ELSEVIER

Artificial Intelligence 68 (1994) 303–366

Artificial Intelligence

Grammatically biased learning: learning logic programs using an explicit antecedent description language

William W. Cohen *

AT&T Bell Laboratories, Room 2A-427, 600 Mountain Avenue, Murray Hill, NJ 07974, USA

Received September 1991; revised April 1993

Abstract

Every concept learning system produces hypotheses that are written in some sort of constrained language called the concept description language, and for most learning systems, the concept description language is fixed. This paper describes a learning system that makes a large part of the concept description language an explicit input, and discusses some of the possible applications of providing this additional input. In particular, we discuss a technique for learning a logic program such that the antecedent of each clause in the program can be generated by a special antecedent description language; it is shown that this technique can be used to make use of many different types of background knowledge, including constraints on how predicates can be used, programming clichés, overgeneral theories, incomplete theories, and theories syntactically close to the target theory. The approach thus unifies many of the problems previously studied in the field of knowledge-based learning.

1. Introduction

Every concept learning system produces hypotheses that are written in some sort of constrained language called the *concept description language*. Most learning algorithms are specialized to work for a narrow class of concept description languages, and hence, for most learning systems, the concept description language is largely fixed; typically it is possible to modify the concept description language only by adding or removing features, and (possibly) by specifying

* E-mail: wcohen@research.att.com.

SSDI 0004-3702(93)E0048-Q

a generalization hierarchy for the features. The ability to modify the concept description language except in very limited ways, however, is rare [21].

This paper describes a learning system that makes a larger part of the concept description language an explicit input to the learner, and discusses some of the possible applications of providing this additional input. In particular, we discuss a technique for learning a set of Horn clauses such that the antecedent of each clause can be generated by a special *antecedent description language*; an antecedent description language is an extended context-free language which generates a string of literals, rather than a string of symbols from a fixed alphabet.

The problem we hope to address with this technique is the problem of using *background knowledge* of the target concept to improve concept learning performance; here improving performance means either reducing the time required for learning, or improving the quality of the learned knowledge. Previous research has suggested a variety of techniques for taking advantage of special types of background knowledge, some examples of which are described below.

- *Constraints on how predicates can be used.* For example, if $less\_than(X, Y)$ is one of the predicates available for inclusion in the hypothesis, then knowing that $less\_than(X, Y)$ always fails might be useful background knowledge. Similarly, knowing that $equal(X, Y)$ is logically equivalent to $equal(Y, X)$, or knowing that the two arguments to the *equal* predicate must be of compatible types, might also be useful. Pazzani and Kibler [26] describe a way of extending Quinlan's FOIL system [28] to obey such constraints.
- *Knowledge of "programming clichés".* The knowledge that various programming constructs are common, and hence likely to appear in the target theory, can be useful. An example of such programming clichés in logic programming are conjunctions of the form

  $$p(X_1, \ldots, X_i, \ldots, X_k) \wedge greater\_than(X_i, n)$$

  where X_i is a new (previously unbound) variable and n is a numeric constant. For technical reasons, such clichés can be useful in learning relational concepts; [34] describes a second extension to FOIL that takes advantage of programming clichés.
- *Theories of related concepts.* In some circumstances it is helpful to have a theory defining a concept that is related in some specific way to the target theory. Some techniques that use such related-concept theories are IOE [15], IOU [22] and A-EBL [6].
- *Incomplete theories of the target concept.* It may be the case that some but not all of the clauses in the target theory are known. For example, it might be known that the target theory includes the clause

  $$high\_risk\_of\_heart\_attack(X) :-$$
  $$parent(X, Y) \wedge died\_of\_heart\_disease(Y) \wedge high\_blood\_pressure(X)$$

but the definition of *high_blood_pressure*(X) may be unknown. Several researchers have attacked this problem using a variety of methods; see for instance [13,17,37,40].

- *Theories syntactically close to the target theory.* In some cases, a theory may be available that could be transformed into the target theory by a small number of local syntactic changes, such as adding a condition to the antecedent of a clause, deleting a condition from a clause, adding a clause, or deleting a clause. Some examples of learning systems that address this problem are [16,23,26,25,36].

The contribution of this paper is to describe a *single* technique which can make use of *all* of these types of background knowledge—as well as other types of information about the target concept—in a uniform way. (This is in contrast to systems like FOCL [25] which integrate several *different* techniques for using background knowledge in the same learning system.) All background knowledge is represented in a single formal structure—the antecedent description grammar—that has a clear declarative meaning. In other words, the various types of background knowledge listed above are used by first transforming them into an appropriate antecedent description grammar, and then using a learner that is biased by this grammar. One can view the antecedent description grammar as a sort of "common coin" into which other sorts of background knowledge can be translated; this translation is possible because we allow very general classes of grammars to bias learning. In contrast, previous techniques for allowing an explicit grammatical bias to influence learning allowed only restricted classes of grammars [21].

The technique we describe is applied to the specific problem of learning a set of Horn clauses. In this problem, it is assumed that the input is a series of labeled positive and negative examples of the form

$$\pm p(t_{1,1},\ldots,t_{1,n_1}), \pm p(t_{2,1},\ldots,t_{2,n_2}),\ldots$$

where each $t_{i,j}$ is a ground term, and it is assumed that the *target concept* (i.e., the concept to be learned) can be expressed as a set of Horn clauses of the form

$$\begin{aligned} &p(X_1,\ldots,X_n) \text{ :- } body_1\\ &p(X_1,\ldots,X_n) \text{ :- } body_2\\ &\vdots\\ &p(X_1,\ldots,X_n) \text{ :- } body_k \end{aligned}$$

where each $body_i$ is a conjunction of literals. The literal $p(X_1,\ldots,X_n)$ is called the *goal formula.* Negative literals in the $body_i$'s are allowed, but are interpreted using the Prolog rule of negation as failure. This learning problem is a generalization to relational concepts of the learning problem addressed by attribute–value learning systems that learn concepts in disjunctive normal form [18,24].

This learning problem has received increasing attention recently, in part because of the apparent success of learning systems like GOLEM [3] and FOIL [28]. FOIL's main restriction is that it learns only clauses with no function symbols; however, experiments indicate that FOIL solves this restricted problem quite well. Because we are considering FOIL-like learning problems, and because the learning technique described in this paper is an extension of the algorithm used in Quinlan's FOIL system, this paper could be viewed as presenting a technique for adding an explicit grammatical bias to FOIL.

As we will show later, there is a close connection between grammatically biased learning and *theory specialization* as performed by the A-EBL system (see [9], and also Section 5.2 of this paper). Yet another way to view this paper is applying FOIL-like learning techniques to the A-EBL theory specialization problem. However, we prefer to introduce the new metaphor of grammatically biased learning to describe the extended system, as these more powerful learning techniques allow us to attack a broader range of problems, including many for which the metaphor of theory specialization is clearly inappropriate.

In the remainder of the paper, we first define the antecedent description grammars that are used to bias learning, and then motivate and describe our semantic interpretation of these grammars. Section 3 then describes a learning algorithm, based on Quinlan's FOIL algorithm, which is guided by an antecedent description grammar. Sections 4 and 5 describe experimental results dealing with our algorithm. Finally, we conclude by discussing related work, further research issues, and conclusions that can be drawn from this research.

The experimental portion of this paper (Sections 4 and 5) is rather long. This is a consequence of the nature of our claim, which is a claim about the generality of our method. To substantiate this claim requires experimental comparison with a large number of systems; each of these comparisons requires some discussion and explanation.

2. Antecedent description grammars

2.1. Notation for antecedent description grammars

As a running example, we will use a problem discussed in various places in the literature (see [28,26], among many others): the problem of learning when a chess position containing two kings and one rook is illegal. Formalized, this becomes the problem of learning the predicate $illegal(A, B, C, D, E, F)$, where A and B are the rank and file of the white king, C and D are the rank and file of the white rook, and E and F are the rank and file of the black king. A position is *illegal* if the two kings are adjacent, if two pieces occupy the same square, or if the black king is in check (since we assume white-to-move). The predicates available for inclusion in the hypothesis are $adj(X, Y)$, which is true if the ranks or files X and Y are adjacent, $less\_than(X, Y)$, which is

goal_formula(*illegal*(*A*, *B*, *C*, *D*, *E*, *F*)).

1) *body*(*illegal*(*A*, *B*, *C*, *D*, *E*, *F*)) → *rels*(*A*, *B*, *C*, *D*, *E*, *F*).

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

4) *rel*(*A*, *B*, *C*, *D*, *E*, *F*) → *pred*(*X*, *Y*)
where *member*(*X*, [*A*, *B*, *C*, *D*, *E*, *F*]), *member*(*Y*, [*A*, *B*, *C*, *D*, *E*, *F*]).

5) *pred*(*X*, *Y*) → [*X* = *Y*].
6) *pred*(*X*, *Y*) → [¬*X* = *Y*].
7) *pred*(*X*, *Y*) → [*adj*(*X*, *Y*)].
8) *pred*(*X*, *Y*) → [¬*adj*(*X*, *Y*)].
9) *pred*(*X*, *Y*) → [*less_than*(*X*, *Y*)].
10) *pred*(*X*, *Y*) → [¬*less_than*(*X*, *Y*)].

Fig. 1. An antecedent description grammar for *illegal*.

true if rank or file X precedes Y, and *equal*(X, Y). Because these predicates play the same role as features in nonrelational inductive learning, we will call them *feature predicates*.[1] In the remainder of the paper we will use the infix operator "=" for the predicate *equal*.

A grammar defining one possible antecedent description language for this learning problem is shown in Fig. 1. An *antecedent description grammar* differs from a standard context-free grammar in that its symbols are logical literals; to emphasize the fact that the symbols of the language are logical literals rather than letters of a finite alphabet, we will henceforth use the terms ℓ-symbol, ℓ-terminal, and ℓ-nonterminal to denote symbols, terminals, and nonterminals respectively. In this paper, antecedent description grammars will be shown in a notation similar to that used for definite clause grammars [35, Chapter 16]; this parallels the notation used in the implementation. In particular, the *terminal ℓ-symbols* of the language, which are all literals, are enclosed in square brackets: for example, [*adj*(X, Y)]. All other literals are *nonterminal ℓ-symbols*. Commas denote concatenation. An empty pair of square brackets [ ] denotes the empty string. Capital letters like A and X denote logical variables. (The number before each rule is a label for use in discussions of the grammar, and is not part of the syntax.)

We have also introduced some special notation. The first line of the grammar serves to declare which predicate is the *goal formula*; that is, it defines the name of the concept to be learned. If the goal formula is G, then the ℓ-nonterminal

[1] Others have simply called these predicates "background knowledge"; we will use the term "feature predicate" here to distinguish this particular type of background knowledge (which is readily made available to a learning system) from other types of background knowledge that are more difficult to exploit.

body(G) is the designated *start* ℓ*-symbol* of the grammar. Finally, if P is a Prolog goal, a rule of the form

$$A \rightarrow B \text{ where } P$$

is simply shorthand for the set of rules

$$\begin{aligned} A\theta_1 &\rightarrow B\theta_1 \\ A\theta_2 &\rightarrow B\theta_2 \\ &\vdots \\ A\theta_n &\rightarrow B\theta_n \end{aligned}$$

where $\theta_1, \ldots, \theta_n$ are the substitutions associated with the various proofs of the goal P. In the current implementation, these rules are generated by a macro-expansion process when the grammar is read in. For example, rule 4 of Fig.1 is expanded to the following set of thirty-six rules[2]

$$\begin{aligned} &rel(A,B,C,D,E,F) \rightarrow pred(A,A). \\ &rel(A,B,C,D,E,F) \rightarrow pred(A,B). \\ &\vdots \\ &rel(A,B,C,D,E,F) \rightarrow pred(F,F). \end{aligned}$$

The semantics of the grammar are as one would expect; it differs from a standard context-free grammar only in manipulating logical terms, rather than symbols from a fixed alphabet. Thus, in the grammar of Fig. 1, the start ℓ-symbol $body(illegal(A,B,C,D,E,F))$ expands to the ℓ-symbol $rels(A,B,C,D,E,F)$; this ℓ-symbol, in turn, expands to an list (of arbitrary length) of $rel(A,B,C,D,E,F)$ ℓ-symbols. Each $rel(A,B,C,D,E,F)$ ℓ-symbol expands to some relationship between some pair of the variables A, B, C, D, E and F as follows. The ℓ-symbol $rel(A,B,C,D,E,F)$ first expands to an ℓ-symbol of the form $pred(X,Y)$, where X and Y are one of the logical variables A, B, C, D, E, or F. Thus this set of rules picks a pair of variables that will be related by some predicate. Then, an ℓ-symbol of the form $pred(X,Y)$ rewrites to one of the feature predicates *adj*, *equal*, or *less_than*, negated or un-negated, with the arguments bound appropriately.

Thus, using the grammar, $body(illegal(A,B,C,D,E,F))$ can be rewritten to strings of literals of the form $L_1, \ldots, L_k$ where k is arbitrarily large, and each L_i is any of the feature predicates *adj*, *equal*, or *less_than* with its arguments bound to any of the variables *A*, *B*, *C*, *D*, *E*, or *F*. As used by our learning system, this grammar defines a relatively weak learning bias, comparable to the

[2] There is a corresponding notation which can be used to express a long right-hand side of a single grammar rule; an example of this will be given in Section 4.3.

bias shown by FOIL.[3] Later we will show how this bias can be strengthened[4] by modifying the antecedent description grammar.

Readers familiar with our previous work on theory specialization [9] will probably notice that there is a close connection between antecedent description grammars and overgeneral theories: grammar rules correspond to clauses in an overgeneral theory, and deriving a sentence in the grammar corresponds to finding an operationalization of the theory. (This connection is exploited in the results of Section 2.6, and is explained at length in Section 5.2.) It might be wondered if it is possible to recast the process of learning given a grammatical bias as a theory specialization problem. However, while grammatically guided learning tasks can be recast as theory specialization problems, the theories which must be constructed are often extremely artificial; this point is illustrated by many of the learning problems described in this paper. We thus prefer to introduce the new and more natural notion of an explicit grammatical bias.

Finally, we note that it is easy to show that the addition of logical variables makes antecedent description grammars strictly more powerful than context-free grammars; this additional computation power is necessary to encode several sorts of background knowledge, such as the knowledge encoded in nonpropositional domain theories.

2.2. Derivations and languages

We will now give a more rigorous description of antecedent description grammars. If $\alpha A'\beta$ is a string of ℓ-symbols, there is a rule $A \rightarrow \gamma$ in the grammar $\mathcal{G}$, and A' and A have a most general unifier (mgu) θ, then we say that $\alpha A'\beta$ *derives* $(\alpha\gamma\beta)\theta$ *in one step in* $\mathcal{G}$, and write

$$\alpha A'\beta \overset{\mathcal{G}}{\Rightarrow} (\alpha\gamma\beta)\theta.$$

If $\alpha_1, \ldots, \alpha_n$ are strings of ℓ-symbols and

$$\alpha_1 \overset{\mathcal{G}}{\Rightarrow} \alpha_2 \overset{\mathcal{G}}{\Rightarrow} \cdots \overset{\mathcal{G}}{\Rightarrow} \alpha_{n-1} \overset{\mathcal{G}}{\Rightarrow} \alpha_n$$

then we say that α_1 *derives* α_n *in* $\mathcal{G}$, and write

$$\alpha_1 \overset{\mathcal{G}}{\Rightarrow}{}^{*} \alpha_n$$

If the derivation is of length one or greater, we will write $\alpha_1 \overset{\mathcal{G}}{\Rightarrow}{}^{+} \alpha_n$ When the grammar $\mathcal{G}$ is clear from context, the superscript will be omitted, and we will write simply $\alpha \Rightarrow \beta$, $\alpha \Rightarrow^{*} \beta$, or $\alpha \Rightarrow^{+} \beta$.

---

[3] The only difference between this bias and FOIL's bias is that while FOIL allows new variables to be introduced in the antecedent of a rule, this grammar does not.

[4] We will also show how the bias can be weakened, for example by allowing new variables to be generated.

Derivation 1:

$body(illegal(A,B,C,D,E,F))$

$\Rightarrow rels(A,B,C,D,E,F)$ using rule 1
$\Rightarrow rel(A,B,C,D,E,F),\ rels(A,B,C,D,E,F)$ using rule 3
$\Rightarrow pred(A,E),\ rels(A,B,C,D,E,F)$ using rule 4
$\Rightarrow adj(A,E),\ rels(A,B,C,D,E,F)$ using rule 7
$\Rightarrow adj(A,E),\ rel(A,B,C,D,E,F),\ rels(A,B,C,D,E,F)$ using rule 3
$\Rightarrow adj(A,E),\ pred(B,F),\ rels(A,B,C,D,E,F)$ using rule 4
$\Rightarrow adj(A,E),\ adj(B,F),rels(A,B,C,D,E,F)$ using rule 7
$\Rightarrow adj(A,E),\ adj(B,F)$ using rule 2

Derivation 2:

$body(illegal(A,B,C,D,E,F))$

$\Rightarrow rels(A,B,C,D,E,F)$ using rule 1
$\Rightarrow rel(A,B,C,D,E,F),\ rels(A,B,C,D,E,F)$ using rule 3
$\Rightarrow rel(A,B,C,D,E,F)$ using rule 2
$\Rightarrow pred(C,E)$ using rule 4
$\Rightarrow C = E$ using rule 5

Fig. 2. Sample derivations in the example grammar.

If S is the start ℓ-symbol, then a string of ℓ-symbols α such that $S \Rightarrow^* \alpha$ is called a *sentential form* of the grammar. A sentential form that contains only terminal ℓ-symbols is a *sentence* of the language. The *language* of a string of ℓ-nonterminals S, written $\mathcal{L}(S)$, is the set of sentences derivable from S; the *language* of a grammar $\mathcal{G}$, denoted $\mathcal{L}(\mathcal{G})$, is the language of its start ℓ-symbol.

For the grammar of Fig. 1, the sentential forms are those strings of ℓ-symbols derivable from the ℓ-symbol $body(illegal(A,B,C,D,E,F))$. Fig. 2 gives some example derivations in the grammar of Fig. 1; the end product of the derivations are sentences, and all of the intermediate results are sentential forms.

2.3. A plausible next step

The natural step to take at this point is to consider learners that take as input an antecedent description grammar $\mathcal{G}$, and output hypotheses that are sets of clauses of the form

$$p(X_1,\ldots,X_n) :- body_1$$
$$p(X_1,\ldots,X_n) :- body_2$$
$$\vdots$$
$$p(X_1,\ldots,X_n) :- body_k$$

where each $body_i$ is not an arbitrary conjunction of literals, but a sentence of $\mathcal{G}$, converted from a string of ℓ-symbols to a conjunction. Such a learner could

be biased—i.e., given background knowledge about the target concept—by appropriately restricting the antecedent description language.

In this paper, we will consider instead a slight variant of this learning problem. In particular, we will allow the learner to propose as a hypothesis a set of clauses where each $body_i$ is a *sentential form*, rather than a sentence, of the antecedent description language. To motivate this variant learning problem, we begin with a slight digression on what properties make a hypothesis space useful for learning (where the *hypothesis space* of a learner refers to the set of possible hypotheses that it can generate).

2.4. Desirable properties for a hypothesis space

Learning can be thought of as search through the hypothesis space for a hypothesis that fits the data [19]. In principle, weak methods like enumeration can be used to perform this search; however, most practical learning systems make use of special properties of the hypothesis space in order to search the space more efficiently. The following have empirically proven to be desirable properties for a learner's hypothesis space to have.

- *Ease of "navigation".* Many learning systems make use of the fact that the hypothesis space is partially ordered under generality to search the hypothesis space—usually in either general-to-specific or specific-to-general order. Such "navigation" through the hypothesis space is easiest if the position of a hypothesis in the partial order (i.e., its generality) is reflected in its syntax; when this is true, syntactic changes to a hypothesis can be used to navigate up and down the partial ordering.
- *Efficiency of testing membership.* Usually many hypotheses are tested against the data in searching through the hypothesis space. Thus being able to efficiently test membership in a hypothesis is important.
- *Perspicuity of hypotheses.* Although for some applications this consideration is of secondary importance, in general it is useful if hypotheses are understandable to people.

These considerations partially explain the success of representations based on propositional logic, which is perspicuous, tractable, and has a syntax which closely reflects its semantics (thus making "navigation" easy). To a lesser extent, these considerations explain the popularity of learning systems based on predicate logic; the main difference is that representations based on predicate logic are less likely to be tractable. One motivation for considering the learnability of Horn clauses is that they are believed to be a relatively tractable subset of predicate logic.

2.5. Solving the navigation problem by using sentential forms

The hypothesis space described in Section 2.3—namely, the space of theories comprised of Horn clauses whose antecedents are sentences of an antecedent

description grammar—is reasonably efficient and perspicuous.[5] However, it is difficult to navigate through the space of sentences of the antecedent description grammar; while the grammar could be used to determine which syntactic changes to a clause are legal, it is difficult to predict the effect of these changes on the semantics of a clause; in short, it is hard to predict if the changes generalize or specialize a clause. We call this problem the *navigation problem.*

This problem can be partially solved by first, allowing the learner to hypothesize sentential forms as well as sentences of the grammar and second, defining an appropriate semantics for sentential forms. We will define the semantics of a sentential form so that it is a *generalization* of the clauses it derives; thus the syntactic relationship $\alpha \Rightarrow^* \beta$ can be used to determine the semantic relationship that α is more general than β.

In particular, let $ext(A :- B)$ denote the *extension* of the Horn clause $A :- B$; that is, the set of literals classified as "true" by the clause. (More precisely, $A' \in ext(A :- B)$ iff A' and A have an mgu θ and $B\theta$ is provable.) Let $translate_s(\alpha)$ be a function that inputs a string of ℓ-symbols $\alpha = A_1 \ldots A_n$ and outputs the corresponding conjunction of literals $A_1 \wedge \cdots \wedge A_n$; $translate_s([\,])$ is defined to output the literal *true*, which always succeeds.[6] Let α be a sentential form, and let $body(G)$ be the start ℓ-symbol from which α was derived. We define

$$ext(\alpha) \equiv \bigcup_{\beta \in \mathcal{L}(\alpha)} ext(G :- translate_s(\beta)).$$

The semantics of sentential forms are thus inherited from the semantics of Horn clauses.

For example, if the start ℓ-symbol is $body(illegal(A,B,C,D,E,F))$, then the sentential form

$$adj(B,D), pred(D,F)$$

has the following extension:

$$\begin{aligned}
&ext(illegal(A,B,C,D,E,F) :- adj(B,D) \wedge D = F)\\
\cup\ &ext(illegal(A,B,C,D,E,F) :- adj(B,D) \wedge \neg(D = F))\\
\cup\ &ext(illegal(A,B,C,D,E,F) :- adj(B,D) \wedge adj(D,F))\\
\cup\ &ext(illegal(A,B,C,D,E,F) :- adj(B,D) \wedge \neg adj(D,F))\\
\cup\ &ext(illegal(A,B,C,D,E,F) :- adj(B,D) \wedge less_than(D,F))\\
\cup\ &ext(illegal(A,B,C,D,E,F) :- adj(B,D) \wedge \neg less_than(D,F)).
\end{aligned}$$

A consequence of this definition is that the derivability relationship has a direct correspondence to generality: if $\alpha \Rightarrow^+ \beta$, then α is at least as general

[5] If a suitable grammar is used; it is of course possible to design a grammar such that hypotheses will be neither efficient nor perspicuous. In this paper we will always be assuming that the antecedent description grammar is appropriate for the problem at hand; the experimental section of the paper shows that such grammars do sometimes exist.

[6] We are again using definite clause notation; [ ] denotes the empty string.

as β. Thus, this interpretation of sentential forms largely solves the navigation problem.

We say "largely solves" rather than "solves" for this reason: although it would be preferable if $\alpha \Rightarrow^+ \beta$ implied that α was strictly more general than β, this is quite often not the case. The fact that the partial order of generality is nonstrict leads to some minor difficulties in learning, which are discussed in Section 3.2.2.

2.6. *Implementing the semantics of sentential forms*

To use sentential forms as a hypothesis space we also require a way of taking a sentential form α and efficiently checking whether or not some instance $p(t_1,\ldots,t_n)$ is a member of $ext(\alpha)$. Recall that by definition, an instance $p(t_1,\ldots,t_n)$ is a member of $ext(\alpha)$ iff $p(t_1,\ldots,t_n)$ is a member of $ext(G :- \beta)$ for some β such that $\alpha \Rightarrow^* \beta$; thus checking membership of an instance $p(t_1,\ldots,t_n)$ in a clause $G :- \alpha$ can be broken down into two steps:

(1) finding a β such that $\alpha \Rightarrow^* \beta$,

(2) verifying that $p(t_1,\ldots,t_n) \in ext(G :- \beta)$.

There is clearly some work involved in testing these conditions, since we need to search for the right β. How can this search be performed efficiently? One way of performing this search problem is suggested by the observation that the operation of rewriting a string of ℓ-symbols via a rule $A \rightarrow B$ is much the same as the operation of resolving a conjunction of literals against the Horn clause $A :- B$. Similarly, searching for a β that can be derived from α is much like constructing a Horn proof; this suggest that a Horn clause theorem prover can be adapted to perform this search.

In fact, it is straightforward to enlist a Horn theorem prover to this task. When the grammar $\mathcal{G}$ is read in, we create for each grammar rule $A \rightarrow B$ in $\mathcal{G}$ a Horn clause of the form $A :- translate_s(B)$. These Horn clauses are added to the clauses which define the feature predicates. The resulting theory we will denote as $Th_{\mathcal{G}}$. To continue with our running example, Fig. 3 shows the theory which would be created by this procedure from the grammar of Fig. 1.

Now, when testing to see if a literal $p(t_1,\ldots,t_n)$ is a member of $ext(\alpha)$, we can simply use a Horn theorem prover to test if $p(t_1,\ldots,t_n)$ is a member of $ext(G :- translate_s(\alpha))$. The following theorem shows that this procedure is correct (and incidentally clarifies the close connection between theory specialization and grammatically biased learning).

Theorem 2.1. *Let $\mathcal{G}$ be an antecedent description grammar, let $body(G)$ be the associated start ℓ-symbol, let α be a sentential form derived from $body(G)$, and let $Th_{\mathcal{G}}$ be the Horn theory created using the construction above. Then*

$$p(t_1,\ldots,t_n) \in ext(\alpha) \quad \textit{iff} \quad p(t_1,\ldots,t_n) \in ext(G :- translate_s(\alpha)).$$

Proof. See Appendix A. □

$body(illegal(A,B,C,D,E,F)) :- rels(A,B,C,D,E,F).$

$rels(A,B,C,D,E,F) :- true.$
$rels(A,B,C,D,E,F) :- rel(A,B,C,D,E,F) \wedge rels(A,B,C,D,E,F).$

$rel(A,B,C,D,E,F) :- pred(A,A).$
$rel(A,B,C,D,E,F) :- pred(A,B).$
$\vdots$
$rel(A,B,C,D,E,F) :- pred(F,F).$

$pred(X,Y) :- X = Y.$
$pred(X,Y) :- \neg X = Y.$
$pred(X,Y) :- adj(X,Y).$
$pred(X,Y) :- \neg adj(X,Y).$
$pred(X,Y) :- less\_than(X,Y).$
$pred(X,Y) :- \neg less\_than(X,Y).$

Fig. 3. Derived theory $Th_{\mathcal{G}}$ for the example grammar.

When reading in the grammar $\mathcal{G}$**:**
let the Horn theory $Th_{\mathcal{G}}$ initially contain the clauses defining the feature predicates
macro-expand each rule written "$A \rightarrow B$ where P"
for each rule $A \rightarrow B$ in $\mathcal{G}$ **do**
  add the clause A :- $translate_s(B)$ to $Th_{\mathcal{G}}$
endfor

To test if $p(t_1,\ldots,t_n) \in \boldsymbol{ext}(\alpha)$:
let $body(G)$ be the start ℓ-symbol from which α was derived
if G and $p(t_1,\ldots,t_n)$ have a mgu θ **then**
  if $translate_s(\alpha\theta)$ is provable from $Th_{\mathcal{G}}$ **then**
    return true
return false

where $translate_s(\alpha) \equiv \begin{cases} A_1 \wedge \cdots \wedge A_k, & \text{if } \alpha = A_1,\ldots,A_k, \\ true, & \text{if } \alpha \text{ is the empty string.} \end{cases}$

Fig. 4. Naive theory creation and membership test.

Thus membership in a sentential form can be checked using only a Horn clause theorem prover, and a small amount of preprocessing. The algorithm is shown in Fig. 4. In the next section, we will show how this naive membership test can be improved.

2.7. Improving perspicuity and efficiency

One cost in allowing the learner to hypothesize sentential forms, rather than sentences, is that the hypotheses of the learner are more difficult to understand. In general, the clauses hypothesized by the learner contain ℓ-nonterminals as well as ℓ-terminals; while terminal ℓ-symbols correspond to predicates that the user has selected as being meaningful, nonterminal ℓ-symbols do not. As an example, the following translation of a sentential form

$$A = C \wedge B = D \wedge rels(A, B, C, D, E, F).$$

is certainly not as meaningful as the following translation of a sentence

$$A = C \wedge B = D.$$

True, for each nonterminal ℓ-symbol there is a predicate defined in the theory Th_G, which could in principle be used to interpret it; however, even if the grammar rules are sensible, the corresponding predicate definitions may not be. In our example grammar, for instance, every predicate corresponding to an ℓ-nonterminal always succeeds, and thus expresses a vacuous condition. This loss in perspicuity is paralleled by a loss in efficiency: testing a hypothesis against the data is slowed by having to repeatedly prove these vacuous predicates.

These problems can be addressed by *simplifying* the clauses hypothesized by the learner before they are used. In testing membership of an instance $p(t_1, \ldots, p_n)$ in $ext(\alpha)$, we first construct the clause G :- $translate_s(\alpha\theta)$ described above, but then simplify the body of the clause before sending it to the theorem prover. Similarly, when the learning system produces a final hypothesis

$$\begin{aligned} G &:- body_1 \\ G &:- body_2 \\ &\vdots \\ G &:- body_k \end{aligned}$$

each $body_i$ is simplified before the hypothesis is presented to the user. Simplification thus improves both efficiency and perspicuity.

The simplification techniques we use are designed to remove vacuous predicates whenever possible—in particular, to remove those vacuous predicates that correspond to nonterminal ℓ-symbols of the antecedent description grammar.[7] The simplification technique is sound, but not complete: i.e., it is guaranteed to perform only simplifications that do not change the meaning of a clause, but is not guaranteed to perform all such simplifications. The basic idea of the

[7] It may seem odd that removing vacuous predicates is the only simplification that we attempt; however, the experiments of Sections 4 and 5 indicate that this is the crucial optimization for many types of problems.

simplification procedure is to use static analysis of the theory Th_G to determine which predicates of Th_G always succeed. These predicates can then be simply discarded from any conjunctive goal.

In more detail, the static analysis phase proceeds as follows. In the first phase, we look for predicates that can easily be shown to always succeed. In the second phase, we propagate this information. Currently, the following conditions are tested in the first phase to see if A always succeeds.

(1) There is a clause of the form A :- *true*.
(2) There are a pair of clauses of the form A :- B and A :- $\neg B$ that (except for the sign of B) are substitutional variants.
(3) The predicate is declared by the user to always succeed.

For example, consider the theory of Fig. 3. In phase one, the predicate[8] *rels/6* will be marked as "always true" since there is a clause

$$rels(A, B, C, D, E, F) \text{ :- } true$$

in Th_G; also, *pred/2* will be marked "always true" since there are a pair of clauses

$$pred(X, Y) \text{ :- } X = Y,$$
$$pred(X, Y) \text{ :- } \neg(X = Y).$$

In the second phase, this information is propagated to other predicates by looking for clauses of the form

$$A \text{ :- } B_1 \wedge \cdots \wedge B_k$$

where each B_k has already been marked as always succeeding. Whenever such a clause is found, A is also marked as always succeeding. The propagation phase ends when there no such clauses are found.

For example, in the theory of Fig. 3, the predicates *rel/6* and *body/1* will be marked as "always true" in the first iteration of the propagation procedure; *rel/6* by virtue of the clause

$$rel(A, B, C, D, E, F) \text{ :- } pred(A, A)$$

since *pred/2* is marked as "always true", and *body/1* by virtue of the clause

$$body(illegal(A, B, C, D, E, F)) \text{ :- } rels(A, B, C, D, E, F)$$

since *rels/6* is marked as "always true". On the second pass, no new predicates are discovered to be "always true", and hence the propagation phase terminates. Notice that static analysis need be done only once, when the grammar is initially read in.

[8] For the purpose of static analysis, a predicate is defined by its principle functor (e.g., *rels*) and its arity (i.e., its number of arguments).

In general, it is undecidable to determine if a predicate will always succeed; thus conditions like the ones tested above will always be only sufficient conditions, not necessary and sufficient conditions, for detecting that a predicate is vacuous. This problem is partially addressed by providing an *always_true* declaration, which allows the user to give the appropriate information to the system. This declaration is not needed for any of the learning problems studied in this paper; however, it is easy to construct examples where it is necessary. For example, if $pred(X, Y)$ were defined as

$$pred(X, Y) :- less\_than(X, Y),$$
$$pred(X, Y) :- X = Y,$$
$$pred(X, Y) :- less\_than(Y, X),$$

then *pred/2* would not be detected in the first phase as being vacuous, although it is easy to see that it always succeeds.

Given the results of the static analysis, simplification is trivial; one merely drops from a conjunction those predicates that are guaranteed to always succeed. For example, given the static analysis above, the conjunction

$$A = C \wedge B = D \wedge rels(A, B, C, D, E, F)$$

can be simplified to

$$A = C \wedge B = D.$$

It is important, of course, that simplification is inexpensive, since it is performed whenever a hypothesis is tested against the data.

The improved procedures for checking membership in a sentential form and preprocessing a theory, including the algorithms for static analysis and simplification of sentential forms, are shown in Fig. 5.

3. Learning using antecedent description grammars

We have now constructed a hypothesis space in which it is easy to navigate, has an efficient and easy-to-implement membership test, and is reasonably perspicuous, in the sense that hypotheses from this space (usually) have a clear meaning. The next step is to develop a learning algorithm for this hypothesis space.

3.1. The FOIL learning algorithm

The learning algorithm we have developed is an adaptation of the algorithm used in Quinlan's FOIL system [28]; we will now briefly review the FOIL algorithm. This review is useful both as background, and also because we will

When reading in the grammar $\mathcal{G}$:
let the Horn theory $Th_{\mathcal{G}}$ initially contain the clauses defining the feature predicates
macro-expand each rule written "$A \rightarrow B$ where P"
for each rule $A \rightarrow B$ in $\mathcal{G}$ **do**
  add the clause A :- $translate_s(B)$ to $Th_{\mathcal{G}}$
(*perform static analysis on* $Th_{\mathcal{G}}$)
for each clause A :- *true* **do**
  mark a/n as "always true", for (a, n) = *functor_arity*(A)
for each pair of clauses A_1 :- L_1 and A_2 :- $\neg L_2$ so that
    A_1 :- L_1 and A_2 :- L_2 are substitutional variants **do**
  mark a/n as "always true", for (a, n) = *functor_arity*(A_1)
repeat
  if there is a clause A :- $B_1 \wedge \cdots \wedge B_n$ so that each b_i/n_i is marked "always true", for (b_i, n_i) = *functor_arity*(B_i) **then**
    mark a/n as "always true", for (a, n) = *functor_arity*(A)
until nothing was marked in the last iteration

To test if $p(t_1, \ldots, t_n) \in \boldsymbol{ext}(\alpha)$:
let $body(G)$ be the start ℓ-symbol from which α was derived
if G and $p(t_1, \ldots, t_n)$ have a mgu θ **then**
  if $simplify(translate_s(\alpha\theta))$ is provable from $Th_{\mathcal{G}}$ **then**
    return true
return false

where

$$functor\_arity(A) \equiv \text{the pair } (a, n) \text{ so that } A \text{ unifies with } a(X_1, \ldots, X_n),$$

$$translate_s(\alpha) \equiv \begin{cases} A_1 \wedge \cdots \wedge A_k, & \text{if } \alpha = A_1, \ldots, A_k, \\ true, & \text{if } \alpha \text{ is the empty string,} \end{cases}$$

$$simplify(A_1 \wedge \cdots \wedge A_k) \equiv \begin{cases} true, & \text{if } k = 0, \\ simplify(A_2 \wedge \cdots \wedge A_k), & \text{if } a_1/n_1 \text{ is "always true", for} \\ & \quad (a_1, n_1) = functor\_arity(A_1) \\ A_1 \wedge simplify(A_2 \wedge \cdots \wedge A_k), & \text{otherwise.} \end{cases}$$

Fig. 5. Improved theory creation and membership test.

The outer loop of FOIL is the following procedure.

- Let $i = 1$, P be the complete set of positive examples, and N be the complete set of negative examples.
- While P is nonempty:
  - find a clause C_i that covers some positive examples in P,
  - remove the positive examples covered by P from C_i, and then
  - increment i.

- Return the theory $\{C_1, \ldots, C_{i-1}\}$ as the hypothesis.

The more interesting part of the FOIL algorithm is the procedure used to find each clause C_i. The inner loop of FOIL builds a new clause C_i by starting with the clause G :- *true* (where G is the goal formula) and repeatedly specializing it by adding a new literal L to the antecedent. The inner loop of FOIL is "greedy" in the sense that once a literal is added to C_i, it is never removed. In determining which literal L to add onto the end of C_i, FOIL considers all possible literals that could be added on, given the fixed set of input relations; the literal that is chosen is one which maximizes an information-theoretic measure of clause quality called *information gain.*

In a bit more detail, the operation of the inner loop is as follows.

- Let C_i be the clause G :- *true.*
- While C_i covers some negative examples:
  - For each literal L_j which shares some variables with the variables of C_i, compute the *information gain* of the clause G :- $B \wedge L_j$ relative to the clause G :- B, where B is the body of C_i.
  - Replace C_i with the clause G :- $B \wedge L_{\max}$, where $L_{\max}$ is the literal that results in the largest information gain.

The final critical component of FOIL is the *information gain* heuristic, which is a means of evaluating the quality of a clause. *Information gain* can be defined as follows. Let $X_1, \ldots, X_k$ be the variables that occur in the clause C_i, and let $\theta_1, \ldots, \theta_n$ be the substitutions associated with the various proofs that $p(t_1, \ldots, t_l) \in ext(C_i)$. The n k-tuples

$$\langle X_1\theta_1, \ldots, X_k\theta_1\rangle, \ldots, \langle X_1\theta_n, \ldots, X_k\theta_n\rangle$$

are called the *tuples of* C_i *for* $p(t_1, \ldots, t_l)$. A tuple is a *positive tuple of* C_i if it is a tuple of C_i for any positive example; *negative tuples* are defined analogously. The information gain of clause C_{i+1} relative to clause C_i can be defined as

$$Gain(C_{i+1}, C_i) \equiv T_i^{++} \times \left(-\log_2 \frac{T_i^+}{T_i^+ + T_i^-} + \log_2 \frac{T_{i+1}^+}{T_{i+1}^+ + T_{i+1}^-} \right)$$

where T_j^+ (respectively T_j^-) is the number of positive (negative) tuples of clause C_j, and T^{++} is the number of positive tuples in C_i that correspond to one or more tuples in C_{i+1}. For more discussion of information gain, see [28].[9]

[9] Information gain is the primary means by which clauses are evaluated; however, we follow Quinlan in giving a small bonus to clauses which introduce a new variable. The rationale for this is that introducing a new variable sometimes paves the way for a subsequent useful specialization. We also impose a small penalty on clauses C_{i+1} which cover no positive examples. To see why this is useful, note that there are two cases in which information gain is zero. If C_{i+1} covers no positive examples, then $T_i^{++} = 0$, and the information gain is zero. There is clearly no value in including such a clause, or any specialization of it, in a hypothesis. However, if C_{i+1} covers exactly the same tuples as C_i, the information gain is also zero. In contrast to the previous situation, however, it may be that some specialization of C_{i+1} is useful. Adding a penalty for covering no positive examples serves to distinguish between these two cases.

It should be noted that we have described here a somewhat simplified version of Quinlan's actual FOIL program; in particular, we are not considering important features that prune the search for new literals, deal with noisy data, postprocess the clauses of the hypothesis, and infer partial orderings among the clauses so that recursive definitions can be learned. However, this simplified version is very competent at learning nonrecursive relational concepts in noise-free conditions, and is adequate as a benchmark against which to compare our system; furthermore, we believe that most of the extensions mentioned above can be considered independently of the problem of providing a grammatical bias.

3.2. *Modifying FOIL*

3.2.1. *The basic algorithm*

Consider now the generalization of the FOIL learning algorithm described in Fig. 6. The outer loop is the same; in the inner loop, rather than starting with a clause G :- *true* and repeatedly adding a literal, the algorithm starts by letting C be some "universal concept" which is always true, and then repeatedly specializes C by replacing it with some element of a set of "designated refinements" of C. As in FOIL, this process terminates when C covers no negative examples. We will call this generalization EFOIL. EFOIL retains the basic control structure of the FOIL algorithm, and, like FOIL, returns a set of concepts $C_1,\ldots,C_n$ that together cover the positive examples; however, it abstracts away the particular representation used for each C_i.

We will now describe an alternative instantiation of EFOIL, called GRENDEL,[10] which uses as concepts sentential forms of an antecedent description grammar. The *universal concept* will simply be the start symbol of the grammar. To measure the information gain of the sentential form α relative to β, we simply convert α and β to the clauses

$$G \text{ :- } \mathit{simplify}(\mathit{translate}_s(\alpha)),$$
$$G \text{ :- } \mathit{simplify}(\mathit{translate}_s(\beta)),$$

and compute the information gain of these clauses, using the same procedure used by FOIL. It remains only to describe the set of designated refinements of a sentential form.

3.2.2. *Designated refinements of a sentential form*

An obvious choice would be to let *Designated_Refinements*(α) return the set of all sentential forms β that can be derived by applying a single grammar rule to α; in other words, to let

$$\mathit{Designated\_Refinements}(\alpha) \equiv \{\beta : \alpha \Rightarrow \beta\}.$$

[10] For Grammatically REstricted Non DEductive Learner.

function EFOIL(P, N) :
begin
  while P is not empty **do**
    find a concept that covers some positive examples
    let C_i = *Universal_Concept*()
    while C_i covers some negative examples **do**
      $\mathcal{R} \leftarrow$ *Designated_Refinements*(C_i)
      replace C_i with the $C_i' \in \mathcal{R}$ such that
        Information_Gain(C_i', C_i, P, N) is maximal
    endwhile
    replace P with $P - ext(C_i)$
    add C_i to the hypothesis H
  endwhile
  return H
end

where to instantiate EFOIL as FOIL one uses
  Universal_Concept() $\equiv G$:- *true*,
    where G is the goal formula,
  Designated_Refinements(G :- B)
    $\equiv \{G$:- $B \wedge L$: literal L shares variables with G or $B\}$,
  Information_Gain(C_i', C_i, P, N) $\equiv$ as defined in the text,

and to instantiate EFOIL as GRENDEL one uses
  Universal_Concept() $\equiv$ *body*(G),
    where G is the goal formula,
  Designated_Refinements(α)
    $\equiv \{\beta$: $\alpha \Rightarrow^{+}_{\text{lin}} \beta \wedge$ the rule sequence of $\alpha \Rightarrow^{+}_{\text{lin}} \beta$ is nonlooping,$\}$
  Information_Gain(α, β, P, N)
    $\equiv$ information gain of the corresponding clauses.

Fig. 6. The EFOIL algorithm.

However, this choice turns out to present difficulties for the EFOIL algorithm; the problem is that often there will be two different designated refinements β_1 and β_2 that have identical extensions, but which lead to different areas of the search space. In this case, the information gain heuristic will not be able to make a sensible choice between β_1 and β_2, no matter how large the set of training examples.

To illustrate this problem, consider the possible refinements of the sentential form *rels*(A, B, C, D, E, F). There are two applicable rules, leading to either the empty string [ ] or to the sentential form

rel(A, B, C, D, E, F), *rels*(A, B, C, D, E, F).

However, both of these correspond (after simplification) to the clause

$$illegal(A, B, C, D, E, F) :- true$$

and hence EFOIL can make no sensible choice between them. However, they are radically different, as the second sentential form can be further refined to many different reasonable hypotheses, while the first cannot.

This problem is solved by allowing the *Designated_Refinement* function to return all β that can be derived from α via a "nonlooping linear sequence" of rewrites. Intuitively, a "linear sequence" of rewrites is a sequence of rewrites in which each grammar rule rewrites a symbol that was introduced by the immediately preceding rule; a nonlooping sequence of rewrites is one in which no grammar rule is used more than once. By allowing the sequence of rewrites to be of any length, we force the *Designated_Refinements* of any α to include some sentential forms that contain ℓ-terminals; hopefully, the information gain criterion will be able to make sensible choices among at least these possible refinements. However, by requiring the sequence of rewrites to be linear and by not allowing recursive grammar rules to be used more than once, we reduce the number of possible designated refinements to a reasonable number; empirically, the branching factor of our learning algorithm (using this *Designated_Refinement* routine) is comparable to the branching factor of FOIL when the antecedent description language induces a comparable bias.

More precisely, let us define the relation $\Rightarrow^+_{\text{lin}}$ as follows.

- If $\alpha \Rightarrow \beta$, then $\alpha \Rightarrow^+_{\text{lin}} \beta$.
- If

$$\alpha = A_1, \ldots, A_{i-1}, A_i, A_{i+1}, \ldots A_n,$$
$$A_i \Rightarrow \gamma,$$
$$\gamma \Rightarrow^+_{\text{lin}} \gamma',$$

and

$$\beta = A_1, \ldots, A_{i-1}, \gamma', A_{i+1}, \ldots, A_n$$

then $\alpha \Rightarrow^+_{\text{lin}} \beta$.

Finally, if $\alpha_0 \Rightarrow^+_{\text{lin}} \alpha_n$ via the series of rewrites $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n$ and for each i, $1 \leqslant i \leqslant n$, the rule $A_i \rightarrow \beta_i$ was used to convert α_{i-1} to α_i, we call the sequence

$$A_1 \rightarrow \beta_1,\ A_2 \rightarrow \beta_2,\ \ldots,\ A_n \rightarrow \beta_n$$

the *rule sequence* of the derivation $\alpha_0 \Rightarrow^+_{\text{lin}} \alpha_n$. A rule sequence is *nonlooping* if all of the rules in the sequence are distinct.

The actual implementation of the *Designated_Refinements* subroutine can now be succinctly described as follows:

Designated_Refinements(α)
$\equiv \{\beta : \alpha \Rightarrow^+_{\text{lin}} \beta \wedge$ the rule sequence of $\alpha \Rightarrow^+_{\text{lin}} \beta$ is nonlooping$\}$.

To return to our example, there are now many possible designated refinements of the ℓ-symbol $rels(A,B,C,D,E,F)$ of our example grammar. Using the base case of the recursive definition of $\Rightarrow_{\text{lin}}^{+}$ we see that we can derive either of the strings

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

However, since the derivation can be of any length, the set of designated refinements of $rels(A,B,C,D,E,F)$ also includes (some) strings formable by rewriting *these* ℓ-symbols. For example, the string

$$rel(A,B,C,D,E,F)$$

(formed by rewriting $rels(A,B,C,D,E,F)$ to the empty string) is also a designated refinement of $rels(A,B,C,D,E,F)$. Note however that although

$$rel(A,B,C,D,E,F), rel(A,B,C,D,E,F), rels(A,B,C,D,E,F)$$

can also be derived from $rel(A,B,C,D,E,F), rels(A,B,C,D,E,F)$, it is *not* a designated refinement, since this derivation requires the rule

$$rels(A,B,C,D,E,F) \rightarrow rel(A,B,C,D,E,F), rels(A,B,C,D,E,F)$$

to be used twice. As another example,

$$pred(A,A), rels(A,B,C,D,E,F)$$

is a designated refinement of $rels(A,B,C,D,E,F)$, as it can be formed by rewriting $rel(A,B,C,D,E,F), rels(A,B,C,D,E,F)$ using the rule

$$rel(A,B,C,D,E,F) \rightarrow pred(A,A).$$

However, $pred(A,A)$ is not a designated refinement, since the derivation

$$\begin{aligned} & rels(A,B,C,D,E,F) \\ \Rightarrow\ & rel(A,B,C,D,E,F), rels(A,B,C,D,E,F) \\ \Rightarrow\ & rel(A,B,C,D,E,F) \\ \Rightarrow\ & pred(A,A) \end{aligned}$$

violates the linearity condition: in the last step, the ℓ-symbol that was rewritten was not introduced in the previous step. Fig. 7 lists the entire set of designated refinements of $rels(A,B,C,D,E,F)$.

3.3. An example

As an example, we will describe the behavior of GRENDEL using the grammar of Fig. 1 and a particular sample of 100 randomly chosen examples. (The reader may wish to refer to Fig. 6.) The sample contains 34 board

$[\,]$
$rel(A,B,C,D,E,F), rels(A,B,C,D,E,F)$
$pred(A,A), rels(A,B,C,D,E,F)$
$A = A, rels(A,B,C,D,E,F)$
$\neg(A = A), rels(A,B,C,D,E,F)$
$adj(A,A), rels(A,B,C,D,E,F)$
$\neg adj(A,A), rels(A,B,C,D,E,F)$
$less\_than(A,A), rels(A,B,C,D,E,F)$
$\neg less\_than(A,A), rels(A,B,C,D,E,F)$
$pred(A,B), rels(A,B,C,D,E,F)$
$A = B, rels(A,B,C,D,E,F)$
$\neg(A = B), rels(A,B,C,D,E,F)$
$\vdots$
$pred(A,C), rels(A,B,C,D,E,F)$
$\vdots$
$\vdots$
$pred(F,F), rels(A,B,C,D,E,F)$
$F = F, rels(A,B,C,D,E,F)$
$\vdots$
$\neg less\_than(F,F), rels(A,B,C,D,E,F)$
$rel(A,B,C,D,E,F)$

Fig. 7. Designated refinements of $rels(A,B,C,D,E,F)$.

positions which are *illegal*, and 66 which are not *illegal*. The examples are encoded as labeled ground atoms

$+illegal(0,2,7,4,0,1)$, $+illegal(1,1,1,3,1,0),\ldots$ % illegal positions
$-illegal(0,0,5,7,4,3)$, $-illegal(0,1,1,0,0,7),\ldots$ % legal positions

In the first iteration of the inner loop, GRENDEL begins with the start ℓ-symbol $body(illegal(A,B,C,D,E,F))$, which corresponds to the clause

$$illegal(A,B,C,D,E,F) :- body(illegal(A,B,C,D,E,F)).$$

Since $body/1$ always succeeds, this clause is simplified to

$$illegal(A,B,C,D,E,F) :- true.$$

Since this clause covers some negative examples, GRENDEL next enumerates the set of designated refinements of it: this set consists of the ℓ-symbol $rels(A,B,C,D,E,F)$ and all the refinements of it shown in Fig. 7. The refinement with maximum gain is the result of the following linear nonlooping derivation:

$$
\begin{aligned}
& body(illegal(A,B,C,D,E,F)) \\
\Rightarrow\ & rels(A,B,C,D,E,F) \\
\Rightarrow\ & rel(A,B,C,D,E,F), rels(A,B,C,D,E,F) \\
\Rightarrow\ & pred(C,E), rels(A,B,C,D,E,F) \\
\Rightarrow\ & C = E, rels(A,B,C,D,E,F).
\end{aligned}
$$

This string of ℓ-symbols corresponds (after translation and simplification) to the clause

$$illegal(A,B,C,D,E,F) :- C = E$$

which has an information gain of 21.405. (Notice that the condition corresponding to $rels(A,B,C,D,E,F)$ is dropped, as it is known to always succeed.) Intuitively, the clause says that a position is illegal if the white rook and the black king are in the same rank. This is not quite true, since the white king can be between the two; as it happens, however, there are no examples of this in the sample. In all, the clause covers 14 positive example and no negative examples; since it covers no negative examples, the inner **while** loop terminates, the clause it is added to the hypothesis, and the positive examples that it covers are removed.

In the second iteration of the outer **while** loop, GRENDEL again begins with the start ℓ-symbol $body(illegal(A,B,C,D,E,F))$ and, since it is not consistent with the negative data, enumerates the designated refinements of it. The set of refinements is the same as in the previous iteration; however, since the sample is different, the information gain of each refinement will be different. On this iteration, GRENDEL chooses the refinement

$$D = F, rels(A,B,C,D,E,F)$$

which corresponds to the clause

$$illegal(A,B,C,D,E,F) :- D = F.$$

This is a dual of the clause that was previously found; it says that a position is illegal if the white rook and the black king are on the same file. It covers ten of the remaining positive examples and one negative example, and has a gain of 19.251. Since the clause still covers some negative examples, GRENDEL will refine it further. On the next iteration of the inner **while** loop, GRENDEL selects the refinement generated by the following derivation.

$$
\begin{aligned}
& D = F, rels(A,B,C,D,E,F) \\
\Rightarrow\ & D = F, rel(A,B,C,D,E,F), rels(A,B,C,D,E,F) \\
\Rightarrow\ & D = F, pred(A,B), rels(A,B,C,D,E,F) \\
\Rightarrow\ & D = F, \neg A = B, rels(A,B,C,D,E,F).
\end{aligned}
$$

This qualification to the preceding clause excludes the single negative example covered by the clause without excluding any of the positive examples. Intuitively, it says that a position is illegal if the white rook and black king are on

the same file, and the white king is not on the main diagonal; this is not a correct refinement of the previous clause (relative to the target concept) but it is very hard to tell this from the examples. GRENDEL thus adds the clause that corresponds to this string of ℓ-symbols to the hypothesis:

$$illegal(A, B, C, D, E, F) \text{ :- } D = F \wedge \neg A = B.$$

24 of the 34 positive examples have now been covered.

Since more positive examples remain to be covered, GRENDEL will continue. On the next iteration of the outer **while** loop, GRENDEL will build the clause

$$illegal(A, B, C, D, E, F) \text{ :- } adj(B, F) \wedge adj(A, E).$$

This clause covers another eight positive examples; it corresponds to the two kings being next to one another or on the same location. Finally, the clause

$$illegal(A, B, C, D, E, F) \text{ :- } B = D \wedge A = C$$

is built, which covers the last two positive examples. This clause is true if the white king and rook are on the same location. Building each of these clauses, of course, requires two iterations of the inner **while** loop.

Since there are no positive examples that are uncovered, the outer **while** loop will terminate. The final hypothesis GRENDEL generates is the theory:

$$illegal(A, B, C, D, E, F) \text{ :- } C = E.$$
$$illegal(A, B, C, D, E, F) \text{ :- } D = F \wedge \neg A = B.$$
$$illegal(A, B, C, D, E, F) \text{ :- } adj(B, F) \wedge adj(A, E).$$
$$illegal(A, B, C, D, E, F) \text{ :- } B = D \wedge A = C.$$

These four clauses are certainly not an exactly correct definition of the *illegal* predicate. The first clause is too general, the second clause is only approximately correct, and several cases are missing: for example, a position is also illegal if the black king and the white rook are on the same location. However, on the whole the theory is quite accurate; testing the hypothesis on 5000 test cases gives an error rate of only 1.68%.

We note in passing that our version of FOIL (which is implemented by simply running the EFOIL procedure with different *Universe* and *Designated_Refinements* procedures) learns almost exactly the same theory: the only difference is the final conjunct of the final clause, which unfortunately FOIL incorrectly guesses to be $\neg less\_than(C, A)$.[11] As a consequence, FOIL's error rate on this problem is 5.32%.

[11] The conditions $\neg less\_than(C, A)$ and $A = C$ discriminate equally well on the data. FOIL and GRENDEL make a different choice simply because, while they both break ties in information gain by using position in the enumeration of possible refinements, the *Designated_Refinements* procedures they use enumerate refinements in different orders.

4. Results for empirical learning problems

In this section, we will use antecedent description grammars to improve performance on various types of empirical learning (or similarity-based learning) problems. The following section will use antecedent description grammars to improve performance on various types of knowledge-based learning tasks. The main purposes of these two sections are first, to show by example how antecedent description grammars can be used, and second, to convince the reader of the generality of our approach.

Most of the experiments we perform will be of the following type. First, we will describe some type of background knowledge. Then, we will encode this knowledge in an appropriate grammar, and demonstrate that GRENDEL can use this knowledge effectively. In most cases, the type of background knowledge that we consider has been previously described in the literature, and some learning system exists that can use this sort of knowledge; in these cases, our experiments can be viewed as using GRENDEL to perform a qualitative simulation of the earlier system. (Or course, performing this qualitative simulation usually means that we must show quantitative improvement as knowledge is added to the learner.)

In this paper, we will distinguish between "active" and "passive" biases. Background knowledge which actually restricts the hypothesis space, rather than simply removing redundancy, is called an *active bias*. Background knowledge that simply removes redundancy from a hypothesis space is called a *passive bias*. In this section of the paper, we will consider passive biases and some rather weak types of active bias, notably typing information. The next section will consider stronger types of active bias, in particular active biases derived from various sorts of background-knowledge theories.

4.1. Eliminating symmetries and vacuous conditions

One simple type of background knowledge is knowledge about how the arguments to predicates can be instantiated. In particular, for the *illegal* problem, there are terminal predicates which can be either vacuously true or vacuously false: for example, $less\_than(X, X)$ always fails, and $equal(X, X)$ always succeeds. Avoiding such vacuous predicates is an example of a passive bias. Similarly, since $equal(X, Y)$ is logically equivalent to $equal(Y, X)$, only one of these predicates need be tested.

On the *illegal* problem, this sort of information can be encoded by first, replacing the rule

$$rel(A, B, C, D, E, F) \rightarrow pred(X, Y)$$
$$\text{where } member(X, [A, B, C, D, E, F]), member(Y, [A, B, C, D, E, F]).$$

with the rule

$$rel(A,B,C,D,E,F) \rightarrow pred(X,Y)$$
$$\text{where } subset([X,Y],[A,B,C,D,E,F]).$$

The *subset* predicate is implemented so that $subset(L_1, L_2)$ is true if and only if each element the list L_1 appears in L_2, and the relative order of the elements in L_1 is the same as the relative order of the elements in L_2; for example, the goal $subset([X,Y],[A,B,C])$ has the three solutions

$$subset([A,B],[A,B,C]).$$
$$subset([A,C],[A,B,C]).$$
$$subset([B,C],[A,B,C]).$$

Thus the rules above ensure that the variables X and Y will be distinct and in a canonical order in every ℓ-symbol $pred(X,Y)$ generated by the grammar. Next, the rules for the $pred(X,Y)$ are rewritten as follows:

$$pred(X,Y) \rightarrow [X = Y].$$
$$pred(X,Y) \rightarrow [\neg X = Y].$$
$$pred(X,Y) \rightarrow [adj(X,Y)].$$
$$pred(X,Y) \rightarrow [\neg adj(X,Y)].$$
$$pred(X,Y) \rightarrow [less\_than(X,Y)].$$
$$pred(X,Y) \rightarrow [\neg less\_than(X,Y)].$$
$$pred(X,Y) \rightarrow [less\_than(Y,X)].$$
$$pred(X,Y) \rightarrow [\neg less\_than(Y,X)].$$

Notice that for the symmetric predicates *equal* and *adj*, only one order of the arguments X and Y is allowed, whereas for the nonsymmetric predicate *less_than*, either order is allowed. The modified grammar now generates no feature predicates in which the arguments are the same, and only generates one possible ordering of the arguments to the *adj* and *equal* predicates.

4.2. Predicate combinability constraints

Another way to bias a learner is to constrain how terminal predicates can be combined; that is, to constrain the classes of predicates that can be conjoined together. For the *illegal* problem, many combinations of predicates are either vacuous or can be simplified to a single predicate. For example

$$X = Y \wedge less\_than(X,Y)$$

always fails, and

$$\neg less\_than(X,Y) \wedge \neg less\_than(Y,X) \equiv X = Y.$$

In fact, given this set of feature predicates, it is not hard to see that there are a very limited number of ways to relate two variables X and Y.

(1) One can relate them by one of the ordering conditions *less_than*, *equal*, or $\neg$*less_than*.
(2) One can relate them by one of the adjacency conditions *adj* or $\neg$*adj*.
(3) One can relate them with a conjunction of an ordering condition and an adjacency condition.

Any other conjunction of predicates involving X and Y is either vacuous or can be simplified to a relationship of the types described above.

To enforce this bias, a more substantial revision of the grammar must be made. First, we will rewrite the first rule of the grammar of Fig. 1 as follows:

$$body(illegal(A,B,C,D,E,F)) \rightarrow (rels(X,Y) \text{ where } subset([X,Y],[A,B,C,D,E,F])).$$

The "where" keyword here indicates another macro-expansion, parallel to the expansion of rules: the rule above actually expands to a single rule with a very long right-hand side:

$$body(illegal(A,B,C,D,E,F)) \rightarrow rels(A,B), rels(A,C), rels(A,D), \ldots, rels(E,F).$$

Again, use of the subset condition in the "where" goal means that every ℓ-symbol $rels(X,Y)$ will have its arguments in a canonical order, and that they will be distinct. We then rewrite the rules for *rels* to generate a conjunction of one ordering condition and one adjacency condition:

$$rels(X,Y) \rightarrow ordering(X,Y), adjacency(X,Y).$$

and finally define these new ℓ-nonterminals as follows:

$$ordering(X,Y) \rightarrow [X = Y].$$
$$ordering(X,Y) \rightarrow [less\_than(X,Y)].$$
$$ordering(X,Y) \rightarrow [less\_than(Y,X)].$$
$$ordering(X,Y) \rightarrow [\neg less\_than(X,Y)].$$
$$ordering(X,Y) \rightarrow [\neg less\_than(Y,X)].$$

$$adjacency(X,Y) \rightarrow [adj(X,Y)].$$
$$adjacency(X,Y) \rightarrow [\neg adj(X,Y)].$$

Notice that since the predicates for *ordering* and *adjacency* always succeed, either the ordering condition or the adjacency condition (or both) between two variables X and Y can be effectively dropped by simply retaining that of the nonterminal ℓ-symbols *ordering*/2 or *adjacency*/2; for example, the sentential forms

$$ordering(B,F), adj(B,F)$$

simplify to the clauses

$$illegal(A, B, C, D, E, F) \text{ :- } adj(B, F)$$
$$illegal(A, B, C, D, E, F) \text{ :- } A = C$$

respectively. Thus, while the two rules

$$ordering(X, Y) \rightarrow [\,]$$
$$adjacency(X, Y) \rightarrow [\,]$$

could be added to the grammar, they are not necessary.

4.3. Typing information

Another form of background knowledge is *typing information*. In the *illegal* problem, for example, the arguments of the predicate $illegal(A, B, C, D, E)$ are of two types: ranks and files. A natural constraint to impose is to insist that the predicates *equal*, *less_than* and *adj* are only used between compatible types: thus if R_1 and R_2 are variables denoting ranks and F is a variable denoting a file, the literal $R_1 = R_2$ could be used in a hypothesis, but the literal $R_1 = F$ could not.

Although it is not obvious, constraining the predicates in this way is actually an active bias. For example, if we impose the apparently reasonable constraint that the arguments of *equal* are of the same type, it is no longer possible to learn such concepts as

$$illegal(A, B, C, D, E) \text{ :- } C = D$$

which represents the condition "the rook is on the main diagonal". However, we will treat typing along with the other passive biases discussed above for the following reasons. First, *a priori* knowledge of the argument types is an extremely common occurrence in computer science generally. In particular, it is not hard to believe that typing knowledge would be as or more readily available than the sorts of knowledge of predicate semantics that enabled us to impose the passive biases of the previous sections; thus, if one rates types of background knowledge by how likely they are to be available, typing information biases are comparable to the passive biases discussed above. The second reason is methodological: we wish to compare our system to FOIL, and comparing it to typeless FOIL is unfair in that typing information is not only easy to add to a FOIL-type learner, but in fact has been added to the most current version of FOIL [30].

To add typing information in the *illegal* problem, it is sufficient to replace the rule

$$body(illegal(A, B, C, D, E, F)) \rightarrow$$
$$(rels(X, Y) \text{ where } subset([X, Y], [A, B, C, D, E, F])).$$

```
goal_formula(illegal(A,B,C,D,E,F)).

body(illegal(A,B,C,D,E,F)) →
  rels(A,C),rels(A,E),rels(C,E),   % relations between rank variables
  rels(B,D),rels(B,F),rels(D,F).   % relations between file variables

rels(X,Y) → ordering(X,Y),adjacency(X,Y).

ordering(X,Y) → [less_than(X,Y)].
ordering(X,Y) → [less_than(Y,X)].
ordering(X,Y) → [¬less_than(X,Y)].
ordering(X,Y) → [¬less_than(Y,X)].
ordering(X,Y) → [X = Y].
ordering(X,Y) → [¬X = Y].

adjacency(X,Y) → [adj(X,Y)].
adjacency(X,Y) → [¬adj(X,Y)].
```

Fig. 8. A second antecedent description grammar for the *illegal* problem.

with the rule

```
body(illegal(A,B,C,D,E,F)) →
  (rels(X,Y) where subset([X,Y],[A,C,E])),
  (rels(W,Z) where subset([W,Z],[B,D,F])).
```

This macro-expands to the rule

```
body(illegal(A,B,C,D,E,F)) →
  rels(A,C),rels(A,E),rels(C,E),
  rels(B,D),rels(B,F),rels(D,F).
```

Note that A, C and E are the rank variables, and B, D and F are the file variables. The remainder of the grammar is unchanged; a complete grammar incorporating all of the constraints discussed in the preceding sections is given in Fig. 8.

4.4. Experiments with passive bias and typing

To test the value of the passive biases and typing biases described above, we compared the performance of GRENDEL using the antecedent description grammar of Fig. 8 to FOIL. The implementation of FOIL we used in these experiments is an alternative instantiation of the EFOIL algorithm schema; thus GRENDEL and FOIL share a large amount of code, which facilitates CPU-time comparison of the two systems. We ran twenty trials: in each trial 100 king–rook–king positions were randomly selected, classified according to their legality, and then provided as training examples to both FOIL and GRENDEL. The resulting hypotheses were then tested against an independently

Table 1
Effect of passive biases on *illegal* problem.

Learner	Error (%)	Search	Time
FOIL	3.094	1347.4	208.5
Typed FOIL	1.986	656.4	127.7
Biased GRENDEL	1.336	314.2	45.7
Biased′ GRENDEL	1.336	339.5	56.7
Biased′ GRENDEL (2 new vars)	1.772	597.4	134.2
Unbiased GRENDEL	1.937	1408.7	229.8

chosen sample of 5000 king–rook–king positions for accuracy. Passively biased GRENDEL had an average error of 1.336%, while FOIL had an average error of 3.094%; the difference between the two is statistically significant ($t = 2.2381$, $p > 0.975$).[12] GRENDEL also requires much less time than FOIL, averaging 46 seconds versus 209 ($t = 29.2$, $p > 0.99$), and searches a much smaller space, checking an average of 314 clauses versus 1347 ($t = 16.4$, $p > 0.99$).[13]

Comparing GRENDEL to a typeless FOIL is somewhat unfair in that typing information is easy to add to a FOIL-type learner; in fact, typing has been added to the most current version of FOIL [30]. We also performed the same experiment on a version of FOIL that uses typing information. Typed FOIL is much faster than FOIL on this problem, but it is still more than twice as slow as GRENDEL (with $t = 37.6$, $p > 0.99$ for time measurements, and $t = 17.5$, $p > 0.99$ for search). In accuracy, typed FOIL also performed better ($t = 20.2$, $p > 0.99$) than typeless FOIL, obtaining an average error of 1.986%; however, GRENDEL still seems to be more accurate, although the difference in accuracy just misses the 95% confidence level required for statistical significance ($t = 2.027$, $0.900 < p < 0.950$).[14]

Table 1 summarizes these results. For comparison, we also summarize the performance of GRENDEL using several weaker biases. The row of the table labeled "biased′ GRENDEL" indicates the performance of GRENDEL us-

[12] Throughout, our measures of significance are based on using a 2-tailed t-test (or z-test, if there are enough datapoints) on the difference of the two quantities, testing against the null hypothesis that the average difference is zero. The value given for p is the confidence that the null hypothesis should be rejected.

[13] The search space is the number of clauses whose information gain is computed; CPU time is measured in CPU seconds on a SparcStation 1+. In our implementation, computing information gain dominates the time spent learning; however, since information gain of longer clauses is more difficult to compute, run-time is not always strictly linear in the search space. The substantial difference in CPU times between our implementation and Quinlan's is probably due largely to the difference in implementation languages: our implementation is written in Prolog while Quinlan's is written in C.

[14] The improved accuracy of GRENDEL over typed FOIL is somewhat surprising, as one would expect passive biases to reduce the time complexity of learning, but not to improve learning speed. The likely explanation of the difference is that while typed FOIL can introduce new variables in the antecedent, GRENDEL (with this antecedent description grammar) cannot; thus GRENDEL is searching a somewhat smaller space. The relative performance of unbiased GRENDEL and FOIL (below) also suggests that this difference can affect accuracy.

ing a grammar that encodes typing constraints and predicate-use constraints, but not predicate-combinability constraints; the reason for giving this result separately is that we conjecture that it is easy to automate adding typing and predicate-use constraints to a grammar, but that automatically adding predicate-combinability constraints would be difficult. Biased′ GRENDEL obtains the same error rates as passively biased GRENDEL, but searches more clauses ($t = 13.9$, $p > 0.99$) and requires more time ($t = 19.6$, $p > 0.99$).

The next row describes a still weaker bias, in which the grammar has been additionally modified to allow one new rank variable G and one new file variable H to be introduced in the antecedent of a rule; this is done by replacing the grammar rules

$$body(illegal(A,B,C,D,E,F)) \rightarrow rels(A,B,C,D,E,F)$$
$$rels(A,B,C,D,E,F) \rightarrow [\,]$$
$$rels(A,B,C,D,E,F) \rightarrow rel(A,B,C,D,E,F), rels(A,B,C,D,E,F)$$
$$rel(A,B,C,D,E,F) \rightarrow pred(X,Y)$$
$$\text{where } subset([X,Y],[A,C,E])$$
$$rel(A,B,C,D,E,F) \rightarrow pred(X,Y)$$
$$\text{where } subset([X,Y],[B,D,F])$$

with the grammar rules

$$body(illegal(A,B,C,D,E,F)) \rightarrow rels(A,B,C,D,E,F,G,H)$$
$$rels(A,B,C,D,E,F,G,H) \rightarrow [\,]$$
$$rels(A,B,C,D,E,F,G,H) \rightarrow$$
$$rel(A,B,C,D,E,F,G,H), rels(A,B,C,D,E,F,G,H)$$
$$rel(A,B,C,D,E,F,G,H) \rightarrow pred(X,Y)$$
$$\text{where } subset([X,Y],[A,C,E,G])$$
$$rel(A,B,C,D,E,F,G,H) \rightarrow pred(X,Y)$$
$$\text{where } subset([X,Y],[B,D,F,H]).$$

This version of GRENDEL seems to be somewhat less accurate than biased′ GRENDEL, although the difference is not statistically significant; it also searches more clauses ($t = 16.1$, $p > 0.99$) and requires more time ($t = 26.6$, $p > 0.99$). The main reason for considering this bias is that it demonstrates that a grammatical bias can allow new variables to be introduced in the antecedents of clauses; this issue will be returned to in Section 4.5.

Finally, the row labeled "unbiased GRENDEL" shows performance using the weak bias imposed by the grammar of Fig. 1. GRENDEL's performance with this weak bias is roughly comparable to untyped FOIL's performance: its accuracy is better ($t = 3.05$, $p > 0.99$) and its search and run-time are

somewhat (about 10%) worse.[15] The improvement in accuracy is due the fact that unbiased GRENDEL introduces no new variables in a clause body, while untyped FOIL may; thus unbiased GRENDEL is actually searching a somewhat smaller search space. This statistic shows that allowing new variables degrades accuracy in this domain; also, it is useful in evaluating the performance cost that is paid in switching from FOIL, a relatively inflexible learning system, to GRENDEL, a relatively flexible system, on problems where GRENDEL's additional flexibility is not being used.

4.4.1. Testing generality with a second domain

The experiments reported in Table 1 show that passive biases improve performance on the *illegal* problem. As a test of the generality of this technique, we have applied it to a number of additional learning problems.

Eleusis 1, *Eleusis 2*, and *Eleusis 3* are learning problems from the card game Eleusis. The object of the card game of Eleusis is to learn a secret rule which determines when a sequence of cards can be extended. In play, players attempt to add to the sequence. Each new card is placed to the right of the last card if it is a legal continuation, and underneath the last card otherwise; these new cards become positive and negative examples, respectively, of the secret rule. The three learning problems are based on three "layouts" which arose in human play, taken from [11]; for these problems, we used the feature predicates that were used in [28] to solve the same problems using the FOIL system. As it turns out, similar typing, symmetries, and predicate-combinability constraints can also be used in this domain.

Since the amount of data is small, we used the "leave one out" technique [39] to estimate error rates. For a layout with n cards, n runs of each learning system were made, where in each run, one of the n examples was withheld during training and used as a test case. The average error rates for these n runs was used as an estimate of the true error rate of the learning system's hypothesis. The run-times and search spaces reflect the performance of the learners given the full set of n examples.

The results of these experiments are shown in Table 2. For these examples, passively biased GRENDEL shows a slight superiority in accuracy, and requires substantially less time to learn than either typed or untyped FOIL. Because the sample sizes are small, these results are not statistically significant.

4.4.2. Testing generality with random problems

Another experiment investigating the effects of passive bias and typing was designed to test the thesis that the passive bias is indeed passive, and not some sort of subtle encoding of conditions present in these four specific learning problems. In this experiment, 50 target theories were generated randomly, 25 of them using the relations from the *illegal* problem, and 25 of them using

[15] The difference in run-time is statistically significant ($t = 4.7$, $p > 0.99$) but the difference in search space is not.

Table 2
Effect of passive biases on *Eleusis* problems

Problem	Learner	Error	Search	Time
Layout 1	FOIL	4/26	1198	59.7
	Typed FOIL	4/26	258	14.3
	Biased GRENDEL	4/26	167	9.5
Layout 2	FOIL	3/29	958	74.9
	Typed FOIL	3/29	206	20.2
	Biased GRENDEL	3/29	131	11.8
Layout 3	FOIL	2/29	958	54.3
	Typed FOIL	1/29	206	13.4
	Biased GRENDEL	1/29	132	9.5

Table 3
Effect of passive biases on random learning problems

	Average		
Learner	Error (%)	Search	Time
Typed FOIL	1.174	390.02	123.87
Biased GRENDEL	1.074	199.02	32.22

the relations from the *Eleusis* problems. Each theory contained between one and five clauses, and each clause contained between one and five conjuncts. The generated theories were nonrecursive and contained no variables in the antecedents of any clause that were not also in the body of the clause; also typing constraints were obeyed, and efforts were made to ensure that the random theories did not contain meaningless conditions. Otherwise, however, no constraints were placed on the generated theories. Training was done on a sample of size proportional to the complexity[16] of the target theory, and hypotheses were tested by testing against an independently chosen sample of 1000 examples.

Table 3 summarizes the results of this experiment. Using the passive bias, GRENDEL seems to give a slightly better generalizations than typed FOIL, although this difference is not statistically significant. However, it obtains these generalizations much more quickly than FOIL does, searching about half the space in about a quarter of the time. These results are statistically significant ($z = 2.27$, $p > 0.975$), indicating that biased GRENDEL really does outperform FOIL on problems randomly chosen from this distribution.

4.4.3. *A comparison to FOCL's predicate use constraints*

The FOCL learning system [25,26] also includes a mechanism for constraining FOIL (on which it is based) with information such as typing information, information about which predicates must have distinct arguments,

[16] In particular, we followed the methodology of Pagallo and Hassler [24] by generating n/ε examples for a target concept of description length n. For these experiments, ε was fixed at 0.3, and the number of examples generated ranged from 50 to 780.

Table 4
Comparison of passive biases to FOCL mechanisms

Learner	Search
Untyped FOIL (from [26])	10,366
FOCL (from [26])	711
Biased GRENDEL	585

and information about which predicates are symmetric. Antecedent description grammars provide a different means of doing the same thing. Our approach differs from Pazzani and Kibler's in two ways: first, it can be used to encode many other biases as well, and second, it can encode information (like predicate-combinability information) which FOCL cannot encode.

A final experiment with passive biases compares GRENDEL using typing and passive biases to FOCL. In order to do this, we will consider the variant of the *illegal* problem described in [26]. In this problem, the feature predicates are *equal*, *adj*, and *between*, where *between* is defined[17] as

$$between(X, Y, Z) \equiv (X < Y < Z) \vee (X > Y > Z).$$

A grammar for this problem was constructed which enforces predicate-combinability constraints and recognizes various redundancies of the types described above. All of the constraints described in [26] were incorporated into the grammar, as well as some additional constraints, such as the predicate combinability constraints and the symmetry

$$between(X, Y, Z) \equiv between(Z, Y, X)$$

which cannot be encoded in FOCL. (FOCL has no mechanism for limiting how predicates can be conjoined, and allows declaration of only a few types of symmetry.)

When given a random sample of 641 examples, FOCL's constraints reduced the search space from 10,366 clauses (for a version of untyped FOIL) to 711. GRENDEL's typing constraints and passive biases reduced the search space by an additional 18%, to 585 clauses, on the same random sample; these results are summarized in Table 4. Both GRENDEL and FOCL output 100% correct hypotheses. This suggests that GRENDEL with a passive-bias grammar will search a smaller space than FOCL with typing and predicate-use constraints (but without a domain theory); this is a direct consequence of the fact that GRENDEL's grammatical biases allow more types of knowledge about predicate usage to be used.[18]

[17] It should also be noted that Pazzani and Kibler use a slightly different definition of *adj* than Quinlan does (and hence, different from our previous version of the *illegal* problem). If ranks and files are encoded as integers between one and eight, Quinlan's definition of adjacency is $adj(X, Y) \equiv |X - Y| \leqslant 1$ whereas Pazzani and Kibler's definition is $adj(X, Y) \equiv |X - Y| = 1$.

[18] The author would like to thank Michael Pazzani for providing the data used in this experiment.

4.5. Relational clichés

Another type of knowledge which may be useful in learning is knowledge of common programming constructs in the concept description language. For GRENDEL, the concept description language is a subset of Prolog; hence knowledge of Prolog's "programming clichés" may be useful in learning. An example of such a programming cliché is the class of conjunctions of the form

$$p(X_1, \ldots, X_i, \ldots, X_k) \wedge comparator(X_i, n)$$

where X_i is a new (previously unbound) variable and n is a numeric constant, and *comparator* is some numeric comparison operator, like *greater_than* or *less_than*. A second example of a programming cliché is the class of conjunctions of the form

$$r(X_1, \ldots, X_i, \ldots, X_k) \wedge goal(\ldots, X_i, \ldots)$$

where r is a predicate that brings some argument closer to the base case (e.g., r finds the tail of a list, or the predecessor of a natural number) and then binds the new value to X_i, $goal(\ldots, X_i, \ldots)$ is a recursive call to the predicate *goal*, and all of the arguments except X_i in the recursive call to *goal* are the same as in the initial call to *goal*.

An extension to FOIL that takes advantage of programming clichés such as these is described in [34]. Silverstein and Pazzani call the first type of cliché a *threshold comparator cliché*, and the second type a *recursive cliché*; several other types of clichés are also described in [34]. For reasons discussed in [34], such clichés are necessary in learning some types of relational concepts; Silverstein and Pazzani thus call them *relational clichés*.

This type of background information can be encoded in an antecedent description grammar in a straightforward manner. For instance, to use recursive clichés, one first defines a nonterminal ℓ-symbol *rels* which expands to all possible strings of conditions that can be placed on the input arguments (an example of this is the definition of *rels* used in the grammar of Fig. 1.) One then defines a nonterminal ℓ-symbol *cliché* which expands to any of the possible instantiations of the recursive cliché; finally, one defines the start symbol as

$$body(goal(X_1, \ldots, X_n)) \rightarrow rels(X_1, \ldots, X_n), cliche(X_1, \ldots, X_n).$$

As an example, Fig. 9 gives the antecedent description grammar for learning the recursive predicate $list(X)$, which is true if X is a null-terminated list. We follow [28] in insisting that the hypotheses must use only the following predicates:

- $components(A, B, C)$: B and C are the head and tail of A,
- $null(A)$: A is the empty list.

$goal\_formula(list(X))$.

$body(list(X)) \rightarrow rel(X), cliche(X)$.
$rel(X) \rightarrow [null(X)]$.
$rel(X) \rightarrow [\neg null(X)]$.
$cliche(X) \rightarrow [components(X, Y, X_1), list(X_1)]$.
$cliche(X) \rightarrow [\,]$.

Fig. 9. Antecedent description grammar for learning *list*.

Table 5
Learning using recursive clichés

Problem	Learner	Correct?	Search	Time
List	Typed FOIL	yes	21	1.03
	GRENDEL	yes	11	0.47
Member	Typed FOIL	yes	49	2.77
	GRENDEL	yes	17	1.03

Also, recursive invocations of the goal predicate are allowed.[19] In this simple problem, there are only two relations that can be tested—X can be either null or nonnull—and only one possible recursive cliché.[20] To investigate the utility of the recursive cliché, as defined by Silverstein and Pazzani, we attempted to duplicate Quinlan's experiments in learning recursive concepts using FOIL [28]. Only two of the four predicates, *list* and *member*, can be represented by Silverstein and Pazzani's recursive cliché; this suggests, perhaps, that this definition of recursive cliché is overly specific. The data for the *list* learning task consists of all the possible sublists of the structure $[b, [a], d]$ as positive examples, and $[e|f]$ as the sole negative example; the data for the *member* task consists of all membership relationships over the list $[a, b, [c]]$ and its subcomponents. The results are summarized in Table 5. Both FOIL and GRENDEL were always able to learn the correct concepts from this data; however, GRENDEL learned them somewhat more quickly.

The threshold comparator cliché is perhaps a more interesting extension, as it allows GRENDEL to learn a class of concepts which FOIL cannot learn at all: concepts which require threshold tests. FOIL is unable to learn such concepts for two reasons. First, threshold tests are not encoded in function-free Horn clauses (as numerical constants are actually 0-ary function symbols).

[19] One problem in allowing recursive copies of the goal predicate is that theorem proving cannot be used to determine if this recursive call will succeed, since of course the definition of the goal predicate is unknown. Following Quinlan's FOIL system, we use the examples to determine if the goal predicate succeeds; the predicate $goal(X_1, \ldots, X_n)$ is considered to succeed if and only if there is a positive example $goal(X_1, \ldots, X_n)$.

[20] We are assuming the following type declarations: *null*(*listType*), *components*(*listType*, *objectType*, *listType*), *list*(*listType*), and *member*(*objectType*, *listType*). FOIL was given the same type declarations in the experiments.

goal_formula(*variety*(*Iris*)).

variety(*Iris*) → *cliches*(*Iris*).

cliches(*Iris*) → [ ].
cliches(*Iris*) → *cliche*(*Iris*), *cliches*(*Iris*).

cliche(*Iris*) → [*sepal_length*(*Iris*, *X*), *X* < *Th*] where *member*(*Th*, [20,...,44]).
cliche(*Iris*) → [*sepal_length*(*Iris*, *X*), *X* > *Th*] where *member*(*Th*, [20,...,44]).
cliche(*Iris*) → [*sepal_width*(*Iris*, *X*), *X* < *Th*] where *member*(*Th*, [43,...,79]).
cliche(*Iris*) → [*sepal_width*(*Iris*, *X*), *X* > *Th*] where *member*(*Th*, [43,...,79]).
cliche(*Iris*) → [*petal_length*(*Iris*, *X*), *X* < *Th*] where *member*(*Th*, [1,...,25]).
cliche(*Iris*) → [*petal_length*(*Iris*, *X*), *X* > *Th*] where *member*(*Th*, [1,...,25]).
cliche(*Iris*) → [*petal_width*(*Iris*, *X*), *X* < *Th*] where *member*(*Th*, [11,...,67]).
cliche(*Iris*) → [*petal_width*(*Iris*, *X*), *X* > *Th*] where *member*(*Th*, [11,...,67]).

Fig. 10. Antecedent description grammar for the iris problem.

More importantly, neither of the two predicates in the threshold cliché has any information gain taken individually; this means that FOIL's greedy search strategy will not be able to construct such a conjunction. It is only when the predicates are considered in pairs that the information gain metric can be used to make a sensible choice between them.

To verify that threshold clichés actually do enable GRENDEL to learn threshold concepts, we applied GRENDEL to the classic problem of learning to distinguish between various types of irises [14]. The grammar for this experiment, which is shown in Fig. 10, allows each rule to be a string of threshold tests on the four attributes sepal length, sepal width, petal length, and petal width. The possible threshold values listed in the grammar are all of the numeric values that actually appeared for that value in the training data. The dataset consists of 150 examples classified into one of three classes. A random sample of 44 examples was extracted to use as test data, and the remainder was used as training data. Since GRENDEL learns binary predicates, we used GRENDEL to learn each of the three classes in turn.

As FOIL cannot learn concepts that include numerical threshold tests, we compared GRENDEL's to Quinlan's C4.5 program [29] on the same partition. Although C4.5 can learn multiclass concepts, for the purpose of comparison, we used it in the same manner that GRENDEL was used: a definition for each class was learned in turn, using nonmembers of that class as negative examples.

The results of the experiment are summarized in Table 6. Using the threshold comparator cliché, GRENDEL does quite well in generating accurate hypotheses; this is somewhat surprising, given that GRENDEL has no mechanism for coping with noise. However, GRENDEL is orders of magnitude slower than C4.5. Some of this difference is due to differences in implementation language:

Table 6
Learning using threshold comparator clichés

Problem	Learner	Error	Search	Time
Setosa	C4.5	0/44	—	0.1
	C4.5 (pruning)	0/44	—	0.1
	GRENDEL	0/44	214	152.8
Versicolor	C4.5	4/44	—	0.1
	C4.5 (pruning)	3/44	—	0.1
	GRENDEL	3/44	1665	603.1
Viginica	C4.5	4/44	—	0.1
	C4.5 (pruning)	3/44	—	0.1
	GRENDEL	3/44	1873	623.3

C4.5 is written in C, while GRENDEL is written in Prolog. However, some of the difference is probably due to the overhead of using tuple-based measures of hypothesis quality rather than the simpler example-based measures of quality employed by C4.5.

We also note that many other learning techniques show good performance on this dataset [39, pp. 152–153]. As another aside, the experiments with the iris data are also interesting because they are a real-world dataset, whereas most of the experiments described in this paper deal with artificial learning problems.

To summarize the results of this section, although we have not compared GRENDEL's performance to the extension of FOIL which makes use of clichés, GRENDEL appears to be quite competent at making use of programming cliché knowledge. This competence has been demonstrated on three problems using two different types of clichés. GRENDEL's main advantage is that this knowledge is made use of by a uniform mechanism rather than a special-purpose one. However, there are some advantages to be gained by applying a special-purpose mechanism to this task; for example, Silverstein and Pazzani's system caches those instantiations of a cliché that were actually used in a learning problem, which may improve performance on later, similar, learning problems.

5. Results for learning from theory and data

The types of background knowledge discussed in the previous section are in one sense unusual; most recent research on using background knowledge in learning has focused on background knowledge that is represented as a logical theory [2, Part VII]. In this section, we will demonstrate that antecedent description grammars can be used to capture these types of background knowledge as well; in particular, we will show how several well-known algorithms that learn from theory and data can be emulated by GRENDEL.

$drinking\_vessel(X) \leftarrow stable(X), liftable(X), open\_vessel(X).$
$stable(X) \leftarrow bottom(X, B), flat(B).$
$open\_vessel(X) \leftarrow concavity(X, C), upward\_pointing(C).$
$liftable(X) \leftarrow graspable(X), light(X).$
$graspable(X) \leftarrow handle(X, H).$
$graspable(X) \leftarrow small\_width(X), insulating(X).$

Fig. 11. Theory for "drinking vessel".

5.1. Emulating IOU

The first knowledge-based learning system that we will use GRENDEL to emulate is *Induction Over the Unexplained* (*IOU*) [22]. The idea that underlies IOU is that sometimes we are interested in learning concepts such that some aspects of concept membership can be explained, but some aspects are conventional. For example, a shot glass has some features that are easy to explain (such as a flat bottom so that it is stable) and some that are conventional and difficult to explain (such as the fact that it holds 1.5 fluid ounces). IOU uses an overgeneral theory to learn such concepts; in this case, it might use a theory defining "drinking vessel" to learn various types of drinking vessels, such as a shot glass: such a theory, taken from [22], is shown in Fig. 11.

The IOU algorithm works as follows. First, negative examples that are not accepted by the theory are discarded (in this case, examples that are not drinking vessels). Second, features of the remaining examples that were used in the theory are discarded: in this example, all features except for *color*, *volume*, and *shape* are discarded. Finally, a standard inductive learning algorithm is invoked on the remaining examples, as described by the remaining features.

We will illustrate our emulation of IOU with the same example used to illustrate IOU. To encode the problem of learning "cup" from "drinking vessel", a very simple antecedent description grammar can be used. First, we constrain hypotheses to contain only conditions concerning the unexplained features *color*, *volume*, and *shape*, as well as the feature of being a drinking vessel.

$$body(cup(X)) \rightarrow [drinking\_vessel(X)], color\_cond(X), volume\_cond(X), shape\_cond(X).$$

Next, we define the new ℓ-nonterminals. For example, a *volume_cond* can expand to any of the possible volumes, or to the empty string.

$volume\_cond(X) \rightarrow [small(X)].$
$volume\_cond(X) \rightarrow [tiny(X)].$
$volume\_cond(X) \rightarrow [large(X)].$
$volume\_cond(X) \rightarrow [\,].$

The ℓ-symbols *color_cond* and *shape_cond* are defined analogously.

Consider the operation of GRENDEL using the antecedent description language. Since *drinking_vessel*(X) must appear in every clause of the hypothesis, examples that are not accepted by the *drinking_vessel* theory can never be included; thus, they are effectively discarded. Notice that IOU also discards these examples. The features used to separate the remaining positive examples from the remaining negative examples are precisely the features that IOU would use. Thus, this is a very close emulation of IOU. The only difference is that a different similarity-based learning algorithm is used for the final inductive learning step; we use a FOIL-like algorithm while Mooney and Ourston use ID3.

The hypothesis that GRENDEL finds using this grammar and the training data of [22] is

cup(A) :- *drinking_vessel*(A), *small*(A).

This is the same as the simplest description generated by IOU. For this small example, GRENDEL tests 13 clauses in 0.4 CPU seconds.

One difference between GRENDEL and IOU is that GRENDEL's grammar *explicitly states* the relation between the background theory for *drinking_vessel* and the target theory. IOU requires no such explicit statement; instead, the relationship between background theory and target theory is assumed by the algorithm.

Although we have not done so, it seems likely that the procedure for generating the antecedent description grammar for IOU-type theories can be automated for fairly broad classes of background theories. The difficult part of such a procedure would be analyzing the theory to see which features are "unexplained"; if nothing else, this could be done empirically, as in [22], by actually explaining the training examples with the theory.

5.2. *Emulating A-EBL*

A-EBL [9,6] is another knowledge-based learning system that uses an overgeneral theory: that is, a theory defining a concept that is a superset of the target concept. The intuition behind A-EBL is that sometimes we have background knowledge that is sufficient to allow us to generate "plausible explanations" as to why an object is a member of the target concept, but that is not sufficient to tell us which of the explanations are actually correct. A-EBL takes as input a theory expressing this sort of background knowledge and a set of training data, and tries to come up with a set of "good" explanations; these explanations are generalized using explanation-based generalization (EBG) [10,20] to form a generalization of the examples.

As an example of a problem to which A-EBL is suited, suppose that we have the following background knowledge about the task of bidding in the game of contract bridge.[21]

[21] Contract bridge is a card game played by two partnerships of two persons each. Bridge play is preceded by dealing a *hand* of 13 cards to each player and then conducting an *auction* in which partnerships compete for the right to name the trump suit; each *bid* in the auction is a number and a suit name. For the purposes of this example, the number can be assumed to always be equal to one.

- A suit is *biddable* if the hand contains 4 or 5 cards in that suit.
- If a hand contains two biddable suits, then there are two rules for choosing between them. In some circumstances, one chooses the *longer* suit of the hand, where the *length* of a suit is the number of cards of that suit the hand contains; in other circumstances, one chooses the *higher* suit of the hand, where higher refers to the following ordering: clubs (low), diamonds, hearts, spades (high).

With this background knowledge, it may be that given an example of the concept *correct_bid* one can construct several explanations justifying the correctness of the bid. For instance, given the positive example

+*correct_bid*(♠KQ984♡3◇AJ54♣KJ6, spade)

one can construct two explanations for why the bid is correct.

Explanation 1. Spades are biddable because there are five spades, diamonds are biddable because there are four diamonds, and spades is preferred because it is the higher suit.

Explanation 2. Spades are biddable because there are five spades, diamonds are biddable because there are four diamonds, and spades is preferred because it is the longer suit.

From the theory and this single example, there is no way of telling which of these explanations is the correct one. However, further examples can help in differentiating these explanations; for instance, the negative example

−*correct_bid*(♠KQ84♡3◇AJ954♣KJ6, diamond)

suggests that Explanation 2 was incorrect, since Explanation 2 could also be used to justify this incorrect bid. Additional positive examples might also give information, as they might give additional support to one of the two explanations.

The A-EBL algorithm works as follows. First, one constructs all possible explanations for the positive examples using the background theory, and generalizes these explanations using EBG. The result of this step is a (possibly large) set of candidate rules; for example, if the background knowledge above is codified into the theory for *plausible_bid* shown in Fig. 12, the example set

+*correct_bid*(♠KQ984♡3◇AJ54♣KJ6, spade)
−*correct_bid*(♠KQ84♡3◇AJ954♣KJ6, diamond)

would generate the two candidate rules

plausible_bid(*Hand*, *Suit1*) :-
  five_cards(*Suit1*) ∧ *four_cards*(*Suit2*) ∧ *higher*(*Suit1*, *Suit2*).
plausible_bid(*Hand*, *Suit1*) :-
  five_cards(*Suit1*) ∧ *four_cards*(*Suit2*) ∧ *longer*(*Suit1*, *Suit2*).

Overgeneral theory:

plausible_bid(*Hand*, *Suit*) :-
  biddable(*Hand*, *Suit1*) ∧ *biddable*(*Hand*, *Suit2*) ∧
  prefer(*Hand*, *Suit1*, *Suit2*).

biddable(*Hand*, *Suit1*) :- *four_cards*(*Hand*, *Suit*).
biddable(*Hand*, *Suit1*) :- *five_cards*(*Hand*, *Suit*).

prefer(*Hand*, *Suit1*, *Suit2*) :- *longer*(*Hand*, *Suit1*, *Suit2*).
prefer(*Hand*, *Suit1*, *Suit2*) :- *higher*(*Hand*, *Suit1*, *Suit2*).

The predicates *four_cards, five_cards, longer,* and *higher are* "operational"

Antecedent description grammar:

body(*correct_bid*(*Hand*, *Suit*)) → *plausible_bid*(*Hand*, *Suit*).

plausible_bid(*Hand*, *Suit*) →
  biddable(*Hand*, *Suit1*), *biddable*(*Hand*, *Suit2*), *prefer*(*Hand*, *Suit1*, *Suit2*).

biddable(*Hand*, *Suit1*) → [*four_cards*(*Hand*, *Suit*)].
biddable(*Hand*, *Suit1*) → [*five_cards*(*Hand*, *Suit*)].

prefer(*Hand*, *Suit1*, *Suit2*) → [*longer*(*Hand*, *Suit1*, *Suit2*)].
prefer(*Hand*, *Suit1*, *Suit2*) → [*higher*(*Hand*, *Suit1*, *Suit2*)].

Fig. 12. Theory and antecedent description grammar for a simple bridge problem.

A-EBL then filters this set of rules by testing them against the negative data; in this example, the second rule would be discarded. The final step is to use set cover techniques to come up with a relatively small set of the surviving rules that covers all the positive data; in this example, this consists of simply choosing the first rule. These rules are used as the definition of the target concept (in this case *correct_bid*); thus in this example, A-EBL would return the hypothesis

correct_bid(*Hand*, *Suit1*) :-
  five_cards(*Suit1*) ∧ *four_cards*(*Suit2*) ∧ *higher*(*Hand*, *Suit1*, *Suit2*).

In the introduction, we stated that there is a close connection between grammatically biased learning and the type of theory specialization performed by A-EBL; by virtue of this connection, antecedent description grammars can very easily encode A-EBL's hypothesis space. Like GRENDEL, A-EBL's hypothesis is a set of Horn clauses. For this theory, each such clause consists of three conditions: a "biddability condition" justifying the biddability of the first suit, which can be either a *four_cards* or a *five_cards* predicate; an analogous condition justifying the biddability of the second suit; and a "preference condition", which is expanded to either a *higher* or a *longer* predicate. An appropriate grammar is shown in Fig. 12; to emphasize the similarity between it and the

theory used by A-EBL, I have introduced a terminal called *plausible_bid* that expands to all of the partial operationalizations of the *plausible_bid* predicate, and I have used the name *biddable* for the ℓ-nonterminal that expands to a "biddability condition" and *prefer* for the ℓ-nonterminal that expands to a "preference condition".

The relationship between A-EBL's overgeneral theory and the GRENDEL grammar is thus extremely close: in fact, except for simple syntactic changes, they are identical. In fact, the A-EBL theory is precisely the theory $Th_{\mathcal{G}}$ which defines the semantics of the nonterminal symbols of the grammar $\mathcal{G}$. This correspondence is no accident; it is a result of the correspondence between grammar-rule rewrites and Horn clause theorem proving discussed in Section 2.6. In particular, this correspondence means that *a sentential form of a grammar $\mathcal{G}$ is identical to a partial operationalization of $Th_{\mathcal{G}}$, and that a sentence of $\mathcal{G}$ is identical to a rule formable from $Th_{\mathcal{G}}$ by applying EBG to an example.* This observation leads to the following approach to emulating A-EBL with GRENDEL: given an A-EBL learning problem with a background theory T_0, construct a grammar $\mathcal{G}_0$ such that $Th_{\mathcal{G}_0} = T_0$, and use this grammar as the input to GRENDEL.

One of the problems to which A-EBL has been applied is a more realistic version of the bridge bidding problem described in the example above [6]. The background knowledge for this problem is a medium-sized (124-clause) theory defining the concept "plausible bid" which was manually extracted from a textbook on playing contract bridge [33]; the goal of learning is to specialize this theory to a definition of the concept "correct bid". The training data consists of 45 sample hands, also presented in the textbook, from which were taken 46 correct and 16 incorrect bids. The test data consists of 16 hands taken from a self-test in the same book. As in [6], a hypothesis was judged to be correct on a test problem if it did not suggest any incorrect bids, and suggested at least one correct bid.

A-EBL applied to this problem returns, in 105.4 CPU seconds, a hypothesis that is correct on 14 of the 16 test cases. ANA-EBL [6], a variant of A-EBL that tests some nonoperational clauses as well, can improve the accuracy to 15 of 16 test cases; in doing so run-time is degraded to 229 CPU seconds, with the k parameter set to $k = 1$, or to 949.8 CPU seconds, with the k parameter set to $k = 2$. Given the same inputs, GRENDEL returns a hypothesis that is perfect (16 of 16 correct) on the test cases; however, GRENDEL requires more time than any of the variants of A-EBL in finding this hypothesis, testing 1365 clauses in 3193.2 CPU seconds. GRENDEL's longer run-time results from the fact that GRENDEL searches a larger hypothesis space than A-EBL: GRENDEL's hypotheses also include partial operationalizations of the initial theory as well as full operationalizations. Testing these partial operationalizations is also more expensive than testing fully-operationalized rules (as A-EBL does) or rules that are almost fully operationalized (as ANA-EBL does).

As a further test of GRENDEL's ability to emulate A-EBL, we also measured GRENDEL's performance on randomly generated examples of the *correct_bid*

Table 7
Comparison of GRENDEL and A-EBL

Problem	Learner	Error	Search	Time
fixed dataset	A-EBL	2/16	—	105.4
	ANA-EBL ($k = 1$)	1/16	—	229.0
	ANA-EBL ($k = 2$)	1/16	—	949.8
	GRENDEL	0/16	1365	3193.2
60 random	A-EBL	14.3	—	—
	ANA-EBL ($k = 1$)	9.3	—	—
	GRENDEL	12.6	434.4	925.0
120 random	A-EBL	8.9	—	—
	ANA-EBL(k=1)	6.4	—	—
	GRENDEL	9.6	757.0	2654.5

concept. We used the example generator described in [5] to generate random training sets of 60 and 120 examples, and then tested the accuracy of the hypotheses that GRENDEL produced against an independently generated set of 1000 test cases. These experiments were repeated 10 times and the results were averaged. The results of these experiments are shown in Table 7. They show that GRENDEL may not actually generalize more rapidly than A-EBL, as the fixed data suggests; on the random data, its accuracy is statistically indistinguishable from A-EBL, and statistically significantly worse (with $t = 2.3$, $p > 0.95$ for the 60-example case and $t = 2.7163$, $p > 0.95$ for the 120-example case) than ANA-EBL with the k parameter set to 1. (Raising the k parameter of ANA-EBL to $k = 2$ yields a slight but statistically significant improvement relative to $k = 1$ for this dataset [6].) These results are summarized in Table 7.

To summarize, GRENDEL can be used to emulate A-EBL, and constructing a grammar for the emulation can be done automatically, using a simple syntactic transformation of the domain theory; this simple transformation has in fact been automated. Using the grammar, GRENDEL searches a strictly larger space than A-EBL or any of its variants. One advantage of GRENDEL over A-EBL is that it does not explicitly enumerate all of the proofs for an example; hence GRENDEL, unlike A-EBL, does not require that the number of proofs for a example be bounded. (For example, A-EBL could not be used to specialize the theory of Fig. 3.) An advantage of A-EBL is that there are mathematical guarantees of its performance [9]; the experiments indicate that it is also more efficient in time and sample complexity than GRENDEL.

The similarities between GRENDEL and A-EBL are perhaps more interesting than the differences: although the learning algorithms themselves are quite different, A-EBL's brand of theory specialization is, in an important sense, equivalent to the grammatically biased learning performed by GRENDEL. A grammatical bias is thus revealed to be simply a new metaphor for the old idea of theory specialization. This issue will be raised later in Section 6.4, when we discuss the relative advantages of the two metaphors; from a practical point of view, however, the question of which metaphor is more appropriate

1) $illegal(A,B,C,D,E,F)$:- $same\_location(A,B,C,D)$.
2) $illegal(A,B,C,D,E,F)$:- $same\_location(A,B,E,F)$.
3) $illegal(A,B,C,D,E,F)$:- $same\_location(C,D,E,F)$.
4) $illegal(A,B,C,D,E,F)$:- $king\_attacks\_king(A,B,E,F)$.
5) $illegal(A,B,C,D,E,F)$:- $rook\_attacks\_king(A,B,C,D,E,F)$.

6) $same\_location(W,X,Y,Z)$:- $W = Y \wedge X = Z$.
7) $king\_attacks\_king(A,B,E,F)$:- $adj(A,E) \wedge adj(B,F)$.

8) $rook\_attacks\_king(A,B,C,D,E,F)$:-
   $C = E \wedge king\_not\_between\_file(A,B,C,D,E,F)$.
9) $rook\_attacks\_king(A,B,C,D,E,F)$:-
   $D = F \wedge king\_not\_between\_rank(A,B,C,D,E,F)$.

10) $king\_not\_between\_file(A,B,C,D,E,F)$:- $\neg A = C$.
11) $king\_not\_between\_file(A,B,C,D,E,F)$:- $A = C \wedge \neg between(D,B,F)$.

12) $king\_not\_between\_rank(A,B,C,D,E,F)$:- $\neg B = D$.
13) $king\_not\_between\_rank(A,B,C,D,E,F)$:- $B = D \wedge \neg between(C,A,E)$.

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

Fig. 13. A complete theory for the *illegal* problem.

can be avoided by allowing the user to make use of either. In the current implementation of GRENDEL clauses of the form A :- B are automatically converted to grammar rules $A \rightarrow B$ if they are declared by the user to be part of the "domain theory". Thus GRENDEL supports either a clause-like or grammar-rule like syntax for its background knowledge, or a mix of clauses and grammar rules; for example, GRENDEL would accept either of the two formats shown in Fig. 12.

5.3. Learning from an incomplete theory

Both A-EBL and IOU work by specializing theories that are overgeneral: that is, theories that define supersets of the target concept. We turn our attention now to a second type of theory: a theory that contains some, but not all of the clauses of the target theory. Such a theory will be called an *incomplete theory.*

For example, consider the theory of Fig. 13. This theory is a complete and correct definition of the *illegal* predicate used in previous examples; if we remove any of the clauses of this theory, the result will be an incomplete theory for the goal concept *illegal.* The motivation for studying this problem is that there may be situations in which some subconcepts used in a defining a concept are either unknown or imperfectly known.

We will consider two ways for a theory to be incomplete: it can be missing predicates "from the top" or "from the bottom". An example of the first type

of theory would the theory of Fig. 13 with clauses 1–5 deleted: such a theory contains complete definitions of some of the predicates that are part of the goal predicate, but does not actually contain a definition of the goal predicate. An example of a theory that is incomplete "at the bottom" is obtained by deleting clauses 10–15 of the theory of Fig. 13: such a theory contains a correct definition of *illegal* in terms of lower-level predicates, but does not include definitions of all of the lower-level predicates.

Learning using an incomplete theory that is missing predicates "from the top" is actually no different from the learning task attacked by FOIL: in both cases, one is attempting to construct a definition of the goal predicate using a fixed set of feature predicates. Thus this knowledge-based learning problem can be solved by applying standard inductive learning techniques to a larger set of feature predicates.[22]

The more interesting case is an incomplete theory that is missing predicates "from the bottom". How can GRENDEL make use of such an incomplete theory? As in constructing the emulation of A-EBL, we will consider an incomplete theory, determine the constraints that incomplete theory places on the clauses of the target theory, and finally express these constraints in an antecedent description grammar.

The incomplete theory that we will consider as an example is the theory that contains clauses 1–9 of the theory of Fig. 13. The first four clauses of this theory contain only completely-defined predicates; thus these clauses should be included in GRENDEL's hypothesis. One way to encourage[23] GRENDEL to do this is to include the following rules in the antecedent description grammar:

$$body(illegal(A,B,C,D,E,F)) \rightarrow [same\_location(A,B,C,D)].$$
$$body(illegal(A,B,C,D,E,F)) \rightarrow [same\_location(A,B,E,F)].$$
$$body(illegal(A,B,C,D,E,F)) \rightarrow [same\_location(C,D,E,F)].$$
$$body(illegal(A,B,C,D,E,F)) \rightarrow [king\_attacks\_king(A,B,E,F)].$$

Since *same_location* and *king_attacks_king* are now ℓ-terminals, clauses 6 and 7 of the incomplete theory are treated in the same way as the clauses defining the other feature predicates like *less_than* and *adj*; they are included in the theory Th_G, but otherwise have no impact on the grammar.

Clause 5 is the more interesting case. We cannot simply introduce the grammar rule

$$body(illegal(A,B,C,D,E,F)) \rightarrow [rook\_attacks\_king(A,B,C,D,E,F)].$$

[22] Quinlan's implementation of FOIL imposes special restrictions on the feature predicates, which make it impractical to make predicates such as *rook_attacks_king* available in learning. These restrictions can, however, be dropped, albeit at some cost in efficiency; our implementation places no restrictions on the feature predicates.

[23] Notice that GRENDEL is encouraged to include these clauses, but is not forced; if there are no examples for which these clauses are useful, they will not be included in the final hypothesis.

because the definition of *rook_attacks_king* is unknown. Instead, however, we can introduce a new nonterminal ℓ-symbol *rook_attacks_king*, which will expand to all possible definitions of *rook_attacks_king*, and add the following grammar rule to the theory:

$$body(illegal(A,B,C,D,E,F)) \rightarrow rook\_attacks\_king(A,B,C,D,E,F).$$

We now need to construct rules so that the ℓ-nonterminal *rook_attacks_king* behaves appropriately; that is, so it expands to all possible definitions of the predicate *rook_attacks_king*. Clauses 8 and 9 give rise to the following two grammar rules, which describe two ways in which the *rook_attacks_king* nonterminal can be expanded.

$$rook\_attacks\_king(A,B,C,D,E,F) \rightarrow$$
$$[C = E], king\_not\_between\_file(A,B,C,D,E,F).$$
$$rook\_attacks\_king(A,B,C,D,E,F) \rightarrow$$
$$[D = F], king\_not\_between\_rank(A,B,C,D,E,F).$$

Again, *king_not_between_file* and *king_not_between_rank* are new nonterminals, which will expand into all possible definitions of these predicates. In this case, however, the definitions of these predicates are completely unknown: thus we will use a series of relatively weak grammar rules to describe the possible definitions of these predicates. For example, we can make use of the techniques described in Section 4 to define these ℓ-nonterminals as follows:

$$king\_not\_between\_file(A,B,C,D,E,F) \rightarrow$$
$$(rels(X,Y) \text{ where } subset([X,Y],[A,C,E])),$$
$$(rels(W,Z) \text{ where } subset([W,Z],[B,D,F])).$$
$$king\_not\_between\_rank(A,B,C,D,E,F) \rightarrow$$
$$(rels(X,Y) \text{ where } subset([X,Y],[A,C,E])),$$
$$(rels(W,Z) \text{ where } subset([W,Z],[B,D,F])).$$

The expansions of $rels(X,Y)$ can then be defined using the appropriate grammar rules from Fig. 8. The complete grammar for this incomplete theory is shown in Fig. 14.

To evaluate the ability of GRENDEL to learn using an incomplete theory, we deleted each of the predicates *same_location*, *king_attacks_king*, and *rook_attacks_king* from the complete theory. Grammars for each of these incomplete theories were then constructed, and we ran GRENDEL, using these grammars, on twenty different samples containing 100 examples each. The results of these experiments are shown in Table 8. For comparison, we also give again the performance of typed FOIL and GRENDEL with a passive bias on this same problem; from the table it can be seen that incomplete theories improve both the accuracy of GRENDEL's hypotheses and GRENDEL's runtime. All of the differences are statistically significant except for the differences

$body(illegal(A,B,C,D,E,F)) \rightarrow [same\_location(A,B,C,D)].$
$body(illegal(A,B,C,D,E,F)) \rightarrow [same\_location(A,B,E,F)].$
$body(illegal(A,B,C,D,E,F)) \rightarrow [same\_location(C,D,E,F)].$
$body(illegal(A,B,C,D,E,F)) \rightarrow [king\_attacks\_king(A,B,E,F)].$
$body(illegal(A,B,C,D,E,F)) \rightarrow rook\_attacks\_king(A,B,C,D,E,F).$

$rook\_attacks\_king(A,B,C,D,E,F) \rightarrow$
$[C = E], king\_not\_between\_file(A,B,C,D,E,F).$
$rook\_attacks\_king(A,B,C,D,E,F) \rightarrow$
$[D = F], king\_not\_between\_rank(A,B,C,D,E,F).$

$king\_not\_between\_file(A,B,C,D,E,F) \rightarrow$
$(rels(X,Y)$ where $subset([X,Y],[A,C,E])),$
$(rels(W,Z)$ where $subset([W,Z],[B,D,F])).$
$king\_not\_between\_rank(A,B,C,D,E,F) \rightarrow$
$(rels(X,Y)$ where $subset([X,Y],[A,C,E])),$
$(rels(W,Z)$where$subset([W,Z],[B,D,F])).$

$rels(A,B) \rightarrow ordering(A,B), adjacency(A,B).$
$ordering(X,Y) \rightarrow [less\_than(X,Y)].$
$ordering(X,Y) \rightarrow [less\_than(Y,X)].$
$ordering(X,Y) \rightarrow [\neg less\_than(X,Y)].$
$ordering(X,Y) \rightarrow [\neg less\_than(Y,X)].$
$ordering(X,Y) \rightarrow [X = Y].$
$ordering(X,Y) \rightarrow [\neg X = Y].$
$adjacency(X,Y) \rightarrow [adj(X,Y)].$
$adjacency(X,Y) \rightarrow [\neg adj(X,Y)].$

Fig. 14. Antecedent description grammar derived from clauses 1–9.

Table 8
Effect of incomplete theories on the *illegal* problem

Deleted predicate	Error (%)	Search	Time
same_location	0.650	180.7	35.4
king_attacks_king	0.265	216.6	39.4
rook_attacks_king	1.046	253.3	46.4
Biased GRENDEL	1.336	314.2	45.7
Typed FOIL	1.986	656.4	127.7

in error rates between row 3 (the theory with *rook_attacks_king* deleted) and row 4 (biased GRENDEL).

It is beyond the scope of this paper to provide a comprehensive survey of the large body of research in learning from incomplete theories. Much of the work in logically-grounded abduction [27,31] can be viewed as completing an incomplete theory; additionally, there has been a great deal of work in machine learning on this problem—[13,17,37,40] are some examples. GRENDEL's main contribution to this area is perhaps the use of the information gain

heuristic to guide the search for an appropriate completion. The FOCL system [25], which will be described in the next section, also uses this heuristic, but in a rather different way: in computing the heuristic, FOCL counts the number of examples that fall into a *known* completion of a theory, while GRENDEL (used in the manner described in this section) counts the number of examples that fall into *any possible* completion of the theory. Further research is necessary to determine how these two different approaches compare.

5.4. Learning from a syntactically approximate theory

A final class of background theories that we will consider are theories that are syntactically close to the target theory: in other words, theories that can be transformed to the target theory with a small number of syntactic changes such as adding a clause, deleting a clause, adding conditions to the antecedent of an existing clause, or deleting a condition from the antecedent of an existing clause. In this paper, such theories will be called *syntactically approximate theories.* An example of a theory that is syntactically approximate is shown in Fig. 15, along with the theory that it approximates.[24] This syntactically approximate theory was derived from the correct theory by making the following syntactic changes.

- The extraneous condition $adj(B, F)$ was added to the antecedent of clause 1.
- The condition $adj(B, F)$ was deleted from the antecedent of clause 2.
- The clause $king\_attacks\_king(A, B, E, F)$:- $knight\_move(A, B, E, F)$ was added.
- Clause 9 was deleted.

The modified portions of the theory are underlined in the figure.

This section will concentrate on comparing GRENDEL to Pazzani and Kibler's FOCL system, which is one of several systems which learns from syntactically approximate theories [23,16,26]; FOCL is the most appropriate comparison because like GRENDEL, it can also learn relational concepts. The example of this section is taken from [26].

5.4.1. How FOCL works

FOCL is another system derived from FOIL. FOCL is "almost equivalent" to another instantiation of EFOIL, in which a clause may be refined in one of the following ways.

- The clause *Goal* :- *true* can be refined to *Goal* :- $body_i$, if *Goal* :- $body_i$ is a clause in the syntactically approximate theory.
- The clause

[24] Notice that the correct theory is a little different from the theory of Fig. 13; in particular, we are using Pazzani's set of operational predicates rather than Quinlan's (see Section 4.4.3). Also, to avoid confusion, we have changed the name of the main concept defined by the approximate theory from *illegal* to *illegal1*.

Correct Theory

1) $illegal(A,B,C,D,E,F)$:- $same\_location(A,B,C,D)$.
2) $illegal(A,B,C,D,E,F)$:- $same\_location(A,B,E,F)$.
3) $illegal(A,B,C,D,E,F)$:- $same\_location(C,D,E,F)$.
4) $illegal(A,B,C,D,E,F)$:- $king\_attacks\_king(A,B,E,F)$.
5) $illegal(A,B,C,D,E,F)$:- $rook\_attacks\_king(A,B,C,D,E,F)$.
6) $king\_attacks\_king(A,B,E,F)$:- $adj(A,E) \wedge adj(B,F)$.
7) $king\_attacks\_king(A,B,E,F)$:- $adj(A,E) \wedge B = F$.
8) $king\_attacks\_king(A,B,E,F)$:- $A = E \wedge adj(B,F)$.
9) $rook\_attacks\_king(A,B,C,D,E,F)$:-
   $C = E \wedge king\_not\_between\_file(A,B,C,D,E,F)$.
10) $rook\_attacks\_king(A,B,C,D,E,F)$:-
   $D = F \wedge king\_not\_between\_rank(A,B,C,D,E,F)$.
11) $king\_not\_between\_file(A,B,C,D,E,F)$:- $\neg A = C$.
12) $king\_not\_between\_file(A,B,C,D,E,F)$:- $A = C \wedge \neg between(D,B,F)$.
13) $king\_not\_between\_rank(A,B,C,D,E,F)$:- $\neg B = D$.
14) $king\_not\_between\_rank(A,B,C,D,E,F)$:- $B = D \wedge \neg between(C,A,E)$.

Syntactically Approximate Theory

1) $illegal1(A,B,C,D,E,F)$:-
   $same\_location(A,B,C,D) \wedge adj(B,F)$. % condition added
2) $illegal1(A,B,C,D,E,F)$:- $same\_location(A,B,E,F)$.
3) $illegal1(A,B,C,D,E,F)$:- $same\_location(C,D,E,F)$.
4) $illegal1(A,B,C,D,E,F)$:- $king\_attacks\_king(A,B,E,F)$.
5) $illegal1(A,B,C,D,E,F)$:- $rook\_attacks\_king(A,B,C,D,E,F)$.
6) $king\_attacks\_king(A,B,E,F)$:- $adj(A,E) \wedge adj(B,F)$.
7) $king\_attacks\_king(A,B,E,F)$:- $adj(A,E) \wedge B = F$.
8) $king\_attacks\_king(A,B,E,F)$:- $A = E$. % condition $adj(B,F)$ deleted
   $king\_attacks\_king(A,B,E,F)$:- $knight\_move(A,B,E,F)$. % clause added
9) %$rook\_attacks\_king(A,B,C,D,E,F)$:- $C = E \wedge \cdots$—clause deleted
10) $rook\_attacks\_king(A,B,C,D,E,F)$:-
   $D = F \wedge king\_not\_between\_rank(A,B,C,D,E,F)$.
11) $king\_not\_between\_file(A,B,C,D,E,F)$:- $\neg A = C$.
12) $king\_not\_between\_file(A,B,C,D,E,F)$:- $A = C \wedge \neg between(D,B,F)$.
13) $king\_not\_between\_rank(A,B,C,D,E,F)$:- $\neg B = D$.
14) $king\_not\_between\_rank(A,B,C,D,E,F)$:- $B = D \wedge \neg between(C,A,E)$.

Fig. 15. A syntactically approximate theory for the *illegal* problem.

$$Goal :- A_1 \wedge \cdots \wedge A_{i-1} \wedge A_i \wedge A_{i+1} \wedge \cdots \wedge A_k$$

can be refined to

$$Goal :- A_1 \wedge \cdots \wedge A_{i-1} \wedge body_j \wedge A_{i+1} \wedge \cdots \wedge A_k$$

if A_i :- $body_i$ is a clause in the syntactically approximate theory.

- The clause $Goal$:- $body_i$ can be refined to $Goal$:- $body_i \wedge L_k$, where L_k is a feature predicate or a predicate from the syntactically approximate theory. This is the same refinement operation used by FOIL.

There are two reasons that FOCL is only "almost equivalent", and not actually equivalent, to this instantiation of EFOIL. First, FOCL will continue to refine a consistent clause even if it contains nonoperational predicates (i.e., predicates other than feature predicates) while in EFOIL, a clause is refined only until it is consistent.[25] Second, FOCL does not refine clauses in the same way that EFOIL would; clauses are constructed using a procedure that shows a preference to the first and second types of refinements over the third type.

The effect of this is that FOCL forms as a hypothesis a Horn theory that contains clauses formed in one of the following three ways. First, FOCL forms clauses by *operationalizing the initial theory.* For example, the approximate theory for *illegal1* can be operationalized as follows, using first clause 5, then clause 10, and finally clause 14:

$$illegal1(A,B,C,D,E,F) \text{ :- } rook\_attacks\_king(A,B,C,D,E,F).$$
$$illegal1(A,B,C,D,E,F) \text{ :-}$$
$$D = F \wedge king\_not\_between\_rank(A,B,C,D,E,F).$$
$$illegal1(A,B,C,D,E,F) \text{ :- } D = F \wedge B = D \wedge \neg between(C,A,E).$$

Notice that even though the theory for *illegal1* is incorrect (relative to the correct definition of *illegal*), the final operationalized clause *is* correct, since it is derived by chaining together only rules that are correct.

Second, FOCL forms clauses by *adding literals to a rule that is an operationalization* of the initial theory, using FOIL's heuristics for choosing these literals. To illustrate the benefits of doing this, notice that one can also operationalize *illegal1* as follows, using clause 4 and then clause 8:

$$illegal1(A,B,C,D,E,F) \text{ :- } king\_attacks\_king(A,B,E,F).$$
$$illegal1(A,B,C,D,E,F) \text{ :- } A = E.$$

Since clause 8 is missing the condition $adj(B,F)$, the operationalized rule is also missing this condition. However, the resulting operationalized rule can be corrected by adding a single additional condition; this correction can be found using FOIL.

Finally, FOCL forms rules by *building rules from scratch* using the FOIL algorithm. For example, given a large enough sample, FOCL might reconstruct the deleted clause (clause 9) using this approach. The final type of bug in the approximate theory, the added clause, can be corrected because FOCL includes in its hypothesis only some operationalizations of the initial approximate theory.

We emphasize that FOCL's search process is guided only partially by Quinlan's information gain heuristic; partially, the search is guided by predetermined

[25] I.e., covers no negative examples.

preferences between the three subspaces described above. In particular, FOCL prefers using an operationalization of the initial theory to using an operationalization that needs to be corrected by appending one or more additional literals, and FOCL prefers using a corrected operationalization to building a rule from scratch. For more details, see [26].

This description of FOCL is actually incomplete, as FOCL has been improved since these experiments were begun [25]. The most recent version of FOCL can also drop a condition from a rule which has been constructed by partially operationalizing the initial theory. The reordering of the searched space imposed by the addition of this operator will not be discussed in this section, nor will it be addressed the next section, which describes an emulation of Pazzani and Kibler's original implementation of FOCL.

5.4.2. Emulating FOCL with GRENDEL

To describe how to emulate FOCL using GRENDEL, we will again apply the methodology of considering the space of clauses searched by FOCL, and then constructing a grammar that generates only the clauses from that space. The first type of clause FOCL can generate are clauses that are operationalizations of the approximate theory. We saw in Section 5.2 how, given a Horn clause theory T (such as the theory for *illegal1*) one can automatically construct a grammar that generates only clauses that are partial operationalizations of theory. Thus, the first part of FOCL's search space can be easily captured with the grammar rule

$$body(illegal(A,B,C,D,E,F)) \rightarrow illegal1(A,B,C,D,E,F)$$

where *illegal1* is a ℓ-nonterminal that expands to the possible operationalizations of the syntactically approximate theory for *illegal1*. The third part of the search space—the space of all clauses built from scratch using FOIL-like means—is also quite easy to encode; Fig. 8, also an earlier example, shows a grammar which generates all meaningful conjunctions of predicates for this domain.[26] Letting $rels(A,B,C,D,E,F)$ be a ℓ-nonterminal that expands to all strings in such a grammar, we can encode the third part of the search space bias as follows.

$$body(illegal(A,B,C,D,E,F)) \rightarrow rels(A,B,C,D,E,F).$$

Finally, the second part of the search space—the space of all clauses formed by constructing an operationalization of *illegal1* and then appending a series of additional conditions—can be encoded using the following rule.

$$body(illegal(A,B,C,D,E,F)) \rightarrow$$
$$\quad illegal1(A,B,C,D,E,F), rels(A,B,C,D,E,F).$$

[26] Again, however, we will use a grammar that is a little different from the grammar of Fig. 8, since for the purpose of comparison we are using Pazzani's set of feature predicates rather than Quinlan's (see Section 4.4.3).

$body(illegal(A,B,C,D,E,F)) \rightarrow$
$\quad illegal1(A,B,C,D,E,F), rels(A,B,C,D,E,F).$
$body(illegal(A,B,C,D,E,F)) \rightarrow rels(A,B,C,D,E,F).$

% $rels(A,B,C,D,E,F)$ expands to all meaningful conjunctions of predicates in this domain
$rels(A,B,C,D,E,F) \rightarrow$
$\quad rels(A,C), rels(A,E), rels(C,E), rels(A,C,E),$
$\quad rels(B,D), rels(B,F), rels(D,F), rels(B,D,F).$
$rels(X,Y,Z) \rightarrow [between(T,U,V)]$ where $select(U,[X,Y,Z],[T,V])$.
$rels(X,Y,Z) \rightarrow [\neg between(T,U,V)]$ where $select(U,[X,Y,Z],[T,V])$.
$rels(X,Y) \rightarrow [X = Y].$
$rels(X,Y) \rightarrow [\neg X = Y].$
$rels(X,Y) \rightarrow [adj(X,Y)].$
$rels(X,Y) \rightarrow [\neg adj(X,Y)].$

% grammar for the syntactically approximate theory of Fig. 15
$illegal1(A,B,C,D,E,F) \rightarrow [same\_location(A,B,C,D), adj(B,F)].$
$illegal1(A,B,C,D,E,F) \rightarrow [same\_location(A,B,E,F)].$
$illegal1(A,B,C,D,E,F) \rightarrow [same\_location(C,D,E,F)].$
$illegal1(A,B,C,D,E,F) \rightarrow king\_attacks\_king(A,B,E,F).$
$illegal1(A,B,C,D,E,F) \rightarrow rook\_attacks\_king(A,B,C,D,E,F).$
$king\_attacks\_king(A,B,E,F) \rightarrow [adj(A,E), adj(B,F)].$
$king\_attacks\_king(A,B,E,F) \rightarrow [adj(A,E), B = F].$
$king\_attacks\_king(A,B,E,F) \rightarrow [A = E].$
$king\_attacks\_king(A,B,E,F) \rightarrow [knight\_move(A,B,E,F)].$
$rook\_attacks\_king(A,B,C,D,E,F) \rightarrow$
$\quad [D = F], king\_not\_between\_rank(A,B,C,D,E,F).$
$king\_not\_between\_file(A,B,C,D,E,F) \rightarrow [\neg A = C].$
$king\_not\_between\_file(A,B,C,D,E,F) \rightarrow [A = C, \neg between(D,B,F)].$
$king\_not\_between\_rank(A,B,C,D,E,F) \rightarrow [\neg B = D].$
$king\_not\_between\_rank(A,B,C,D,E,F) \rightarrow [B = D, \neg between(C,A,E)].$

Fig. 16. Antecedent description grammar for emulating FOCL.

Notice that since $rels(A,B,C,D,E,F)$ can be expanded to the empty string, this grammar rule also generates all of the clauses generated by the rule

$$body(illegal(A,B,C,D,E,F)) \rightarrow illegal1(A,B,C,D,E,F).$$

Since this rule is now redundant, it can be dropped.

A complete grammar encoding FOCL's hypothesis space for this learning problem is shown in Fig. 16. The rules giving the expansion of the *illegal1* ℓ-nonterminal are derived from the syntactically approximate theory for *illegal1*; the rules giving the expansion of the *rels* ℓ-nonterminal are an encoding of typing constraints and passive biases for this domain, and are taken from the grammar used in Section 4.4.3.

Table 9
Using GRENDEL with a syntactically approximate theory

Grammatical Bias	Error (%)	Search	Time
FOCL bias	5.08	637.8	141.4
FOCL bias + control	1.83	264.9	36.5
Passive bias	2.82	345.0	50.0

5.4.3. Grammatical bias is not enough

Unfortunately, GRENDEL's performance using this grammar is markedly inferior to its performance using simply a passive bias; in other words, GRENDEL does not make good use of the approximate theory. In 20 trials using 100 randomly selected board positions, GRENDEL with the FOCL-emulation grammar of Fig. 16 took an average of almost three times as long to search almost twice the space, and generated hypotheses that are only half as accurate as GRENDEL using a simple passive bias; the results of this experiment are shown in Table 9. This happens even though the space searched by GRENDEL, using its FOCL-emulation grammar, is almost exactly the same as the space searched by FOCL.

Closer analysis suggests two reasons why this grammatical bias degrades rather than improves performance. One reason more time is spent in learning is that the search space defined by the FOCL-emulation grammar is extremely *redundant*. For example, every operational clause generated by the rule

$$body(illegal(A,B,C,D,E,F)) \rightarrow illegal1(A,B,C,D,E,F), rels(A,B,C,D,E,F)$$

is also generated by the rule

$$body(illegal(A,B,C,D,E,F)) \rightarrow rels(A,B,C,D,E,F).$$

Another problem is that the search space is *unordered*. Intuitively, clauses that are formed by operationalizing the *illegal1* theory are more likely to be accurate than clauses built up from scratch by expanding the *rels* ℓ-nonterminal, and hence should be preferred in constructing the hypothesis. However, GRENDEL imposes no such preference. In contrast, FOCL imposes a strong preference for clauses that are formed by operationalizing the approximate theory. One would expect that adding such a preference would lead to more accurate hypotheses; it might also improve learning time, as the parts of the search space most likely to contain useful clauses are searched first.

The remarks above suggest that grammatical bias is not enough: in order to make effective use of a syntactically approximate theory, it is necessary to impose some *preferential ordering* on the clauses considered by the learner, so that the learner is encouraged to use clauses which are syntactically close to clauses implied by the theory. This ordering is a second type of background knowledge. Notice that it cannot be expressed as an antecedent description grammar, since an antecedent description grammar by its very nature imposes

a rigid constraint on the hypothesis space, rather than an ordering of the hypothesis space.

5.4.4. Adding control knowledge to GRENDEL

Of course, use of a grammatical bias does not preclude the possibility of using a preferential bias as well; in fact, it is often relatively straightforward to extended a learning system to obey an ordering constraint. It is reasonable to hypothesize that such an extension to GRENDEL would allow it to make use of a syntactically correct domain theory.

To test this hypothesis, we added a simple preference mechanism to GRENDEL, in the form of a *control directive*. If A is a nonterminal ℓ-symbol that has N arguments, the user can give GRENDEL the directive *avoid_expanding* A/N. Given this directive, GRENDEL will apply the following procedure when selecting a refinement of a clause. First, the information gains of those designated refinements that were derived without expanding any ℓ-nonterminal which should be avoided are computed. If the best clause in this set has a positive information gain, it will be selected; otherwise, the information gains of the remaining clauses are computed, and the clause with the largest overall information gain is selected.

A control directive means that clauses derived by expanding the avoided ℓ-nonterminal will be considered by GRENDEL only if all other clauses have negative (or zero) information gain; thus a strong preference will be shown to clauses which do not contain expansions of the avoided ℓ-nonterminal. Control directives impose a preferential ordering similar to that imposed by FOCL; one important difference is that while FOCL's hypothesis space is implicitly ordered by its learning algorithm, GRENDEL's hypothesis space is explicitly ordered by the control directives.

When GRENDEL is given the control directive "*avoid_expanding rels /6*", and is used with the grammar of Fig. 16, the error rate of the average hypothesis improves to 1.8%; this error rate represents a statistically significant ($t = 2.38$, $p > 0.975$) improvement from the 2.8% error rate obtained using the passive bias grammar. This indicates that this simple extension is sufficient to allow GRENDEL to make effective use of a syntactically approximate theory and improve the accuracy of its hypotheses. Using the control directive also leads to a statistically significant reduction in time ($t = 5.88$, $p > 0.99$) and search space ($t = 4.04$, $p > 0.99$).

5.4.5. Comparison to FOCL

To summarize, a grammatical bias alone proved to be inadequate to allow learning from a syntactically approximate theory. However, by extending GRENDEL to accept control directives from the user, we are able to emulate the technique used in FOCL for learning from a syntactically approximate domain theory. The results we obtain are qualitatively the same as those obtained by FOCL: addition of a syntactically approximate theory causes an improvement in both learning time and in the accuracy of hypotheses. How-

ever, GRENDEL's emulation of FOCL differs from FOCL itself in several ways.[27]

FOCL and GRENDEL's emulation of FOCL both use a mix of empirical and analytic learning methods; however, GRENDEL's empirical method, learning from a passively-biased grammar, was shown in Section 4.4.3 be superior to typed FOIL, the empirical learning method used by FOCL. Another difference is that, unlike FOCL, GRENDEL may output a hypothesis that contains partial operationalizations of the approximate domain theory, as well as complete operationalizations.

Again, we emphasize that the major difference between FOCL and GRENDEL is that FOCL is less adaptable, in that it will always consider a fixed class of corrections to the theory. In contrast, GRENDEL can be directed to consider a smaller class of corrections, as our emulations of A-EBL and IOU illustrate. This could lead to improved performance in situations in which information is given about the kinds of errors present in the background theory. A consequence of greater generality is that GRENDEL's knowledge must be represented in a more explicit manner. In particular, GRENDEL's antecedent description grammar makes explicit the class of errors that can be corrected, and the control directives make explicit the preferential ordering of clauses. However, as in the emulations of IOU and A-EBL, it seems likely that the task of constructing a grammar (and control directives) for emulating FOCL on a problem could be automated, given a grammar encoding a passive bias for that problem and a syntactically approximate theory.

6. Related work

6.1. Previous work in grammatically biased learning

The previous sections have explored a number of possible applications of grammatically biased learning. While the applications are new, the general idea of guiding induction by making some portion of the concept description language explicit is not new. This idea was perhaps most clearly articulated in Mitchell's LEX system [21], in which a version space approach was used to construct hypotheses that were sentential forms in a given grammar.

Our high-level aims are much the same as Mitchell's; however the approach taken in GRENDEL incorporates several technical innovations that greatly extend the range of uses of grammatical bias. First, antecedent description grammars are computationally much more powerful than the context-free grammars used in LEX. This additional power not only makes it possible to learn relational concepts, but also makes it possible to use grammatical constraints to encode nonpropositional background theories and constraints on

[27] Because of these differences, a more direct comparison of FOCL to GRENDEL's emulation of FOCL is difficult; this may be the subject of future work.

variable use (such as typing constraints.) Second, Mitchell's learning technique imposes several additional restrictions on the grammar which are not imposed by the EFOIL algorithm. For example, in order to prevent the size of the S boundary set from growing too large, the number of possible parses of any single example must be constrained. Thus in LEX every example has a unique parse; in contrast, the grammar of Fig. 1 allows an infinite number of parses of any example. The additional power of our grammars and the relaxation of the constraints imposed by the version-space learning method are crucial in being able to unify the several different knowledge-based learning problems discussed in this paper. Finally, while Mitchell's technique requires a recognizer to be built for each nonterminal symbol, our approach does not.

6.2. *Quinlan's FOIL system*

GRENDEL's learning algorithm is derived from Quinlan's FOIL algorithm [28]. The main difference between GRENDEL and FOIL is that while FOIL uses a fixed procedure for generating clauses, GRENDEL's generation of clauses is guided by a grammar, which can be modified. Much of Section 4 is devoted to comparing a knowledge-free GRENDEL to an implementation of FOIL; the experiments in this section indicate that cases exist in which GRENDEL's performance is superior to FOIL's, even if little domain knowledge is available. Quinlan's implementation of FOIL, however, is far more efficient than GRENDEL, and includes several useful extensions which are not included in the current implementation of GRENDEL, including features that prune the search for new literals, deal with noisy data, postprocess the clauses of the hypothesis, and infer partial orderings among the clauses so that recursive definitions can be learned. Another advantage of FOIL over GRENDEL is that we have found it difficult to write grammars that introduce new variables in a clause, except in constrained ways. GRENDEL is thus limited in its ability to learn important classes of concepts such as recursive predicates.

6.3. *FOCL and other work in theory refinement*

A second system which is quite similar to GRENDEL is FOCL [26,25]. FOCL's learning algorithm is also based on FOIL's learning algorithm, and FOCL also can make use of many kinds of background knowledge. In particular, FOCL and GRENDEL both use typing information, information about predicate symmetries, relational clichés, and syntactically approximate theories; since the class of syntactically approximate theories subsumes the class of overgeneral theories and incomplete theories, FOCL can make use of essentially all of the types of background knowledge discussed in this paper. The main difference is that in FOCL, each of these types of background knowledge are handled by different mechanisms, while in GRENDEL they are all handled by grammatical bias. (There are also some more specific differences in how

GRENDEL and FOCL handle these types of background knowledge, which were discussed in the body of the paper.)

A major advantage of the FOCL approach is that it seems to lead to simpler and more declarative ways of presenting many types of background knowledge: for example, typing information is declaratively stated in FOCL, but not in GRENDEL. However, GRENDEL's approach also has several advantages. Grammatical bias can be used to encode some sorts of knowledge that FOCL cannot incorporate, such as predicate-combinability constraints. GRENDEL also allows the bias of the learning system to be controlled more precisely by the user; for example, GRENDEL can perform precisely the theory corrections performed by IOU and no others. While FOCL could be used with an IOU-type theory, it will consider a large space of theory corrections, and (presumably) would require more examples to attain the same degree of accuracy. Another advantage of GRENDEL's approach is that using a uniform learning mechanism holds the potential for closer integration of the various learning techniques. Finally, we would claim that a final advantage of GRENDEL's approach is the elegance of having a uniform technique for making use of background knowledge. This, however, is a rather subjective claim, and even if it is accepted, the benefits are hard to quantify.

Several other learning systems exist that, like FOCL, address the problem of learning from syntactically approximate theories. Most of these systems [16,23,36] are restricted to propositional theories only, and hence are concerned with only some of the types of background knowledge discussed in this paper; for example, none of these systems considers typing information or constraints on predicate usage. Some systems that can learn Horn clause theories are described in [1,32,41]. Like FOCL, the biases of these systems cannot be controlled to the degree that GRENDEL's can; they are specifically intended to learn from syntactically approximate theories. We have not yet systematically compared these systems to GRENDEL.

6.4. A-EBL and other work in theory specialization

6.4.1. Theory specialization versus grammatical bias

In the introduction, we stated that there was a very close relationship between the types of overgeneral theories used as background knowledge by A-EBL [9] and antecedent description grammars; that relationship was later clarified, and illustrated by an extended example, in Section 5.2. In fact, it is fair to say that theory specialization and grammatically biased learning are simply two different computational metaphors for the same basic task.

It can reasonably be argued that the metaphor of grammatically biased learning is in itself a technical advance, although this point is hard to establish in any quantitative way. One problem with the metaphor of theory specialization is that while many learning tasks can be recast as theory specialization problems [8] the theories which must be constructed are extremely artificial, *when viewed as theories.* For instance, consider trying to understand the theory of

Fig. 3 by considering only its semantics as a theory. While the corresponding grammar has a natural interpretation—it describes roughly the set of clauses which can be produced by FOIL—in the theory, every predicate can easily be shown to be always true. In fact, most of the grammars presented in this paper would be similarly vacuous, or would at least have vacuous portions, if they were converted to theories.

The problem is that the appropriateness or inappropriateness of a theory for theory specialization has little to do with its declarative meaning, and very much to do with its syntactic form; however, the theory metaphor strongly encourages one to focus on the declarative meaning of the theory. The grammar metaphor, on the other hand, encourages one to begin thinking in terms of derivable strings and derivations, which are precisely the properties which are important for guiding learning. Thus the grammar corresponding to an overgeneral theory tends to be more comprehensible to a user.

Perhaps the most telling argument in favor of the grammatical metaphor is the fact that many of the encodings described in this paper became evident to us only after the notion of grammatically biased learning had been established. In contrast, although theory specialization algorithms of almost equivalent power have been available for some time, the full range of applications of these techniques has not been explored.

6.4.2. Other differences

GRENDEL also differs from A-EBL and its variants in several more concrete ways. A-EBL requires that all proofs of each positive example be enumerable, and hence is not able to specialize theories such as the one of Fig. 3, which generates an infinite number of proofs for each goal. Several of the other example grammars in this paper correspond to theories which generate an extremely large number of proofs for each example; while A-EBL could in principle be used on such problems, in practice it would be inefficient. Also, all of the variants of A-EBL search only a part of the space of partial operationalizations of the initial theory, while GRENDEL searches the entire space. GRENDEL's search is also asymptotically more efficient than the search used by A-EBL (however, the experiments of Section 5.2 suggest that this may not always result in an improvement in practice.) Also, unlike A-EBL, GRENDEL is able to accept some control knowledge, as described in Section 5.4.4. This ability is crucial in using grammatical bias to learn from a syntactically approximate theory.

In performing theory specialization, GRENDEL's learning technique is actually more similar to FOCL's operation when it is given a theory that is strictly overgeneral. In particular, both GRENDEL and FOCL base their search on information gain, and both search in a general-to-specific order. The major difference is that GRENDEL uses a larger set of "designated refinements" in conducting its search; roughly speaking, FOCL uses the set of designated refinements

$$\textit{Designated\_Refinements}(\alpha) \equiv \{\beta : \alpha \Rightarrow \beta\}$$

introduced as a strawman in Section 3.2.2, while GRENDEL uses a much larger set. Thus, for the reasons discussed in that section, FOCL would be unable to correctly specialize theories like the one of Fig. 3. This is a fairly simple extension, but one which greatly expands the range of applicability of the learning technique: without it, most of the grammatically biased learning tasks described in this paper cannot be performed. GRENDEL is also not required to fully operationalize the theory, as FOCL does, although it may do so if the examples require it.

Finally, GRENDEL's technique of simplifying rules leads to an notable improvement in efficiency on many of learning problems described in this paper. This technique is not used in either A-EBL and its variants or in FOCL.

7. Conclusion

To summarize, this paper has described a learning system that makes a large part of the concept description language an explicit input to the learner. In particular, we have described GRENDEL, an extension of FOIL which learns a set of Horn clauses, each of which is generated by an *antecedent description language* provided by the user. In the paper, we have argued (by example) that many types of background knowledge which until now could be used only by special-purpose techniques can be naturally encoded as grammatical biases. In particular, we have shown that grammatically biased learning can encode the following sorts of knowledge in a way useful to a learner:

- Passive biases, such as typing constraints, constraints on predicate use, and predicate combinability constraints. This knowledge has been previously used in the FOCL system [26].
- Programming clichés, in particular the recursive clichés and the numerical threshold clichés described in [34].
- Two varieties of overgeneral theories: the kind required by the A-EBL algorithm [6] and the kind required by the IOU algorithm [22].
- Incomplete theories, which contain some but not all of the clauses of the target theory. Theories such as these are assumed by many systems, for example [13,17,37,40].
- Syntactically approximate theories, such as are used by theory revision systems like those described in [16,23,25,26,36].

The major advantage of GRENDEL's grammatically-biased learning techniques over the special-purpose techniques is their generality: all of the types of knowledge listed above can used by a single mechanism.

We view these results as promising for several reasons. First, they suggest that many of the problems studied in the field of knowledge-based learning can be unified under the general problem of grammatically-biased learning. This unification may lead to transfer of methods for one subproblem to

another, and to more direct integration of existing techniques. It also provides a readily formalizable model for these problems, which may facilitate theoretical analysis; the existence of several formal results on the closely related problem of theory specialization [5] is encouraging.

The algorithm embedded in the GRENDEL program does, however, have several major limitations. Foremost among these limitations is the fact that GRENDEL has no mechanisms for dealing with noisy data; relaxing this limitation is essential for dealing with real-world learning problems. Recent results in extending FOIL-like learners to learn from noisy data [4,12] may be useful in this regard.

Another disadvantage of GRENDEL is that in many cases, the information present in an antecedent description grammar can be represented in a simpler and more declarative manner: typing information is a good example of this. This problem could perhaps be addressed by the development of methods for automatically compiling declarative constraints (like typing constraints) into an antecedent description grammar. The examples of Sections 4 and 5 suggest that many interesting classes of background knowledge can be quite easily compiled into grammars; however, some of the constraints described in this paper, such as predicate-combinability constraints, seem to be quite difficult to incorporate automatically.

Yet another difficulty with the current implementation is that the grammars needed for certain problems are quite large. From a user's point of view, this is not problematic, given the macro-expansion facilities that we have developed; however, the current approach may well be extremely memory-intensive for some problems. In particular, when GRENDEL is used with a FOIL-like bias, its grammar is exponential in the arity of the feature predicates.[28] This problem can perhaps be addressed by allowing dynamic rather than static expansion of macro-rules.

While the algorithm used in GRENDEL seems to be successful on a wide variety of problems, it is quite difficult to analyze formally; upper bounds on GRENDEL's sample complexity would help to increase our confidence in its performance. However, this is probably a difficult task, as it is likely that GRENDEL only works on a limited class of distributions; obtaining this result would require (at least) characterizing the class of distributions on which GRENDEL is competent.

Finally, as Quinlan has pointed out in [28], the concept description language of Horn clause logic is perhaps unnecessarily restrictive; it may be possible to extend the power of the language somewhat without giving up the two advantages of efficiency and perspicuity. The learning techniques described in [7,38] for extensions of decision trees may be useful in this regard.

[28] Of course, FOIL's run-time is exponential in the arity of the feature predicates, so the asymptotic complexity of the two algorithms is the same. However, it is often more practical to use an algorithm which has long run-time than one which both has long run-time and requires very large data structures; memory limitations usually impose a less flexible constraint.

Appendix A. Proofs of theorems

Theorem 2.1. *Let $\mathcal{G}$ be a clause description grammar, let body(G) be the associated start ℓ-symbol, let α be a sentential form derived from body(G), and let $Th_{\mathcal{G}}$ be the Horn theory created using the construction above. Then*

$$p(t_1,\ldots,t_n) \in ext(\alpha) \quad \text{iff} \quad p(t_1,\ldots,t_n) \in ext(G :\!- translate_s(\alpha)).$$

Proof. We want to show that $A \in ext(\alpha)$ iff $A \in ext(G \leftarrow \alpha')$. In the proof, we will use α' to denote the translation of α to a conjunction, Th_T for the part of $Th_{\mathcal{G}}$ that contains definitions of the terminal-symbol predicates, and Th_N for the constructed part of $Th_{\mathcal{G}}$. For the purpose of this proof the definition of the *true* predicate is considered to be in Th_T; thus no complete proofs can be constructed in Th_N.

For the "only if" direction, consider

$$\begin{aligned} & A \in ext(\alpha) \\ \iff & A \in \bigcup_{\beta \in L(\alpha)} ext(G \leftarrow \beta') && \text{(A.1)} \\ \iff & \exists \beta\colon \beta \in L(\alpha) \wedge A \in (G \leftarrow \beta'). && \text{(A.2)} \end{aligned}$$

However, because of the isomorphism between resolution and grammar rule rewrites

$$\beta \in L(\alpha) \iff \text{there is an SLD resolution of } \alpha' \text{ to } \beta' \text{ in } Th_N. \qquad \text{(A.3)}$$

Also, by definition

$$A \in (G \leftarrow \beta') \iff \text{the mgu } \theta \text{ of } A \text{ and } G \text{ exists } \wedge\, Th_T \vdash \beta'\theta. \qquad \text{(A.4)}$$

Note that (A.3) implies that $\alpha'\theta$ can be reduced to $\beta'\theta$ in Th_N, by the rule of substitution; thus (A.3) and (A.4) together imply that (a) an mgu θ of A and G exists, (b) inference rules from Th_N reduce $\alpha'\theta$ to $\beta'\theta$, and (c) that $\beta'\theta$ is provable using rules in Th_T; thus $A \in ext(G \leftarrow \beta')$.

To show the "if" direction , assume that $A \in ext(G \leftarrow \alpha')$. Then by definition, both of the following hold:

$$A \text{ and } G \text{ have an mgu } \theta. \qquad \text{(A.5)}$$

$$Th_N \cup Th_T \vdash \alpha'\theta. \qquad \text{(A.6)}$$

Because the nonterminal and terminal symbols of the grammar are disjoint, the proof of $\alpha'\theta$ can be reordered so that all of the resolutions using rules from Th_N occur first; since, however, no complete resolution proofs can be made using only rules from Th_N, some rules from Th_T must be used. To summarize, the proof of $\alpha'\theta$ must consist of an initial reduction of $\alpha'\theta$ to some conjunction γ using rules from Th_N, and then a proof of γ using rules from Th_T.

Now, consider applying the same inference rules used to reduce $\alpha'\theta$ to γ to the goal α'. Certainly this is possible; further, by the isomorphism between rewriting and resolution, we will derive some goal β' to that $\beta \in L(\alpha)$.

Our final claim is that $\gamma = \beta'\theta$; is this is true, then we have established that there is a β so that (a) $\beta \in L(\alpha)$ (from the argument above) (b) there is an mgu of A and G (from (A.5)) and (c) that $\beta'\theta = \gamma$ is provable in $Th_{\mathcal{G}}$; note that (b) and (c) imply that $A \in ext(G \leftarrow \beta'\theta)$ and hence (a), (b), and (c) imply (A.2), which is equivalent to our desired result.

So it remains to establish that $\gamma = \beta'\theta$. A relatively simple intuitive justification is to note that β' and γ are both substitutional instances of some more general conjunction, call it δ. In particular, let σ represent the substitutions imposed by rules used in the SLD derivation from α' to β', then

$$\beta' = \delta\sigma.$$
$$\gamma = \delta(\sigma \cup \theta).$$

(If desired this argument can be made rigorous by using induction on the length of the SLD derivation.) It follows trivially from this that $\beta'\theta = \delta\sigma\theta = \gamma$. □

References

[1] F. Bergadano and A. Giordana, Guiding induction with domain theories, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* **3** (Morgan Kaufmann, San Mateo, CA, 1990) chapter 17, pages 474–492.

[2] L. Birnbaum and G. Collins, *Machine Learning: Proceedings of the Eighth International Workshop* (Morgan Kaufmann, San Mateo, CA, 1991).

[3] I. Bratko, S. Muggleton, and A. Varsek, Learning qualitative models of dynamic systems, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).

[4] C. Brunk and M. Pazzani, Noise-tolerant relational concept learning algorithms, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).

[5] W.W. Cohen, Concept learning using explanation based generalization as an abstraction mechanism, Ph.D. Thesis, Tech. Report DCS-TR-271, Rutgers University, New Brunswick, NJ (1990).

[6] W.W. Cohen, Learning from textbook knowledge: a case study, in: *Proceedings AAAI-90*, Boston, MA (1990).

[7] W.W. Cohen, A decision tree approach to theory specialization, Internal Technical Memorandum, AT&T Bell Laboratories (1991). Available from the author on request.

[8] W.W. Cohen, The generality of overgenerality, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).

[9] W.W. Cohen, Abductive explanation based learning: a solution to the multiple inconsistent explanation problem, *Mach. Learn.* **8** (2) (1992).

[10] G.F. DeJong and R. Mooney, Explanation-based learning: an alternative view, *Mach. Learn.* **1** (2) (1986).

[11] T.G. Dietterich and R.S. Michalski, Learning to predict sequences, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* **2** (Morgan Kaufmann, San Mateo, CA, 1986).

[12] S. Džeroski and N. Lavrac, Learning relations from noisy examples, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).

[13] T. Fawcett, Learning from plausible explanations, in: *Proceedings Sixth International Workshop on Machine Learning*, Ithaca, NY (1989).

[14] R.A. Fisher, The use of multiple measurements in taxonomic problems, *Annual Eugenics* **7** (1936) 179–188; also in: *Contributions to Mathematical Statistics* (Wiley, New York, 1950).

[15] N. Flann and T.G. Dietterich, A study of explanation-based methods for inductive learning, *Mach. Learn.* **4** (2) (1989).

[16] A. Ginsberg, Theory reduction, theory revision, and retranslation, in: *Proceedings AAAI-90*, Boston, MA (1990).
[17] R.V. Hall, Learning by failing to explain: using partial explanation to learn in incomplete or intractible domains, *Mach. Learn.* **3** (1) (1988).
[18] R.S. Michalski, I. Mozetic, J. Hong and N. Lavrac, The multipurpose incremental learning system AQ15 and its application to three medical domains, in: *Proceedings AAAI-86*, Philadelphia, PA (1986).
[19] T.M. Mitchell, Generalization as search, in: *Readings in Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA); also: *Artif. Intell.* **18** (1982) 203–226.
[20] T.M. Mitchell, R.M. Keller, and S. Kedar-Cabelli, Explanation-based generalization: a unifying view, *Mach. Learn.* **1** (1) (1986).
[21] T.M. Mitchell, P. Utgoff and R. Banerji, Learning by experimentation: acquiring and refining problem-solving heuristics, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, San Mateo, CA, 1983).
[22] R.J. Mooney and D. Ourston, Induction over the unexplained, in: *Proceedings Sixth International Workshop on Machine Learning*, Ithaca, NY (1989).
[23] D. Ourston and R.J. Mooney, Changing the rules: a comprehensive approach to theory refinement, in: *Proceedings AAAI-90*, Boston, MA (1990).
[24] G. Pagallo and D. Hassler, Boolean feature discovery in empirical learning, *Mach. Learn.* **5** (1) (1990).
[25] M. Pazzani, C. Brunk and G. Silverstein, A knowledge-intensive approach to learning relational concepts, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).
[26] M. Pazzani and D. Kibler, The utility of knowledge in inductive learning, *Mach. Learn.* **9** (1) (1992).
[27] D. Poole, A logical framework for default reasoning, *Artif. Intell.* **36** (1988) 27–47.
[28] J.R. Quinlan, Learning logical definitions from relations, *Mach. Learn.* **5** (3) (1990).
[29] J.R. Quinlan, Probabilistic decision trees, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* **3** (Morgan Kaufmann, San Mateo, CA, 1990) chapter 17, pages 140–152.
[30] J.R. Quinlan, Personal communication (1991).
[31] R. Reiter, A theory of diagnosis from first principles, *Artif. Intell.* **32** (1987) 57–95.
[32] B. Richards and R.J. Mooney, First-order theory revision, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).
[33] A. Sheinwold, *5 Weeks to Winning Bridge* (Simon and Schuster, New York, 1964).
[34] G. Silverstein and M. Pazzani, Relational clichés: constraining constructive induction during relational learning, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).
[35] L. Sterling and E. Shapiro, *The Art of Prolog: Advanced Programming Techniques* (MIT Press, Cambridge, MA, 1986).
[36] G. Towell, J. Shavlik and M. Noordewier, Refinement of approximate domain theories by knowledge-based artificial neural networks, in: *Proceedings AAAI-90*, Boston, MA (1990).
[37] K. VanLehn, Learning one subprocedure per lesson, *Artif. Intell.* **31** (1987) 1–40.
[38] L. Watanabe and L. Rendell, Learning structural decision trees from examples, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).
[39] S. Weiss and C. Kulikowski, *Computer Systems that Learn* (Morgan Kaufmann, San Mateo, CA, 1990).
[40] B. Whitehall and S.C.-Y. Lu, A study of how domain knowledge improves knowledge-based learning systems, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).
[41] J. Wogulis, Revising relational domain theories, in: *Proceedings Eighth International Workshop on Machine Learning*, Ithaca, NY (1991).