is a *preference bias*: a bias that specifies an *order* in which the hypothesis space is to be searched. Currently, we use a fairly primitive mechanism for describing a preference bias. The productions of the grammar are marked as either *preferred* or *deferred.* The learner is constrained to generate antecedents using only the preferred portion of the grammar, unless the

[6] Specifically, an extended context-free language that generates a string of literals, rather than a string of symbols from a fixed alphabet.

data strongly suggest that this space contains no useful hypotheses; only in this case is the learner allowed to use the full grammar.

We now present some simple examples of ADG's.

Emulating FOIL. One simple ADG[7] is shown in Figure 3. The nonterminals of this example grammar have the following meanings. *pred2(X,Y)* expands to some binary predicate relating X and Y; *pred3(X,Y,Z)* is analogous for arity-three predicates. *rel(A,B,C,D,E,F)* expands to some "relationship" between some subset of the variables A, B, C, D, E and F — in particular, either a binary relationship between some pair of these variables, or an arity-three relationship between some triple. (The syntax "$A \rightarrow B$ where P" indicates a macro; it expands into a set of productions $A\theta_i \rightarrow B\theta_i$, one for each possible solution θ_i to the Prolog goal P.) *rels(A,B,C,D,E,F)* expands to a string (possibly empty) of "relationships" between the variables A, B, C, D, E, and F. The *body(G)* notation used in the first production is special; it is used to denote the start symbol of the grammar for the predicate G. Thus the first production says simply that the antecedent of a clause for *file-attack* must be some string of relationships between its six arguments.

This example shows that ADG's can capture a relatively weak learning bias, comparable to the bias used by the knowledge-free version of FOCL, which in turn is roughly the bias used by FOIL [Quinlan, 1990].[8]

Emulating EBL. The other end of the learning spectrum is if a complete and correct theory is given, for example the theory of Figure 1, which we will call Th_c in this discussion. In this case one would like a antecedent description grammar that generates only "correct" clauses — *i.e.*, clauses that are operationalizations of Th_c. Notice that the grammatical constraints on operationalizations can be derived from the clauses of Th_c: the first clause of Th_c, for example, says that an operationalization of *same-file* followed by an operationalization of *unblocked-king* is an operationalization of *file-attack.* This constraint is encoded in the second production of the grammar of Figure 4; other clauses of Th_c can be treated similarly. In general, in this grammar, each nonterminal *foo-op* expands to all possible operationalizations of the predicate *foo*. This ADG thus captures an extremely strong learning bias.

Next we will consider some more useful ADG's.

Emulating SIMPLE. As the next example, we will construct an ADG defining the bias of SIMPLE. Intuitively, we want to express the constraint that each clause must be an operationalization of some *completion* of the incomplete domain theory. One way to do this is to modify the grammar of Figure 4 as shown in Figure 5. The nonterminal *same-file-op* has been replaced with a new nonterminal *same-file-cmp*, which expands to all possible completions of *same-file*: in other words, it expands to an arbitrary string of relations. This ADG thus generates exactly the clauses consistent with some possible completion of the incomplete theory. The bias imposed by this ADG is close to SIMPLE's bias; a difference is that SIMPLE is *not* constrained to output an operational hypothesis.[9]

Emulating FOCL. After a detailed analysis of FOCL, we discovered that its bias can be succinctly described with the ADG of Figure 6: each rule generated by FOCL is either an arbitrary string of conditions, or an operationalization of the initial theory followed by a (possibly empty) string of additional conditions. FOCL also has a preference bias, because its SBL component is invoked only when the EBL component finds no useful rules; this is encoded by deferring the portion of the ADG that generates non-emptystring expansions of *rels*. The nonterminal *file-attack1-op* in the ADG generates all operationalizations of the initial mutilated theory.

Translation. As the preceding examples suggest, translation of initial theories into ADG's is quite trivial. A grammar like that of Figure 4 can be derived from the corresponding theory by performing a small number of syntactic operations on each clause of the theory — specifically, changing each ":-" to a "$\rightarrow$" and surrounding operational predicates with brackets. Similarly, definitions of nonterminals like *rels* can be easily generated given the desired arity and a list of operational predicates. The remaining examples are combinations of these two types of grammars. Currently translators have been implemented for the biases of SIMPLE, A-EBL, FOCL, and FOIL; *i.e.*, the translators take the same non-data inputs as these systems and convert them to ADGs.

---

[7] We use a notation similar to that used in Prolog for definite clause grammars: terminal symbols of the language are enclosed in square brackets (*e.g*, *[X=Y]*), commas denote concatenation, an empty pair of square brackets (*e.g.*, []) denotes the empty string, and capital letters like A and X denote logical variables. The semantics of the grammar are well-defined, and as one would expect; it differs from a standard context free grammar only in manipulating logical terms, rather than symbols from a fixed alphabet.

[8] The difference between this bias and FOIL's bias is that no new variables are generated in an antecedent; FOCL's iterative widening strategy has a similar effect in this domain.In the actual experiments we used a variant of this grammar that also enforces FOCL's typing and unique-variables constraints; see [Cohen, 1991b] for details.

[9] However, an ADG can be extended to generate non-operational clauses, for example by adding a production of the form "*foo-op* $\rightarrow$ *[foo]*" for each non-operational predicate *foo*. For the *file-attack* problem, this means adding the single rule "*unblocked-king-op(A,B,C,D,E,F)* $\rightarrow$ *[unblocked-king(A,B,C,D,E,F)]*". In the experiments below, we use ADG's that generate non-operational clauses in our emulation of SIMPLE.

body(file-attack(A,B,C,D,E,F)) → rels(A,B,C,D,E,F).

rels(A,B,C,D,E,F) → [].
rels(A,B,C,D,E,F) → rel(A,B,C,D,E,F),rels(A,B,C,D,E,F).

rel(A,B,C,D,E,F) → pred3(X,Y,Z) where subset([X,Y,Z],[A,B,C,D,E,F]).
rel(A,B,C,D,E,F) → pred2(X,Y) where subset([X,Y],[A,B,C,D,E,F]).

pred2(X,Y) → [X=Y]. pred3(X,Y,Z) → [between(X,Y,Z)].
pred2(X,Y) → [¬X=Y]. pred3(X,Y,Z) → [¬between(X,Y,Z)].
⋮

Figure 3: Antecedent description grammar for FOIL

body(file-attack(A,B,C,D,E,F)) → file-attack-op(A,B,C,D,E,F)

file-attack-op(A,B,C,D,E,F) → same-file-op(C,D,E,F),unblocked-king-op(A,B,C,D,E,F).
unblocked-king-op(A,B,C,D,E,F) → [¬A=C].
unblocked-king-op(A,B,C,D,E,F) → [¬between(D,B,F)].
same-file-op(C,D,E,F) → [D=F].

Figure 4: Antecedent description grammar for EBL

body(file-attack(A,B,C,D,E,F)) → file-attack-cmp(A,B,C,D,E,F)

file-attack-cmp(A,B,C,D,E,F) → same-file-cmp(C,D,E,F),unblocked-king-op(A,B,C,D,E,F).
unblocked-king-op(A,B,C,D,E,F) → *as defined in Figure 4*
same-file-cmp(C,D,E,F) → rels(C,D,E,F).
rels(C,D,E,F) → *defined analogously to rels(A,B,C,D,E,F) in Figure 3*

Figure 5: Antecedent description grammar for SIMPLE

body(file-attack(A,B,C,D,E,F)) → file-attack1-op(A,B,C,D,E,F),rels(A,B,C,D,E,F)
body(file-attack(A,B,C,D,E,F)) → rels(A,B,C,D,E,F)
rels(A,B,C,D,E,F) → *as in Figure 3, but with nonemptystring expansions "deferred"*

file-attack1-op(A,B,C,D,E,F) → *some operationalization of the mutilated theory*

Figure 6: Antecedent description grammar for FOCL

4 Experimental Results

To verify that these sorts of ADG's do in fact capture their intended biases, and also to verify that GRENDEL does in fact search the hypothesis spaces defined by ADG's effectively, a series of experiments were run.

Testing the SIMPLE bias. To test the SIMPLE bias, SIMPLE-emulating ADG's were generated for two theory completion problems: *file-attack*, and also *bad-call*, the most difficult[10] of several problems described in a recent thesis on theory completion [Whitehall, 1990]. As shown in Table 2, GRENDEL with this bias (denoted $\text{GRENDEL}_{\text{SIMPLE}}$ in the table) is similar to SIMPLE; differences are due to the fact that GRENDEL uses a different strategy for searching the space than SIMPLE.

Testing the FOCL bias. To test the FOCL bias, FOCL-emulating ADG's were generated for the mutilated *illegal* theory of [Pazzani and Kibler, 1990] and the promoter sequence theory used by Ourston and Mooney [1990] in demonstrating their EITHER system; the results are shown in Table 3. As can be seen, GRENDEL's emulation (denoted $\text{GRENDEL}_{\text{FOCL}}$) closely tracks the performance of FOCL on both problems; the differences again reflect GRENDEL's slightly different search strategy. However, on *promoter* the initial theory does not statistically significantly improve performance.

The reason for FOCL's failure on *promoter* is that this initial theory is extremely over-specific. Since the theory is over-specific, any operationalization of it is also over-specific, and thus the part of the hypothesis space generated by the grammar rule "*body(promoter(X))*

[10] In this experiment, a correct theory for *bad-call* was mutilated by deleting one predicate (*susp-len.*) For comparison Whitehall's system attains about 8% accuracy from 125 examples given this initial theory; however, his system does not use negative examples.

	file-attack				*bad-call*			
Learner	%Error		Search		%Error		Search	
FOCL	(6.8)	6.8	(193.1)	197.1	(31.6)	31.6	(5328.8)	5371.5
SIMPLE	(6.8)	0.0	(193.1)	11.7	(31.6)	3.1	(5328.8)	283.4
$\text{GRENDEL}_{\text{SIMPLE}}$	(6.8)	0.0	(199.3)	14.7	(31.7)	3.5	(5429.3)	379.1
# examples	25 train/1000 test				125 train/2500 test			

Table 2: Results using the SIMPLE bias

$\rightarrow$ *promoter1-op(X),rels(X)*" contains no useful rules. However, this is the also only part of the hypothesis space that makes any use of the initial theory. The simplicity of this explanation gives some anecdotal evidence that ADGs give useful feedback about the appropriateness of the interpretation of a theory.

The explanation also suggests a way of fixing the problem: rather than considering only *specializations* formed by adding conditions to operationalizations of the initial theory, it might be useful to consider *generalizations* formed by deleting conditions.[11] An easy way to extend the FOCL bias to allow such deletions is to replace each occurance of a terminal symbol *[foo]* in the grammar defining *promoters1-op* with a new nonterminal symbol *opt-foo* representing an optional occurance of *foo*; *opt-foo* can be defined with the pair of rules "*opt-foo* $\rightarrow$ *[foo]*" and "*opt-foo* $\rightarrow$ *[]*."

Extending the FOCL bias in this way does not uniformly improve performance; in fact, it degrades accuracy on the *illegal* problem. However, the extension does allow GRENDEL to perform substantially better on the promoter problem than any previously described symbolic learning algorithm;[12] GRENDEL also becomes the first theory-guided learner to obtain good results on both the illegal problem and the promoter problem, two commonly used benchmarks. Notice that being able to flexibly adjust its bias (from $\text{GRENDEL}_{\text{FOCL}}$ to $\text{GRENDEL}_{\text{FOCL+}}$) was necessary to obtain these results.

5 Conclusions

To summarize, we have described a new approach to theory-guided learning, which is based on translating an initial theory into an explicit representation of the bias of an inductive learner; to explicitly represent such a bias, we introduced the notion of an *antecedent description grammar*. In our approach, theory-guided learning is separated into two phases: an explicit *theory interpretation* phase in which the initial theory is translated into a bias, and a *learning* phase in which this bias is used to search for a hypothesis. We have argued that this approach leads to a learning system that is more "predictable" than existing theory-guided learners, by which we mean that it is easier to predict the impact that knowledge will have on a learning system. We have also shown via experiments that this approach leads to a more flexible learning system, in the sense that many different interpretations of an initial theory can be adopted: in addition to faithfully emulating the FOCL system of Pazzani and Kibler [1990], our learner can adopt biases leading to order-of-magnitude improvements in accuracy and learning time on problems for which FOCL is unsuited.

The theory-guided learning problem addressed by GRENDEL has been extensively studied: some examples are [Ourston and Mooney, 1990; Pazzani and Kibler, 1990; Towell *et al.*, 1990; Richards and Mooney, 1991; Wogulis, 1991]. GRENDEL differs from such systems primarily in its emphasis on flexibility and modularity. Historically, this work grew out of our efforts on analyzing the task of learning from an overgeneral theory [Cohen, 1991a]; ADG's are in fact closely related to such theories. GRENDEL improves on the learning system described in [Cohen, 1991a] in that it uses the more comprehensible notation of ADG's for its bias, includes mechanisms for expressing a preference bias, and (although this was not discussed in this paper) uses a more powerful learning algorithm. Another influence on this work was the LEX system [Mitchell *et al.*, 1983], in which a version space approach was used to construct hypotheses that are sentential forms in a given grammar. However, ADG's are computationally more powerful than the grammars used in LEX, and need not have a single parse for each example; these technical extensions are crucial in being able to use grammars to describe the biases of theory-guided learning systems.

Acknowledgements

I would like to thank Haym Hirsh and Mike Pazzani for comments on a draft of this paper.

[11] An extension to FOCL that allows such deletions is described in [Pazzani *et al.*, 1991]; however, such deletions are only considered if the original clause has positive gain, which is not the case on this example. Thus considering a version of FOCL with this extension would not affect our results.

[12] Again shown in Table 3; this version of GRENDEL is denoted $\text{GRENDEL}_{\text{FOCL+}}$. Figures for EITHER and AUDREY are from [Wogulis, 1991]. We note that GRENDEL still lags behind non-symbolic learning algorithms like k-NN, BACKPROP, or KBANN [Towell *et al.*, 1990].

	illegal				*promoters*			
Learner	%Error		Search		%Error		Search	
EITHER	—	—	—	—	(28.0)	21.0	—	—
AUDREY	—	—	—	—	(49.1)	22.6	—	—
FOCL	(3.3)	2.1	(570.0)	247.1	(21.3)	20.8	(3642.0)	3764.6
GRENDEL$_{FOCL}$	(3.3)	2.2	(586.0)	290.2	(21.3)	20.8	(3658.0)	3815.6
GRENDEL$_{FOCL+}$	(3.3)	3.9	(586.0)	278.0	(21.3)	14.2	(3658.0)	308.7
# examples	100 train/5000 test				80 train/26 test			

Table 3: Results using the FOCL bias

A GRENDEL's Learning Algorithm

GRENDEL's learning algorithm is based on Quinlan's FOIL system [Quinlan, 1990]. In its outer loop, FOIL repeatedly constructs a clause that covers no negative examples and (hopefully) a large number of positive examples, and adds this clause to the hypothesis. In its inner loop, FOIL begins with the general clause *foo:-true* and uses a greedy procedure to specialize it until it is consistent with the negative data, repeatedly refining the clause by appending an additional condition to its antecedent. An information gain metric is used to guide this search.

GRENDEL ascribes the following semantics to sentential forms of the ADG: a sentence is defined to cover the examples provable by the clause to which it corresponds, and a sentential form is defined to cover the union of the examples covered by all of the sentences derivable from it. (Various representational tricks are used to quickly test if an example is covered by a sentential form.) Notice that this semantics imposes a partial order on the space of sentential forms, with the start symbol the most general sentential form and sentences the most specific.

GRENDEL then modifies the inner loop of FOIL as follows: it begins with the start symbol of the grammar and repeatedly specializes it (using FOIL's information-gain metric) by applying rewrite rules from the ADG until it is both consistent and contains no nonterminal symbols; this latter condition is actually under the control of the user, so sentential forms could be generated as well. In more detail, at each stage in the greedy search, GRENDEL considers replacing its current clause with the set of all clauses derivable by a sequence of rewrite rules from the ADG such that (a) no single rule is used more than once in the sequence and (b) each grammar rule in the sequence rewrites a symbol that was introduced by the immediately preceding rule. This has empirically been found to give a wide enough selection so that greedy search is both effective and not impractically slow. GRENDEL only uses grammar rules from the preferred part of the ADG, unless none of the sentential forms produced in this way have positive gain; in that case all grammar rules are considered.

GRENDEL's learning algorithm is described in full detail, with examples and additional motivation, in [Cohen, 1991b].

References

(Bratko *et al.*, 1991) Ivan Bratko, Stephen Muggleton, and Alen Varsek. Learning qualitative models of dynamic systems. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Cohen, 1990) 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, 1991a) William W. Cohen. The generality of overgenerality. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Cohen, 1991b) William W. Cohen. Grammatically biased learning: learning Horn theories using an explicit antecedent description language. AT&T Bell Labs Technical Memorandum; available from the author on request., 1991.

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

(Mitchell *et al.*, 1983) Tom Mitchell, Paul Utgoff, and Ranan Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, 1983.

(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.

(Ourston and Mooney, 1990) Dirk Ourston and Raymond Mooney. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, Massachusetts, 1990. MIT Press.

(Pazzani and Kibler, 1990) Michael Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. Technical Report 90-18, University of California/Irvine, 1990.

(Pazzani *et al.*, 1991) Michael Pazzani, Clifford Brunk, and Glenn Silverstein. A knowledge-intensive approach to learning relational concepts. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Quinlan, 1990) J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.

(Richards and Mooney, 1991) Bradley Richards and Raymond Mooney. First-order theory revision. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.

(Towell *et al.*, 1990) Geoffrey Towell, Jude Shavlik, and Michiel Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, Massachusetts, 1990. MIT Press.

(Whitehall, 1990) Bradley L. Whitehall. Knowledge-based learning: Integration of deductive and inductive learning for knowledge base completion. Technical Report UILO-ENG-90-1776, University of Illinois, 1990.

(Wogulis, 1991) James Wogulis. Revising relational domain theories. In *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York, 1991. Morgan Kaufmann.