Artificial Intelligence 66 (1994) 273–309
Elsevier

ARTINT 1071

Theory refinement combining analytical and empirical methods

Dirk Ourston

Science Applications International Corporation, 3045 Technology Parkway, Orlando, FL 32826-3299, USA

Raymond J. Mooney

Computer Sciences Department, University of Texas, Austin, TX 78712, USA

Received December 1991
Revised January 1993

Abstract

Ourston, D. and R.J. Mooney, Theory refinement combining analytical and empirical methods, Artificial Intelligence 66 (1994) 273–309.

This article describes a comprehensive system for automatic theory (knowledge base) refinement. The system applies to classification tasks employing a propositional Horn-clause domain theory. Given an imperfect domain theory and a set of training examples, the approach uses partial and incorrect proofs to identify potentially faulty rules. For each faulty rule, subsets of examples are used to inductively generate a correction. Because the system starts with an approximate domain theory, fewer training examples are generally required to attain a given level of classification accuracy compared to a purely empirical learning system. The system has been tested in two previously explored application domains: recognizing important classes of DNA sequences and diagnosing diseased soybean plants.

1. Introduction

One of the most difficult problems in the development of intelligent systems is the construction of the underlying knowledge base. As a result, the rate of progress in developing intelligent systems is directly related to the speed with which knowledge bases can be assembled. Research in machine learning attempts to solve the *knowledge acquisition problem* by developing

Correspondence to: D. Ourston, Science Applications International Corporation, 3045 Technology Parkway, Orlando, FL 32826-3299, USA. E-mail: cn161@cleveland.freenet.edu.

SSDI 0004-3702(93)E0018-H

systems that automatically acquire the requisite knowledge from experience. However, *empirical learning* systems [29,41,46] do not take significant advantage of existing domain knowledge and *explanation-based learning* systems [9,30] require a complete and correct domain theory. Consequently, a number of recent research projects have focused on integrating these two basic approaches to machine learning [4,47].

Normal knowledge acquisition can be divided into two phases: an initial phase in which a knowledge engineer extracts a rough set of rules from an expert, and *knowledge base refinement*, in which the initial knowledge base is refined to produce a high-performance system [18].[1] The initial knowledge base is acquired as whole rules, or sets of rules, that are used to represent various concepts in the domain. In contrast, during knowledge base refinement, components of the existing rules are modified, in addition to adding and deleting rules, in an effort to improve the *empirical adequacy* of the knowledge base, that is, its ability to reach correct conclusions in its domain.

This article presents a method for automating knowledge base refinement for classification systems employing a propositional Horn-clause theory. The method assumes that an approximately correct initial knowledge base (*domain theory*) is obtained from a textbook or an expert. The method attempts to make small syntactic changes to a domain theory to make it consistent with a provided set of training examples. The advantage of a refinement approach to knowledge acquisition as opposed to a purely empirical learning approach is two-fold. First, by starting with an approximately correct theory, a refinement system should be able to achieve high performance with significantly fewer training examples. Therefore, in domains in which training examples are scarce or in which a rough theory is easily available, the refinement approach has a distinct advantage. Second, theory refinement results in a structured knowledge base that maintains the intermediate terms and explanatory structure of the original theory. Empirical learning, on the other hand, results in a decision tree or disjunctive normal form expression with no intermediate terms or explanatory structure. Therefore, a knowledge base formed by theory refinement is much more suitable for supplying meaningful explanations for its conclusions, an important aspect of the usefulness of an expert system.

The theory refinement system we have developed, EITHER (Explanation-based and Inductive THeory Extension and Revision), is modular and contains independent subsystems for deduction, abduction, and induction.

[1] Bareiss, Porter and Murray [1] divide the knowledge base refinement phase into two stages: the first establishes the correctness of the knowledge base, and second improves efficiency. This paper is only concerned with correctness.

Each of these reasoning components make important contributions to the overall goal of the system. EITHER attempts to integrate analytical methods (deduction and abduction) and empirical methods (induction) in order to combine their individual strengths. The analytical part of the system is used to identify the failing parts of the theory, and to constrain the examples used for induction. The empirical part of the system determines the specific corrections to failing rules that make them consistent with the training examples.

EITHER has successfully refined two real-world rule bases, one in molecular biology and one in plant pathology. The empirical results confirm the hypotheses that theory refinement improves the classification accuracy of the original knowledge base and produces a more accurate classifier than simple induction over the examples. That is, combining theory and data is better than using either one alone. In addition, unlike other existing theory refinement systems, EITHER is guaranteed to produce a theory that is consistent with the training data. Given a theory with arbitrary errors and a consistent set of training examples, the system will return a revised version that classifies all of the examples correctly.

The body of the paper is organized as follows. Section 2 defines the specific problem that EITHER addresses and presents an overview of the system. Section 3 details the basic theory revision algorithm. Section 4 addresses special problems that arise with multiple category theories. Section 5 describes the methods used to determine the place in the theory requiring correction. Section 6 presents a complexity analysis of the EITHER algorithm. Section 7 presents experimental results on revising two actual expert rule bases. Section 8 discuss the relation between EITHER and other recent developments in knowledge-based learning and theory refinement. Section 9 discusses future research issues revealed by the EITHER project. Section 10 summarizes the current results and draws some conclusions.

2. Overview

First, we define the specific problem addressed by EITHER and give a simple example that will be used throughout the paper. Next, we present a taxonomy of errors that is useful in isolating and correcting problems with a propositional Horn-clause theory. Finally, we review how EITHER combines deductive, abductive, and inductive reasoning to solve the theory refinement problem.

2.1. Problem definition

Stated succinctly, the problem addressed by EITHER is:

Given: An imperfect domain theory for a set of categories and a set of classified examples each described by a set of observable features.

Find: An approximately minimal syntactic revision of the domain theory that correctly classifies all of the examples.

Horn-clause logic was chosen as the representational formalism. This provides a relatively simple and useful language for exploring the problems associated with theory revision. Theories currently are restricted to an extended propositional logic that contains feature–value pairs and thresholds on real-valued (*linear*) features as well as binary propositions. In addition, domain theories are required to be acyclic and therefore define a directed acyclic graph (DAG). For the purpose of theory refinement, EITHER makes a closed-world assumption. If the theory cannot prove that an example is a member of a category, then it is assumed to be a negative example of that category. The domain theories upon which EITHER has been tested have all corresponded to *classification* tasks—assigning examples to one of a finite set of predefined categories.

Propositions that are used to describe the examples (e.g. (color black)) are called *observables*. To avoid problems with negation as failure, only observables can appear as negated antecedents. Propositions that represent the final concepts in which examples are to be classified are called *categories*. It is currently assumed that all categories are disjoint. The set of categories may include negative, which is the default category for an example that is not provable as a member of any other category. In a normal domain theory, all of the sources (leaves) of the DAG are observables and all of the sinks (roots) are categories; however, gaps in the original theory may cause these constraints to be violated. Propositions in the theory that are neither observables nor categories are called *intermediate concepts*.

It is difficult to precisely define the notion of a "minimally revised" theory. Since it is assumed that the original theory is "approximately correct" the goal is to change it as little as possible. Syntactic measures such as the total number of symbols added or deleted are reasonable criteria. EITHER uses various heuristic methods to help insure that its revisions are minimal in this sense. However, finding a revision that is guaranteed to be syntactically minimal is computationally intractable. When the initial theory is empty, the problem reduces to that of finding a minimal Horn-clause theory for a set of examples. [2]

A sample theory suitable for EITHER is a version of the cup theory [55]

[2]Related problems like finding a minimum DNF formula are known to be NP-complete [16]; however, we were unable to find a reference to an NP-completeness result for the specific problem of finding a minimum Horn-clause theory. Nevertheless, there are certainly no known polynomial algorithms for solving this sort of optimization problem.

1. (cup) ← (stable) ∧ (liftable) ∧ (open-vessel)
2. (stable) ← (has-bottom) ∧ (flat-bottom)
3. (liftable) ← (graspable) ∧ (lightweight)
4. (graspable) ← (has-handle)
5. (graspable) ← (width small) ∧ (styrofoam)
6. (graspable) ← (width small) ∧ (ceramic)
7. (open-vessel) ← (has-concavity) ∧ (upward-pointing-concavity)

Fig. 1. The cup theory.

shown in Fig. 1. This theory will be used extensively throughout the remainder of the article for illustrative purposes. Figure 2 shows six examples that are consistent with this theory, three positive examples of cup and three negative examples. Each example is described in terms of twelve observable features. There are eight binary features: has-concavity, upward-pointing-concavity, has-bottom, flat-bottom, lightweight, has-handle, styrofoam, and ceramic; three discrete features: color, width, and shape; and a single linear feature: volume. Given various imperfect versions of the cup theory and

	has-concavity	upward-pointing	has-bottom	flat-bottom	lightweight	has-handle	styrofoam	ceramic	color	width	volume	shape
1. +	X	X	X	X	X		X		red	sm	8	hem
2. +	X	X	X	X	X	X		X	blue	med	16	hem
3. +	X	X	X	X	X	X		X	tan	med	8	cyl
4. -	X	X	X	X	X				gray	sm	8	cyl
5. -	X	X	X	X	X		X		red	med	8	hem
6. -	X	X	X	X	X			X	blue	med	16	hem

Fig. 2. Cup examples.

these six examples, EITHER can regenerate the correct theory. For example, if rule 4 is missing from the theory, examples 2 and 3 are no longer provable as cups. If the antecedent (width small) is missing from rule 5, then negative example 5 becomes provable as a cup. EITHER can correct either or both of these errors using the examples in Fig. 2.

EITHER operates in batch mode, processing a complete set of training examples at once. The training examples normally contain both correctly and incorrectly classified examples. The incorrectly classified examples, or *failing* examples, are used to identify errors and to control the correction. The correctly classified examples are used to focus the correction and to

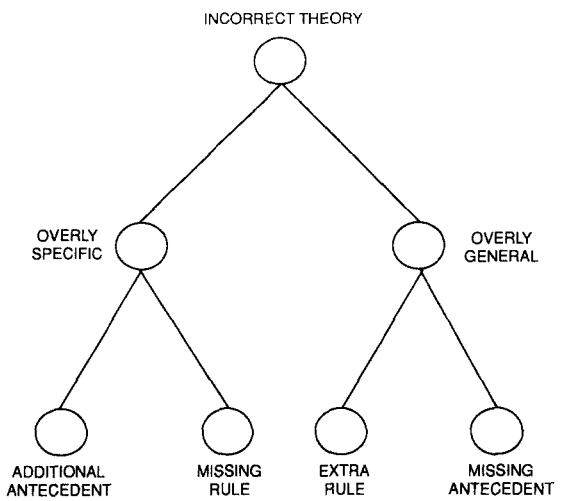

Fig. 3. Theory error taxonomy.

limit its extent. An important property of EITHER is that it is guaranteed to produce a revised theory that correctly classifies all of the training examples, provided they are *consistent*. A set of training examples is consistent if any two examples described by the same set of features are assigned to the same category.

2.2. *Types of theory errors*

Figure 3 shows a taxonomy for theory errors in propositional Horn-clause theories. At the top level, theories can be incorrect because they are either overly general or overly specific. An overly general theory entails category membership for examples which are not members of the category. This will result in negative examples of a concept being proven as positive. One way a theory can be overly general is when rules lack required antecedents, providing proofs for examples which should have been excluded. Another way in which examples can be erroneously included is by having additional rules in the category definition which are not correct. The additional rules provide proofs of category membership for examples which do not properly belong in the category. By contrast, an overly specific theory fails to entail category membership for members of a concept. This can occur because the theory is missing a rule which is required in the proof of concept membership, or because the existing rules have additional antecedents which exclude concept members.

The following terminology is used in the remainder of the paper. "The example is provable", is used to mean "the example is provable as a member of its own category". A *failing positive* refers to an example that is not provable as a member of its own category. A *failing negative* refers to an

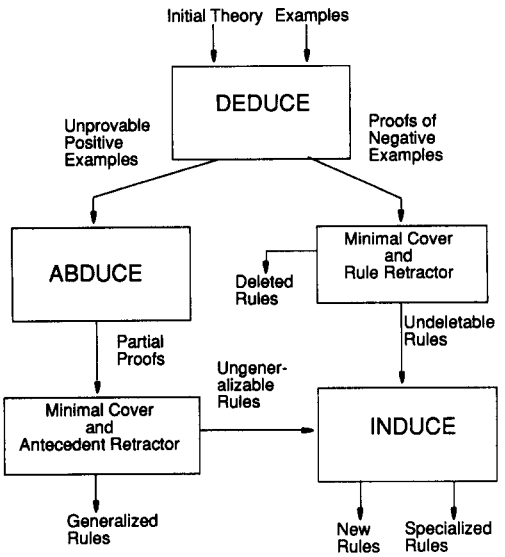

Fig. 4. EITHER architecture.

example that is provable as a member of a category other than its own. Notice that a single example can be both a failing negative and a failing positive.

2.3. *EITHER components*

As shown in Fig. 4, EITHER uses a combination of methods to revise a theory. It first attempts to fix failing positives by removing or generalizing antecedents and to fix failing negatives by removing rules or specializing antecedents since these are simpler and less powerful operations. Only if these operations fail does the system resort to the more powerful technique of using induction to learn new rules to fix failing positives and to add antecedents to existing rules to fix failing negatives.

Horn-clause deduction is the basic inference engine used to classify examples. EITHER initially uses deduction to identify failing positives and negatives among the training examples. It uses the proofs generated by deduction to find a near-minimum set of rule retractions that would correct all of the failing negatives. During the course of the correction, deduction is also used to assess proposed changes to the theory as part of the generalization and specialization processes.

EITHER uses abduction to initially find the incorrect part of an overly specific theory. Abduction identifies sets of assumptions which would allow a failing positive to become provable. These assumptions identify rule antecedents (called *conflicting antecedents*) that, if deleted, would properly generalize the theory and correct the failing positive. EITHER uses the output

of abduction to find a near-minimum set of conflicting antecedents whose removal would correct all of the failing positives.

Induction is used to learn new rules or to determine which additional antecedents to add to an existing rule. In both cases, EITHER uses the output of abduction and deduction to determine an appropriately labeled subset of the training examples to pass to induction in order to form a consistent correction. EITHER currently uses a version of ID3 [41] as its inductive component. The decision trees returned by ID3 are translated into equivalent Horn-clause rules [42]. The remaining components of the EITHER system constitute generalization and specialization control algorithms, which identify and specify the types of corrections to be made to the theory.

One of the main advantages of the EITHER architecture is its modularity. Because the control and processing components are separated from the deductive, inductive, and abductive components, these latter components can be modified or replaced as the need arises. For example, the time complexity of EITHER's abduction algorithm is exponential in the size of the theory. However, this algorithm could be exchanged for one using an ATMS (Assumption-based Truth Maintenance System) and beam search [36] to improve efficiency, without noticeably affecting the remainder of the system.

3. The basic theory revision algorithm

This section details EITHER's method for modifying *leaf rules*, which are rules whose antecedents include an observable or an intermediate concept that is not the consequent of any existing rule. The discussion is based on single-category theories such as the cup theory in Fig. 1. Sections 4 and 5 discuss enhancements for dealing with multiple categories and higher-level rules, respectively. Section 4.2 discusses the reasons for initially focusing on leaf rules.

Figure 5 illustrates EITHER's response to an incorrect theory. First, deduction is used to classify all of the training examples according to the initial theory. EITHER employs a standard backward-chaining Horn-clause theorem prover, like PROLOG. Failing positives signal the need for theory generalization, which is discussed in Section 3.2. Failing negatives signal the need for theory specialization, the subject of Section 3.3. The corrections made by these algorithms are independent: a theory may be generalized, specialized, or both, as dictated by the failing examples. In each case, the corrections made to the theory are *non-interfering*, that is, the prescribed theory generalizations are guaranteed not to introduce new specialization problems (failing negatives) and the theory specializations are guaranteed not to introduce new generalization problems (failing positives).

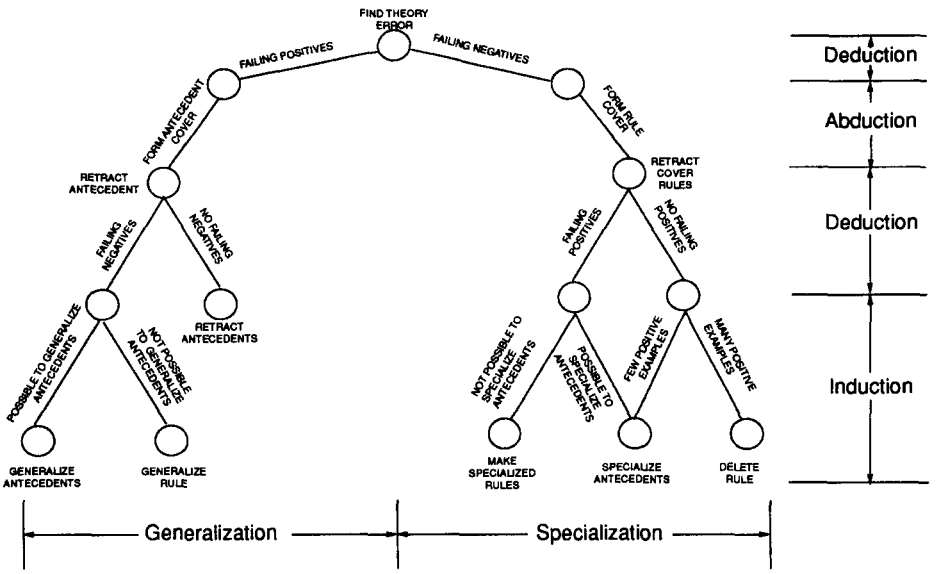

Fig. 5. EITHER system response to theory errors.

3.1. Finding the minimum covers

The input to both theory generalization and specialization is a *cover*, a complete set of leaf rules requiring correction.[3] There are two types of covers used by EITHER: the *antecedent cover* and the *rule cover*. The antecedent cover is used by the generalization procedure to fix all failing positives. The rule cover is used by the specialization procedure to fix all failing negatives. There is an essential property that holds for both types of cover:

> If all of the elements of the cover are removed from the theory, the examples associated with the cover will be correctly classified.

Specifically, if all of the antecedents in the antecedent cover are removed, the theory is generalized so that all of the failing positives are fixed and if all of the rules in the rule cover are removed, the theory is specialized so that all of the failing negatives are fixed. In each case, EITHER attempts to find a *minimum* cover in order to minimize change to the initial theory.

Since EITHER finds a complete cover for both classes of examples, and since the corrections to the theory are *non-interfering*, the rule updates made by EITHER are guaranteed to be consistent with the training examples. Consistency of the rule updates with the training examples leads ultimately to a guarantee of the eventual convergence of the EITHER algorithm to a PAC (Probably Approximately Correct) concept [20,37].

[3]Minimum covering methods have historically been used in machine learning for the induction of DNF formulae [26]. Here we have adapted them for use in theory refinement.

3.1.1. The minimum antecedent cover

Abduction [6,23] is used to find antecedents whose removal would help fix failing positives. The normal logical definition of abduction is:

> **Given**: A domain theory, T, and an observed fact, O.
> **Find**: All minimal sets of atoms, A, called *assumptions*, such that $A \cup T$ is logically consistent and $A \cup T \models O$.

The assumptions A are said to *explain* the observation. Legal assumptions are frequently restricted, such as allowing only instances of certain predicates (*predicate specific abduction*) or requiring that assumptions not be provable from more basic assumptions (*most-specific abduction*) [49].

In order to focus on leaf rules, EITHER's abductive component backchains as far as possible before making an assumption (most-specific abduction). The consistency constraint is removed in order to allow assumptions to be viewed as antecedent retractions. Since an observation states that an example is a member of a category ($E \rightarrow C_E$, where E is the conjunction of observable features of the example and C_E is its category), abduction finds all minimal sets of most-specific atoms, A, such that:

$$A \cup E \cup T \models C_E, \tag{1}$$

where minimal means that no assumption set is a subset of another. The proof supported by each such set is called a *partial proof*. EITHER currently uses an abductive component that employs exhaustive search to find all partial proofs of each failing positive example [35]. Partial proofs are used to indicate conflicting antecedents that, if retracted, would allow the example to become provable. The above definition guarantees that, if all of the assumptions in a set are removed from the antecedents of the rules in their corresponding partial proof, the example will become provable. This is because not requiring a fact for a proof has the same generalizing effect as assuming it.

As a concrete example, assume that rule 4 about handles is missing from the cup theory as presented in Fig. 1. This will cause example 2 from Fig. 2 to become a failing positive. Abduction finds two minimal sets of conflicting antecedents: {(width small)$_6$} and {(width small)$_5$ (styrofoam)$_5$}. The subscripts indicate the number of the rule to which the antecedent belongs, since each antecedent of each rule must be treated distinctly. Notice that removing the consistency constraint is critical to the interpretation of assumptions as antecedent retractions. Assuming (width small) is inconsistent when (width medium) is known; however, retracting (width small) as antecedent from one of the graspable rules is still a legitimate way to help make this example provable.

In a complex problem, there will be many partial proofs for each failing positive. In order to minimize change to the initial theory, EITHER attempts

to find the minimum number of antecedent retractions required to fix *all* of the failing positives. In other words, we want to make the following expression true:

$$E_1 \wedge E_2 \wedge \cdots \wedge E_n, \tag{2}$$

where E_i represents the statement that the ith failing positive has at least one completed partial proof, that is,

$$E_i \equiv P_{i1} \vee P_{i2} \vee \cdots \vee P_{im}, \tag{3}$$

where P_{ij} represent the statement that the jth partial proof of the ith failing positive is completed, that is,

$$P_{ij} \equiv A_{ij1} \wedge A_{ij2} \wedge \cdots \wedge A_{ijp}, \tag{4}$$

where the A_{ijk} means that the antecedent represented by the kth assumption used in the jth partial proof of the ith example is removed from the theory. In order to determine a minimum change to the theory, we need to find the minimum set of antecedent retractions (A's) that satisfy this expression. Pursuing the example of the cup theory that is missing the rule for handles, both failing positives (examples 2 and 3) have the same partial proofs, resulting in the expressions:

$$E_2 \equiv (\text{width small})_6 \vee (\text{width small})_5 \wedge (\text{styrofoam})_5,$$
$$E_3 \equiv (\text{width small})_6 \vee (\text{width small})_5 \wedge (\text{styrofoam})_5.$$

In this case, the minimum antecedent cover is trivial and consists of retracting the single antecedent $(\text{width small})_6$.

Since the general minimum set covering problem is NP-hard [16], EITHER uses a version of the *greedy covering algorithm* to find the antecedent cover. The greedy algorithm does not guarantee to find the minimum cover, but will come within a logarithmic factor of it and runs in polynomial time [22]. The algorithm iteratively updates a partial cover, as follows. At each iteration, the algorithm chooses a partial proof and adds its antecedent retractions to the evolving cover. The chosen partial proof is the one that maximizes *benefit-to-cost*, defined as the ratio of the additional examples covered when its antecedents are included, divided by the number of antecedents added. The set of examples that have the selected partial proof as one of their partial proofs are removed from the examples remaining to be covered. The process terminates when all failing positives are covered.

3.1.2. The minimum rule cover

The proofs of failing negatives generated by the deductive component are used to determine the minimum rule cover. In order to minimize change to the initial theory, EITHER attempts to find the minimum number of leaf

rule retractions required to fix *all* of the failing negatives. In analogy with the previous section, we would like to make the following expression true:

$$\neg E_1 \wedge \neg E_2 \wedge \cdots \wedge \neg E_n, \tag{5}$$

where E_i represents the statement that the ith failing negative has a complete proof, that is,

$$\neg E_i \equiv \neg P_{i1} \wedge \neg P_{i2} \wedge \cdots \wedge \neg P_{im}, \tag{6}$$

where P_{ij} represent the statement that the jth proof of the ith failing negative is complete, that is,

$$\neg P_{ij} \equiv \neg R_{ij1} \vee \neg R_{ij2} \vee \cdots \vee \neg R_{ijp}, \tag{7}$$

where $\neg R_{ijk}$ represents the statement that the kth leaf rule used in the jth proof of the ith failing negative is removed, i.e. a proof is no longer complete if at least one of the rules used in the proof is removed.

As with the antecedent cover, EITHER attempts to find a minimum cover of rule retractions using greedy covering. In this case, the object is to remove all proofs of every failing negative. Note that in picking a retraction, EITHER avoids rules that do not have any disjuncts in their proof path to the goal since these rules are needed to prove *any* example. At each step in the covering algorithm, the eligible rule that participates in the most faulty proofs is added to the evolving cover until all the faulty proofs are covered.

As an example, consider the cup theory in which the (width small) antecedent is missing from rule 5. In this case, example 5 becomes a failing negative. The minimum rule cover is the overly general version of rule 5:

(graspable) ← (styrofoam)

since it is the only rule used in the faulty proof with alternative disjuncts (rules 4 and 6).

3.2. Theory generalization

The left side of Fig. 5 illustrates the generalization process. EITHER first forms the minimum antecedent cover, as discussed in the previous section. The conflicting antecedents in the cover are associated with their corresponding rules, with one or more conflicting antecedents per rule. Each such rule has associated with it the failing positive examples that use the rule in a chosen partial proof. Each rule in the cover is sequentially generalized so that it fixes its failing positives without creating additional failing negatives.

There are three operators EITHER can use to generalize a rule. They are:

- antecedent retraction,
- antecedent generalization,

- inductive rule addition.

The operators are tried in the order given in an attempt to minimize change to the initial theory.

3.2.1. Antecedent retraction

For each rule in the cover, the first step is to remove its conflicting antecedents. If removing the antecedents does not over-generalize the theory by causing new failing negatives,[4] the antecedents are permanently deleted.

An exception to this policy occurs when all of a rule's antecedents are conflicting. In this case, EITHER removes the consequent of the rule as an antecedent from those *parent rules*[5] that are used in the partial proof of one of its failing positives. This limits the correction to just those rules associated with the failing positive examples. If the original rule had all of its antecedents removed, all of its parent rules would, in effect, be generalized. This generalization is unnecessary when the parent rule was not actually used in any of the partial proofs represented in the cover.

3.2.2. Antecedent generalization

If removing antecedents is an over-generalization, EITHER attempts to generalize the conflicting antecedents just enough to cover the rule's failing positive examples. Since antecedent generalization uses the existing features in the rule, it is preferred to inductive rule addition which, in general, will use entirely new features.

How an antecedent is generalized depends on whether its feature is binary, discrete, or linear. For linear antecedents, the interval in the rule is extended just enough to cover its failing positive examples. For discrete antecedents, disjuncts are added for the values present in the failing positive examples. If all of the values are required to account for the failing positive examples, a discrete antecedent is simply removed. For binary antecedents, the antecedent is removed. For example, if the initial rule is[6]

$$(\text{graspable}) \leftarrow (\text{has-handle}) \wedge (\text{color red}) \wedge (\text{volume ?x}) \wedge (\leqslant \text{?x } 3)$$

and it has a single failing positive with the features (not (has-handle)), (color blue), (volume 4), it will be generalized to the rules:

$$(\text{graspable}) \leftarrow (\text{color red}) \wedge (\text{volume ?x}) \wedge (\leqslant \text{?x } 4),$$
$$(\text{graspable}) \leftarrow (\text{color blue}) \wedge (\text{volume ?x}) \wedge (\leqslant \text{?x } 4).$$

[4]For multi-category theories, an existing failing negative that becomes provable in additional incorrect categories also counts as an over-generalization error.

[5]Rule A is a parent of rule B iff the consequent of B is an antecedent of A.

[6]A leading question mark denotes a variable. Variables can only be used to specify ranges on linear features.

Like retraction, antecedent generalization is successful if it does not introduce any new failing negatives. Consequently, antecedent generalization is a *one-sided* generalization [20]: only the positive examples are considered for the generalization, the negative examples are used simply to determine if the generalization was successful.

3.2.3. Inductive rule addition

If both antecedent retraction and generalization result in over-generalization, the inductive component is used to learn entirely new rules for the consequent of the given rule. The set of positive and negative examples for the inductive rule formation is determined as follows. The positive examples are simply the failing positives for the rule. The negative examples are obtained by removing all of the antecedents from the rule and collecting any new failing negatives that are created. This is necessary because if our only goal was to make the positive examples provable, removing all of the antecedents would suffice. Therefore, antecedents are added to the new rule to ensure that no additional failing negatives are created while still covering the failing positives. It can also be viewed as a proof by contradiction: we assume that the consequent of the rule is true and obtain the contradiction that a negative example is provable, implying that the consequent is not true for the negative example.

As an illustration of this process, consider the running example of the cup theory missing the rule for handles. EITHER initially focuses on generalizing one of the remaining rules for graspable. The failing positive examples with the incorrect theory are examples 2 and 3 from Fig. 2, both of which are covered by the conflicting antecedent (width small)$_6$. However, removing (width small) from rule 6 results in example 6 becoming a failing negative. Generalizing the conflicting antecedent to include (width medium) also causes example 6 to fail. As a result, EITHER uses induction to form a new rule for graspable. In this case, the positive examples for the induction are examples 2 and 3, that is, the original failing positive examples. The negative examples are examples 4, 5 and 6, which become provable when graspable is assumed to be true. Since has-handle is the only single feature that distinguishes examples 2 and 3 from examples 4, 5 and 6, the inductive system (ID3) generates the required rule:

(graspable) ← (has-handle).

3.3. Theory specialization

The right side of Fig. 5 illustrates the specialization process. EITHER first forms the minimum rule cover, as discussed in Section 3.1.2. Next, each rule in the cover is sequentially specialized so that it excludes its failing negatives without creating additional failing positives.

EITHER uses the following operators to specialize a rule:

- rule retraction,
- antecedent specialization,
- inductive antecedent addition.

As with generalization, these operators are tried successively in the order given.

3.3.1. Rule retraction

The first step in the specialization process is to determine the effect of removing the rule from the theory. If no new failing positive examples result from retracting the rule, EITHER checks to see if a sufficient number of positive examples have been seen or if specializing a linear antecedent represents a superior correction. If only a few positive examples have been seen, the fact that retracting the rule caused no failures may simply be due to the insufficient number of examples. Hence, the relatively large semantic change to the theory caused by rule retraction is probably not warranted.

The superiority of an antecedent specialization is indicated by the minimum *exclusion factor* for the rule. For a given linear antecedent, the first step in determining the exclusion factor is to find the range of values for the corresponding feature among the negative examples for the rule.[7] This range is then divided by the size of the interval specified in the rule to determine the exclusion factor. A small exclusion factor indicates that antecedent specialization is desirable since it means that the interval specified in the rule would only have to be changed by a small amount in order to exclude the failing negative examples. For example, suppose that the rule contains antecedents for the constraint $0 \leqslant a \leqslant 1000$. Assume rule negative examples have been seen with values for a of 999 and 1000. Then these examples could be *excluded* from the coverage of the rule by changing the interval to: $0 \leqslant a < 999$, a relatively small change to both the definition of the rule and its coverage (the exclusion factor is 1/1000). If the rule has several linear antecedents, the minimum exclusion factor is chosen since only one antecedent needs to be specialized to exclude the negative examples.

The choice between rule retraction and antecedent specialization is determined as follows:

if $(n > s/e)$
then retract rule
else specialize antecedents

[7]For each negative example, the feature value will always be within the interval for the rule, since otherwise the rule could not have been used in a proof for the negative example.

where n is the number of positive examples that have been seen, s is the number of symbols in the rule to be retracted, and e is the exclusion factor for the rule (if there are no linear antecedents in the rule, the exclusion factor is set to ∞). The number of symbols in the rule is included in this formula since this represents the amount of syntactic change to the theory when the rule is retracted.

3.3.2. Antecedent specialization

If either rule retraction fails or antecedent specialization is determined to be superior, EITHER tries to specialize the antecedents of the rule just enough to exclude the failing negatives. This is a one-sided specialization which attempts to specialize the rule away from the provable negative examples without considering the positive examples (if any). If doing so does not introduce additional failing positive examples, then the specialization is successful. Attempting antecedent specialization prior to inductive antecedent addition is justified because it restricts the changes to the rule's existing features.

Unlike antecedent generalization, only linear antecedents can be specialized. If a rule references multiple linear features, the minimum exclusion factor, defined in the previous section, is used to select the best linear antecedent to specialize. As illustrated by the example in the previous subsection, the linear interval is minimally reduced so that no negative examples are covered.

3.3.3. Inductive antecedent addition

If the previous specialization attempts over-specialize by creating additional failing positives, EITHER uses the inductive component to add new antecedents to the rule. The system associates positive and negative examples with the overly general rule and uses the inductive component to find a small set of additional antecedents to add to the rule to fix the failing negatives without creating any additional failing positives.

The negative examples for induction are the failing negatives that use the rule in an erroneous proof, since these are the examples that need to be filtered out by the new antecedents. The positive examples are those that become failing positives when the rule is removed from the theory, since these are the examples that are relying on the current rule for their correct categorization. This selection of examples is essentially the dual of that used for inductive rule addition as described in Section 3.2.

For example, again consider the case of missing the antecedent (width small) from rule 5. Based on the rule cover, EITHER first removes the overly general rule 5:

(graspable) ← (styrofoam)

and tests for additional failing positives. Since example 1 becomes unprovable in this case and since the binary antecedent (styrofoam) cannot be specialized, EITHER decides to add additional antecedents. Example 1 (the failing positive created by retraction) is used as a positive example and example 5 (the original failing negative) is used as a negative example. Since width is the only feature that distinguishes these two examples, ID3 learns the rule

(positive) ← (width small).

This is combined with the original rule to obtain the correct replacement rule:

(graspable) ← (width small) ∧ (styrofoam).

4. Multiple category theories: the correctability problem

For the most part, the procedure described in the previous section applies directly to multiple-category theories. However, in certain situations it is impossible to correct a multiple-category theory by modifying only leaf rules. Therefore, EITHER must choose rules to revise that are *correctable*, where a correctable rule can be modified to properly discriminate between its positive and negative examples. In the case of a rule which is not correctable and requires specialization, any specialization which eliminates failing negative examples will also create failing positives. Similarly, for an incorrectable rule requiring generalization, any generalization that fixes failing positive examples will also create failing negatives. This section defines the correctability problem and describes how EITHER determines a correctable set of rules from the initial covers.

4.1. The reasons for the correctability problem

As discussed in Section 3, the generalization and specialization processes start with a cover of leaf rules, the antecedent cover and the rule cover, respectively. In certain cases, a leaf rule in the initial cover will not be correctable. For example, consider the simple theory

$C_1 \leftarrow R$
$C_2 \leftarrow R$
$R \leftarrow a \wedge b$

where C_1 and C_2 are categories and a and b are observables. This is a pathological theory in which any example will be provable in both categories or neither, and the same remark applies when any *change* is made to the leaf rule for R. As a result, the "R" rule is not correctable. In general, the problem

is detecting that such a condition exists and finding a set of corrections that will classify the examples correctly.

To illustrate the impact of this problem, suppose we have a C_1 example and that this example is provable as C_1 using the initial theory. Therefore, the example also will be provable in category C_2, meaning the theory is overly general. Removing the "R" rule (the first attempted step in the specialization process) will cause the example to fail in category C_2, but will also cause it to fail in its own category. What this means is that the same example is both a positive example (requires the "R" rule in a proof of C_1), and negative example (the example is provable in C_2 using the "R" rule) for the specialization to the "R" rule. If the theory was overly specific, then the example would fail in C_1, but would become provable in C_2 when the conflicting antecedents of the rule were removed. Again, the same example would be both a positive and negative example for the required generalization to the rule. Examples that show such behavior are called *overlapping*. That is, overlapping examples are both positive and negative examples for a rule.

4.2. The response to the correctability problem

Fortunately, there is a simple solution to the correctability problem. In the worst case, a cover can be selected consisting entirely of *category rules* (rules whose consequents are categories). Since these rules imply a single category, updates to them cannot affect membership in other categories. For example, if a given example is erroneously provable as a member of a particular category, then specializing the antecedents of the corresponding category rule will cause the example not to be provable in the category without affecting membership in other categories.

However, we would prefer *not* to make the corrections at the root of the theory. This is because strengthening lower-level rules allows them to participate in more than one category, thereby strengthening the theory as a whole, rather than just a single category. Intermediate concepts that participate in more than one category are called *shared concepts*. Consider the example of missing the handle rule from an enlarged version of the cup theory that includes categories for pots, pans, buckets, etc. Clearly, a new category rule for cup could be learned that includes any handled cups and excludes all non-cups. However, this rule would be more complicated than the has-handle rule and all of the other categories that use the shared concept for graspable would not have the benefit of the correction.

Preferring to modify lower-level rules also allows EITHER to exhibit *cross-category transfer*. This refers to the interesting effect that revising rules for shared concepts frequently can improve performance on test data that is drawn from a completely different population than the training data. For

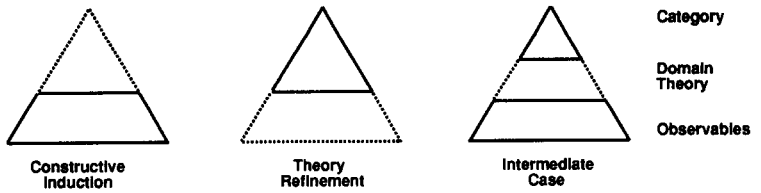

Fig. 6. Different types of gaps in incomplete theories. Dashed lines indicate gaps to be filled by induction.

example, if the system is trained only on cups the system can improve its classification performance on pots and pans by modifying its shared sub-theory for graspable. Empirical results on cross-category transfer in EITHER are reported in [38].

Because of these considerations, EITHER initially starts with leaf rules and only modifies higher-level rules if necessary.[8] If the rules in the initial cover have no overlapping examples, then no changes to higher-level rules are required. If, however, there are rules with overlapping examples, EITHER replaces each such rule with its parent rules, and tests the parents for overlapping examples. This process continues until a set of rules is obtained that introduces no overlapping examples. In the limit, the cover will consist entirely of category rules, which are always correctable. Once a correctable cover is found, generalization and specialization proceed as in the previous section.

5. Revising higher-level rules

Although EITHER's initial bias is to revise leaf rules, it is also capable of identifying and correcting errors at higher levels in the theory. Figure 6 illustrates how a theory can have gaps at various levels. Previous research has focused on handling gaps at a particular level in a theory. For example, some research in constructive induction [12,27] assumes that the existing domain theory defines a set of intermediate concepts (derived features) in terms of observables. The rules connecting these intermediate concepts to the categories are assumed to be missing and must be learned using induction. The theory is used to derive values for all of the intermediate concepts and these are used as additional input features for induction of the category rules. This is the first situation illustrated in Fig. 6 where there is a gap at the "top" of the theory. For example, imagine the category rule for concluding cup was missing from the cup theory.

[8]Or if higher-level corrections are syntactically simpler (see Section 5).

Other research in the refinement of incomplete theories with missing rules [8] assumes that the domain theory has correct rules for inferring categories from intermediate concepts but is instead missing rules connecting observables to intermediate concepts. Partial explanations (incomplete proofs) are used to isolate intermediate concepts that should be provable for some examples but are not. Induction is then used to learn rules for inferring these intermediate concepts from observables. This is the second situation illustrated in Fig. 6 where there is a gap at the "bottom" of the theory. For example, imagine one of the rules for inferring graspable is missing from the cup theory.

A third case is illustrated in the final situation in Fig. 6 where there is a gap is in the "middle" of the theory. For example, imagine the rule for inferring liftable was missing from the cup theory. None of the previous research seems to directly address this issue.

An ideal system should be able to deal with multiple gaps occurring at arbitrary levels in the domain theory. It should also be able to introduce new intermediate concepts in order to handle the situation in which the gap in the theory spans multiple levels. For example, imagine that all rules for inferring both liftable and graspable were missing from the cup theory. In this case, the intermediate concept graspable is not even present in the theory and must be created.

EITHER combines a number of previous techniques from theory refinement and constructive induction in order to deal with this general problem. *Consequent identification* (Section 5.1) identifies the level in the theory that needs correcting. *Concept utilization* (Section 5.2) employs existing rules in the theory to derive high-level features from the data. *Concept creation* (Section 5.3) employs inverse resolution operators [33] to introduce new intermediate concepts in order to fill a gap in the theory spanning multiple levels.

5.1. Consequent identification

The basic EITHER procedure focuses on rules at the "bottom" of the theory, where changes generally have fewer ramifications and can improve multiple categories. Therefore, the basic procedure easily handles the middle case in Fig. 6 where there are missing or buggy leaf rules, as illustrated by the examples in Section 3 involving modifying the rules for graspable.

Since altering higher-level concepts is sometimes preferable, EITHER uses a simple hill-climbing algorithm to determine which level in the existing theory to modify. After forming a correction to the leaf rules identified in the minimum cover, it determines alternative corrections to each rule's parent that would fix the same problems. If the correction to the parent rule is more complex, then EITHER uses the lower-level correction. If the

parent rule correction is less complex, EITHER continues up the theory and examines the corrections required for the parent of the parent rule. This iterative procedure terminates when the correction at the next-higher level in the theory is more complex than the correction at the current level or when the top-level category rule is reached. As an example of this process, assume that the cup theory has been incorrectly specialized by adding the antecedent manipulatable to the liftable rule, and an additional rule for manipulatable:

$$\begin{aligned}&(\text{liftable}) \leftarrow (\text{graspable}) \wedge (\text{lightweight}) \wedge (\text{manipulatable}),\\&(\text{manipulatable}) \leftarrow (\text{has-handle}) \wedge (\text{volume ?x})\\&\qquad\qquad\qquad \wedge (\geqslant \text{?x } 8) \wedge (\leqslant \text{?x } 12).\end{aligned}$$

In correcting this theory, EITHER first proposes changes to the manipulatable rule based on antecedent generalization, resulting in the following rule:

$$(\text{manipulatable}) \leftarrow (\text{volume ?x}) \wedge (\geqslant \text{?x } 8) \wedge (\leqslant \text{?x } 16).$$

EITHER next considers changes to the parent of the manipulatable rule, the liftable rule. In this case, the proposed correction, obtained through antecedent retraction, is the (correct) rule:

$$(\text{liftable}) \leftarrow (\text{graspable}) \wedge (\text{lightweight}).$$

Since this correction is syntactically simpler than the correction to the manipulatable rule, the process continues. EITHER next checks the correction to the parent of the liftable rule, the cup rule. The proposed correction is to introduce a new rule for cup, obtained through induction,

$$(\text{cup}) \leftarrow (\text{lightweight}) \wedge (\text{stable}) \wedge (\text{graspable}).$$

Since this is a larger syntactic change to the theory than that proposed for the liftable rule, the liftable correction is adopted.

5.2. Concept utilization

Concept utilization identifies intermediate concepts in the theory that can be used as antecedents during inductive rule and antecedent addition. First, forward chaining from the observables identifies the truth values of all intermediate concepts for each of the failing examples. These intermediate concepts are then fed to the inductive learner as additional features. In this way, if an intermediate concept is highly correlated with the class of the failing examples, then this concept is returned as an antecedent in the rules formed by the inductive learner. This approach allows the system to learn rules that fill gaps in either the "middle" or the "top" of the theory.

For example, assume that the cup theory is missing the rule for liftable. Forward chaining on the failing positives (in this case, all of the positive examples) will always add the feature graspable, since it is true for all positive

examples. On the other hand, no negative example will deduce both graspable and lightweight, since no negative example is liftable. Remember the negative examples for rule addition are those that become failing negatives when liftable is assumed true. Therefore, given enough examples, the inductive learner will select the intermediate concept graspable as an antecedent in the new rule for liftable. The observable lightweight is also chosen because of the same effect. Consequently, with the liftable rule removed from the theory, EITHER relearned the correct rule given 20 random training examples.

Intermediate concept utilization also allows EITHER to handle gaps at the very top of the theory as in normal constructive induction. For example, when the rule for cup is deleted, EITHER easily relearns it given 30 random examples.

5.3. Concept creation

The goal of concept creation is to simplify the inductively generated rules by making explicit the structure inherent in the revised rules. This process serves the twin purposes of compressing the rule base and identifying new intermediate concepts. The concept creation algorithm used by EITHER is based on the inverse resolution technique of Muggleton and Buntine [34]. In particular, EITHER uses the intra-construction, inter-construction, and absorption operators for compacting the revised rules.

5.3.1. The inverse resolution operators

In DUCE [33], sets of rules are compared in order to identify common patterns, and then combined and compressed using one of the inverse resolution operators. The inter-construction and intra-construction operators introduce new concepts in the process. The basic procedure is an iterative one in which operators are applied repeatedly until no further reduction of the theory is possible.

In the inter-construction technique, a single rule is formed to extract the common pattern associated with the input rules. For example, rules such as $x \leftarrow w \wedge y \wedge z$ and $x \leftarrow u \wedge y \wedge z$ are combined to form the rules $x \leftarrow w \wedge v$, $x \leftarrow u \wedge v$, and $v \leftarrow y \wedge z$, where v is a new intermediate concept.

In intra-construction, new rules are formed representing the differences between the input rules. For example, the same rules as above would be combined as $x \leftarrow v \wedge y \wedge z$ where $v \leftarrow w$ and $v \leftarrow u$ where v is again a new intermediate concept. Note that unlike inter-construction, intra-construction requires that both input rules have the same consequent. The choice of whether to use inter-construction or intra-construction is dependent on the syntactic simplicity of the resultant update.

Absorption occurs when all of the antecedents for one rule (e.g. $x \leftarrow a \wedge b$) are contained in the antecedents of another (e.g. $y \leftarrow a \wedge b \wedge c$).

The consequent for the smaller rule is inserted into the antecedents for the larger rule, in place of the antecedents which the two rules have in common (e.g. $y \leftarrow x \wedge c$). In the general case, absorption could happen even if there were many rules implying the consequent for the smaller rule, and the combination would represent a generalization to the larger rule. Since the basic revision algorithm guarantees consistency with the training set, EITHER only allows absorption when there is a single version of the absorbed rule (i.e. a single rule with the given consequent) so that the semantics of the rules are unchanged.

After EITHER produces a revised theory that is consistent with the training examples, the above operators are used to compress any rules that were modified or created during the revision. In the process, new intermediate concepts are created. The EITHER procedure is slightly different from the original one in DUCE in that it does not employ an oracle, does not actually generalize the input rules, and employs hill-climbing rather than best-first search in order to find a good operator to apply.

Let the original set of rules under consideration for rule reduction be given by

$$X_i \leftarrow A \wedge N_i \quad (1 \leqslant i \leqslant n),$$

where A represents the set of antecedents which are in common among all of the rules, and N_i represents the remaining antecedents for each rule. The objective in choosing A is to produce the greatest syntactic reduction. The computation of A uses a greedy algorithm and is done separately for inter- and intra-construction, since a different set of rules may be involved in each case. At each iteration, a new literal is chosen to add to A which causes the largest reduction in the input rules. If the reduction with the literal added is less than the previous reduction, the process halts. Once A has been computed for each case, the reduction operator that produces the greatest syntactic reduction is chosen. In case of ties, intra-construction is chosen since it focuses the reduction on rules having the same consequent. The overall process of applying operators continues until no further reduction is possible.

5.3.2. A concept creation example

As an example of intermediate concept creation, consider the case in which all of the rules for both liftable and graspable are deleted from cup theory. Given 50 random examples composed of 50% positive examples of *cups* and 50% near-miss negatives, EITHER initially learns the rules:[9]

[9]In order to make the formation of a concept for graspable cause a reduction in the number of literals in the theory, the feature lightweight was changed to a linear feature weight and the correct rule for liftable was changed to (liftable) ← (graspable) ∧ (weight ?w) ∧ (< ?w 1).

(liftable) ←(has-handle) ∧ (weight ?G0009) ∧
(< ?G0009 1.1257166),

(liftable) ←(insulating) ∧ (width small) ∧ (not (has-handle)) ∧
(weight ?G0009) ∧ (< ?G0009 1.1257166).

These rules are then reduced to:

(liftable) ← (intra-0010) ∧ (weight ?G0009) ∧ (< ?G0009 1.1257166),
(intra-0010) ← (has-handle),
(intra-0010) ← (insulating) ∧ (width small) ∧ (not (has-handle)).

The intermediate concept intra-0010 formed using intra-construction is EITHER's new concept for graspable. The extra (not (has-handle)) antecedent on the second rule is a side-effect of translating ID3 decision trees into rules. It does not affect the semantics of the new concept and could be deleted using the sort of rule simplification methods discussed in [42].

6. Computational complexity

Table 1 summarizes the results of the complexity analysis from [37]. In the table, n refers to the number of input examples, s refers to the size of the input theory, and b refers to the average number of rules for a concept (disjunctive branching factor). Clearly, the bottlenecks are the calculation of the partial proofs and possible proofs. In the case of the partial proofs, abduction is the primary reason for the complexity result, since it is exponential in the size of the theory [48]. However, heuristic methods are available for improving the efficiency of abduction by using beam search to explore only the k partial proofs with the fewest assumptions [36]. In addition, reducing the number of partial proofs would directly impact the minimum antecedent cover calculations at the potential cost of increasing the size of the eventual cover.

Table 1
Complexity results.

partial proofs	antecedent cover	rule generalization	rule compression
$O(sb^s)$	$O(sb^s \log s)$	$O(ns \log s)$	$O(s \log s)$
possible proofs	rule cover	rule specialization	rule compression
$O(sb^s)$	$O(sb^s \log s)$	$O(ns \log s)$	$O(s \log s)$

Computing all possible proofs remains an exponential problem. However, it has not proven to be a significant bottleneck in practice. Because the theory is nearly correct, in most cases there will not be many proofs of negative examples. Not only does this reduce the computation of producing

all possible proofs, it also reduces any processing downstream, notably the computation of the minimum rule cover.

These considerations indicate that converting the abduction algorithm to a method that provides a reduced set of partial proofs would be particularly useful. Another approach to improving efficiency is to only partially fit the theory to the training data. Existing experiments with a version of EITHER that computes only partial covers of the failing positive and failing negative examples have demonstrated that this technique can significantly increase efficiency without significantly affecting accuracy [32].[10] It should also be noted that while propositional Horn-clause theorem proving can be performed in linear time [10], the algorithm implemented in EITHER does not use the more efficient methods, making the deductive component another prime target for future improvement.

7. Experimental results

EITHER was tested on two domain theories to determine its ability to revise real expert rule bases using real data. The first of these, a domain theory for recognizing promoters in DNA sequences, constitutes a single-category theory as discussed in Section 3. The second, a theory for the diagnosis of soybean diseases, represents a multiple-category theory as discussed in Section 4. The results from both of these domains is discussed in the remainder of this section. Further information on these tests, including the actual initial and revised theories, is given in [37].

7.1. DNA promoter recognition results

EITHER was first tested on a theory for recognizing biological concepts in DNA sequences. The original theory is described in [52], it contains 11 rules with a total of 76 propositional symbols. The purpose of the theory is to recognize *promoters* in strings of nucleotides. A promoter is a genetic region which initiates the first step in the expression of an adjacent gene (*transcription*). The input features are 57 sequential DNA nucleotides. The examples used in the tests consisted of 53 positive and 53 negative examples assembled from the biological literature. The initial theory classified none of the positive examples and all of the negative examples correctly, thus indicating that the initial theory was entirely overly specific.

Figure 7 shows learning curves obtained when EITHER was used to refine this theory. In each test, classification accuracy was measured using twenty-

[10]Incomplete covering was originally developed to deal with noisy data as discussed in [32,37].

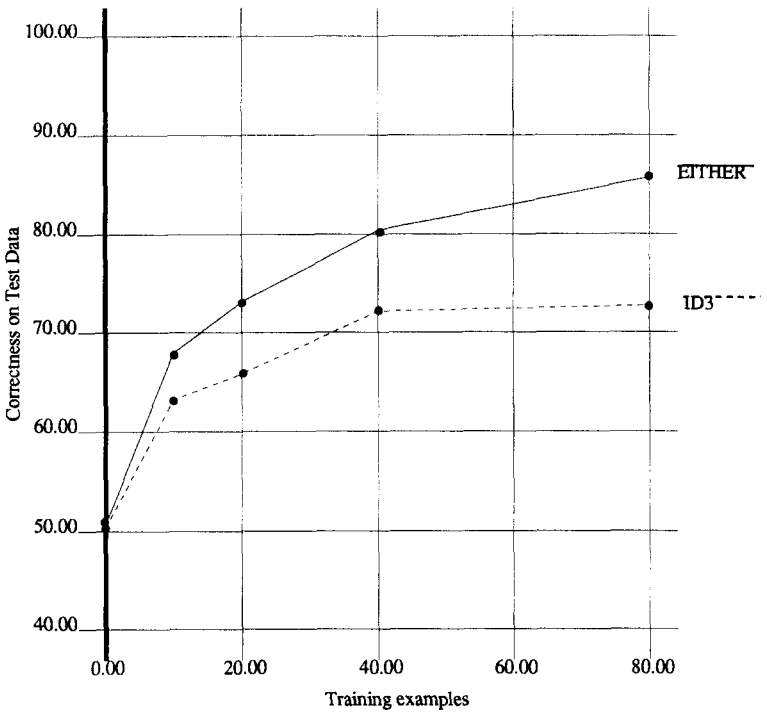

Fig. 7. Results for DNA promoter recognition.

six disjoint test examples. The number of training examples was varied from one to eighty, with the training and test examples drawn at random with no overlap. The results were averaged over 21 training/test divisions. ID3's performance is also shown in order to contrast theory refinement with pure induction.

The accuracy of the initial promoter theory is shown in the graph as EITHER's performance with 0 training examples and is no better than random chance (50%). With no examples, ID3 picks a category at random and exhibits the same accuracy. However, as the number of training examples increases, EITHER's use of the existing theory results in a significant performance advantage compared to pure induction. A one-tailed Student t-test on paired differences showed that the superior performance of EITHER compared to ID3 is statistically significant ($p < 0.05$) for every non-zero point plotted on the learning curves.[11] Overall, from no training to training with 80 examples, EITHER improves the accuracy of the theory by 35 percentage points. EITHER is also fairly efficient at revising the promoter domain theory. Its training time averages about 5 minutes on a Texas Instruments Explorer

[11]However, since we are running several independent t-tests, we cannot claim with 95% confidence that EITHER is *always* better for any number of training examples.

II Lisp Machine when run with 80 training examples.

An additional reason for including ID3 in the performance graphs is that it represents EITHER's performance without an initial theory, since in this case every example is a failing positive and induction would be used to learn a set of rules from scratch. Therefore including ID3's learning curve, provides a clear illustration of the advantage provided by theory-based learning. In fact, if a different inductive system were substituted for ID3, the absolute performance of both learning systems might change, but the relative advantage of EITHER compared to the purely inductive system should remain approximately the same.

Another way of looking at the performance advantage provided by an initial theory is to consider the additional examples required by ID3 in order to achieve equal performance with EITHER. For example, at 75% accuracy, ID3 requires over 60 additional training examples to achieve equal performance with EITHER. Therefore, in some sense the information contained in the theory is equivalent to 60 examples.

The revisions to the promoter theory primarily involved retracting antecedents (both for leaf rules and category rules) and generalizing antecedents. The results were compressed using both intra-construction and inter-construction, as discussed in section 5. In general, EITHER's changes made sense to the expert. In particular, it removed the intermediate concept conformation from the rule for promoter. This correction was validated by the biologist who encoded the theory (M. Noordewier), who indicated that conformation was a weakly-justified constraint when it was originally introduced.

EITHER's corrections to the rules clustered about the nucleotide positions associated with the original rules (that is, the tenth nucleotide position in the case of the minus_10 rules and the thirty-fifth nucleotide position in the case of minus_35 rules). This indicates that the original concept that promoter sequences are indicated by particular nucleotide configurations within certain *regions* of the nucleotide chain was valid, although the original rules themselves were overly specific.

This domain theory was also used to test the KBANN system [52], which translates the initial theory into an equivalent neural net, and then applies the backpropagation algorithm [46] to revise the network. KBANN performs somewhat better in this domain than EITHER (a test accuracy of 92% with 105 training examples). The likely explanation for the performance advantage is that promoter recognition seems to involve concepts of the form *M of these N features must be present.* Experiments comparing backpropagation and ID3 report that backpropagation is better at learning *M of N* functions [14]. Some aspects of the promoter concept fit the *M of N* format where, for example, there are several potential sites where hydrogen bonds can form between the DNA and the protein; if enough of these bonds form,

promoter activity can occur. On the other hand, EITHER attempts to learn this concept by learning a separate rule for each potential configuration by deleting different combinations of antecedents from the initial rules, which makes this a comparatively difficult learning task for a system using Horn clauses. Finally, it should be noted that when KBANN translated its results into Horn clauses, the resulting theory was significantly more complicated than EITHER's [51]. This is because EITHER's goal is to produce a minimally revised Horn-clause theory and KBANN has no such bias.

7.2. Soybean diagnosis results

In order to demonstrate EITHER's ability to revise multiple-category theories, EITHER was used to refine the expert rules given in [28]. This is a theory for diagnosing soybean diseases that distinguishes between nineteen possible soybean diseases using examples that are described with thirty-five features. The original experiments compared expert rules to induction from examples. By revising the expert rules to fit the examples, we hoped to show that one could produce better results than using just the examples or just the rules.

The original expert rules associated probabilistic weights with certain disease symptoms. In addition, some groups of disease symptoms were regarded as *significant* while other groups were regarded as *confirmatory*. The rules were translated to propositional Horn-clause format by only including the significant symptoms and by deleting any symptom from the theory that had a weight less than 0.8. After translation, the theory contained 73 rules with 325 propositional symbols.

Unfortunately, the classification performance of the Horn-clause version was seriously deficient compared to the original probabilistic rules. For example, the Horn-clause theory obtained a 12.3% classification performance compared to the accuracy of 73% reported in the original paper. To circumvent the problem, a "flexible" tester was used to classify the test examples, based on the updated theory provided by EITHER. The flexible tester accounts for two possible classification problems with the EITHER-generated theory. The first problem occurs when a test example is provable as a member of more than one category (that is, the theory is overly general with respect to the example). The second problem occurs when a test example is not provable as a member of any category (indicating the theory is overly specific with respect to the example).

With the standard EITHER tester, such examples are assigned to the most common category among the training examples. In contrast, the original soybean tests assigned a match score to each possible category and chose the category with the highest score. The flexible tester used by EITHER is a simple approximation to the original technique. If an example is assigned

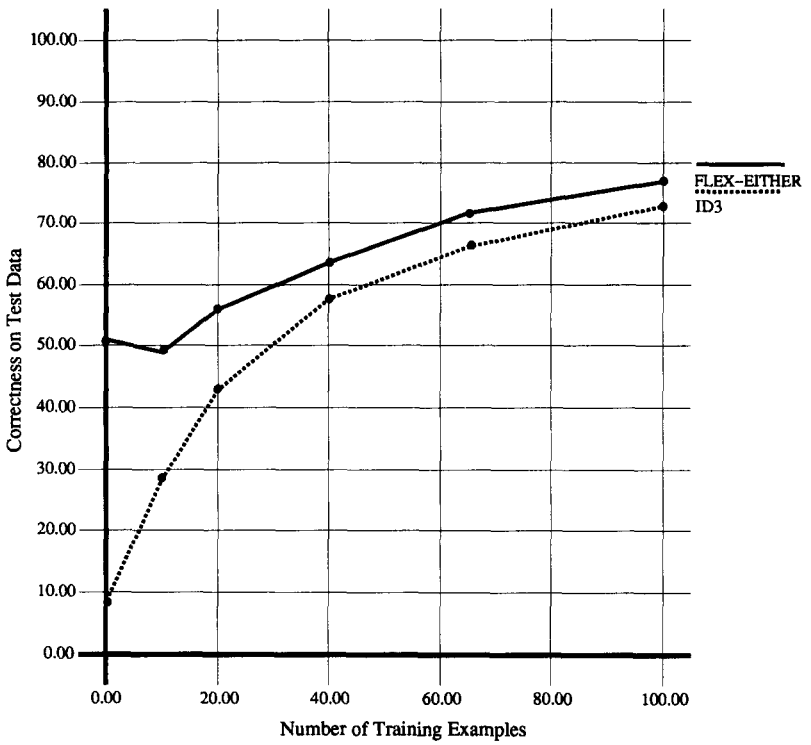

Fig. 8. Results for soybean diagnosis.

to multiple categories, the tester selects the most specific category that makes the most use of the example's features. This is done by choosing the category whose proof of category membership employs the greatest number of example features. If an example is not provable in any category, the flexible tester chooses the category that comes closest to being provable. This is done by choosing the category with a partial proof of category membership that has the least number of assumptions.

Learning curves for the soybean experiments are shown in Fig. 8. In each test, accuracy was measured against 75 disjoint test examples. The number of training examples was varied from one to one hundred, with the training and test examples drawn at random from the entire example population, with no overlap. Each point on the curves was computed from a 22-sample average. Note that even with the flexible tester, the accuracy of the original rules was only 51%, as compared to 73% for the original results presented in [28]. Overall, the accuracy of the initial rules is increased by 26 percentage points when EITHER is trained on 100 examples. Compared to pure induction, EITHER maintains its initial performance advantage over the entire training interval. A one-tailed Student t-test on paired differences showed that the superior performance of EITHER is statistically significant ($p < 0.05$) for every point plotted on the learning curves. Therefore, employing both expert-

provided rules and training examples is better than using either one alone.

The computational complexity of EITHER's revision algorithm is beginning to show in this domain. It takes an average of about 90 minutes on an Texas Instruments Explorer II Lisp Machine to fit the theory to 100 training examples. The relatively large number of categories, rules, and features in this domain makes revision computationally demanding.

Specialization and generalization were both required to correct the soybean theory. Typical modifications included removing antecedents at various levels in the theory, generalizing antecedents, inductively creating new rules, and inductively adding antecedents. The final rules were compressed using both inter-construction and intra-construction resulting in the formation of several meaningful new intermediate concepts representing the disjunction of several values of a particular feature or the conjunction of several related features.

8. Related research

Most previous systems for integrating explanation-based and empirical methods cannot refine arbitrarily imperfect theories. Some previous systems are only capable of generalizing an overly specific theory [8,50,53,54] while others are only capable of specializing an overly general theory [7,15,31]. Many systems do not revise the theory itself but instead revise the *operational definition* of a concept [3,21,40]. Still other systems rely on active experimentation rather than a provided training set to detect and correct errors [44]. Work in the related area of belief revision [2,11] focuses on resolving contradictions by retracting beliefs; however, it does not deal with generalizing a theory or specializing existing rules. Finally, previous systems do not have EITHER's modularity and therefore cannot easily take advantage of advances in the individual areas of deduction, abduction, and induction.

RTLS [17], KBANN [52], FOCL [40], DUCTOR [5], and a recent system by Feldman et al. [13] are theory revision systems that come the closest to handling as many types of imperfections as EITHER. Each of these systems is discussed in more detail below.

In the case of RTLS, a propositional Horn-clause theory is flattened into disjunctive normal form (DNF) prior to correction. Each category or intermediate concept in the theory has a *label* consisting of the terms in its DNF. The reduced theory is then modified to make it consistent with the training examples. Consequently, all corrections to the theory are done independently for each category. If there is an error in a shared intermediate concept, the error must be detected and corrected multiple times in the label for every category that uses the shared concept. EITHER, on the other hand, combines the evidence from all categories to revise a shared concept once and for all.

Once all of the labels are revised, RTLS attempts to translate the changes back into the original Horn-clause version of the theory. Limitations in this process prevent it from revising shared rules (what Ginsberg refers to as *non-eigen-terms*). Also, RTLS cannot deal with actual gaps in the theory (if there are no rules for proving a category or intermediate concept it cannot be deduced) nor create new intermediate concepts.

KBANN (Knowledge-based Artificial Neural Networks) is an approach to theory refinement that uses the backpropagation algorithm for multi-layer neural networks [46] as a method for correcting a domain theory. The technique first translates the existing domain theory into an equivalent neural network, then refines the weights in the network to fit the training examples. It then re-translates the corrected network into an approximately equivalent set of rules.

KBANN cannot deal with arbitrary gaps in the theory (where there are no rules for proving a category or intermediate concept), nor can it introduce new intermediate concepts. These problems could possibly be addressed by adding extra hidden units and connections to the initial network; however, this would require predetermining the number and type of intermediate concepts to be created. In addition, KBANN does not guarantee that the revised theory will be consistent with the training examples due to convergence problems associated with backpropagation. Finally, as discussed in Section 7.1, KBANN is not focused on minimally changing the existing theory.

FOCL (First-Order Combined Learner) is a hybrid system that uses FOIL [43] as its inductive component. It is capable of handling both incomplete and incorrect first-order Horn-clause theories. FOCL is based on the process of *operationalization* using a technique similar to that employed in MLSMART [3]. The system continually attempts to re-express higher-level concepts in the theory in terms of lower-level concepts until the goal concept is expressed in terms of observables. At each step, the system has a choice of using either the theory or induction to operationalize a concept, and it uses FOIL's information-theoretic measure to determine the best option.

Although FOCL works well with many types of incorrect theories, it does not handle certain problems very well. In particular if an intermediate concept is missing a rule for one of its disjuncts (such as missing one of the graspable rules in the cup theory), FOCL must learn a complicated rule at the top level of the theory instead of learning a simple rule directly for the intermediate concept. Also, the original FOCL system does not revise the underlying domain theory. KR-FOCL is a recent theory revision version [39]; however, it requires direct interaction with the user to determine which part of the theory to modify instead of using the complexity of the required change. Finally, FOCL cannot guarantee consistency with the training data since it uses hill-climbing and may encounter local maxima.

The DUCTOR is a recent EITHER-inspired system that integrates deduction, abduction, and induction. However, it does not generate all proofs and partial proofs and does not attempt to find a minimum cover of theory changes. Consequently, it is less focused on finding a *minimal* revision to the initial theory.

Feldman et al. [13] have recently developed a system for incrementally revising approximate domain theories. Their system also incorporates many ideas from EITHER; however, it focuses on revising theories in which rules and antecedents have been assigned numerical *belief values* representing how certain the user is in the various aspects of the theory. During revision, the system prefers to modify parts of the theory with lower belief values. This is a useful addition when such belief values are available; however, the user is frequently unable to provide such information.

9. Future research

Several promising areas for future research have been discovered during the development and testing of EITHER. Suggestions for improving EITHER's efficiency were discussed in Section 6. In this section, we discuss some additional problems with the current system, many of which are the subject of on-going research.

First, the current system cannot handle theories that employ *negation as failure*. Antecedents of the form not(P) complicate the revision process since generalizing or learning a rule that concludes P actually specializes the overall theory by preventing such an antecedent from being satisfied. Conversely, specializing or eliminating a rule for P may actually generalize the overall theory. Therefore, the system will have to consider standard generalization operators as specializers in certain contexts and vice versa.

Second, the current system assumes all examples are instances of exactly one of the top-level categories. It cannot directly accept examples of intermediate concepts nor deal with overlapping categories. A truly robust theory revision system should be able to accept examples of any of its concepts and use them to revise the rules for that concept directly or to revise other concepts indirectly.

A third obvious limitation is EITHER's restriction to propositional Horn clauses. This prevents the system from applying to domains requiring structural descriptions or relational predicates. Many ideas from EITHER are currently being incorporated in a new system, FORTE [45], which can revise theories expressed using first-order Horn clauses. FORTE also incorporates many ideas from the work on FOIL [43] and inverse resolution [34].

A fourth shortcoming in EITHER's knowledge representation is an inability to revise probabilistic rules. Many existing expert rule bases employ

some form of probabilistic reasoning. A first step in in dealing with probabilistic theories was the incorporation of a flexible tester, described in Section 7.2. The general complication that probabilistic reasoning introduces is that, when a system is considering a rule update, it must decide whether to update the probability associated with the rule, the rule itself, or both. Some previous work has addressed the problem of refining the probabilities or certainty factors attached to rules [18,24]; however, such numerical adjustments have not been integrated with more symbolic revisions such as learning new rules. We are currently developing a system that first "tweaks" certainty factors until no more improvement is possible and then resorts to learning new rules. The system cycles between "tweaking" and rule learning until it converges to 100% accuracy on the training data [25].

A final problem involves EITHER's commitment to a single inductive learning strategy, namely ID3. A more general approach would be to provide a variety of inductive learners, where the selection of a particular algorithm is dictated by the current problem. For example, it has been shown that neural networks are particularly suitable for learning concepts involving *M of N* functions [52]. In addition, case-based reasoning has been shown to be an effective adjunct to a rule-based system for exception processing [19]. Finally, when insufficient training data is available, some form of active knowledge acquisition, like experimentation, is required [44]. In each of these cases, using the basic EITHER algorithm to focus the knowledge acquisition should improve both ease of comprehension and accuracy of the knowledge base. The primary research issue is how to pick the appropriate inductive learner for a given problem.

10. Conclusions

A concise summary of the main results presented in this paper is:

> *Using explanations to focus inductive corrections to a domain theory results in a knowledge base which is more comprehensible and accurate than that which is obtained with standard empirical learning.*

Superior ease of comprehension is a result of making small changes to an existing theory. Consequently, the final knowledge base contains intermediate concepts that are already familiar to the domain experts. Empirical results on the DNA problem reported in Section 7.1 confirm that EITHER's revisions are frequently meaningful and acceptable to a domain expert.

Superior classification accuracy is a result of combining information from both background theory and empirical data instead of relying on only one of these sources of knowledge. Support for this hypothesis was provided by empirical results on revising two real expert rule bases (see Section 7). As demonstrated by the results on the DNA promoter problem, the use of an initial theory can provide an advantage even in the case where the initial theory is not able to correctly classify a *single* positive example. In addition, an examination of the changes made by the system in these cases show that the revisions correct multiple faults, correct and discover intermediate concepts within the theory, and are capable of correcting both specialization and generalization errors.

A worst-case complexity analysis shows that the EITHER algorithm is exponential in the size of the theory. Section 6 has shown that there are methods for reducing the complexity of the algorithm, provided that we are willing to relax the requirement for complete consistency between the revised theory and the training examples.

Acknowledgments

We would like to thank Mick Noordewier and Jude Shavlik for providing the DNA promoter theory and data and helping us interpret the results; Ryszard Michalski for providing the soybean diagnosis data; Jeff Mahoney for translating the soybean theory and data and implementing the flexible tester; and Hwee Tou Ng for providing the abduction component. We would also like to thank Bradley Richards for carefully reviewing a preliminary version of this paper. This research was supported by the NASA Ames Research Center under grant NCC 2-629, the National Science Foundation under grant IRI-9102926, and the Texas Advanced Research Program under grant 003658114.

References

[1] R. Bareiss, B.W. Porter and K. Murray, Supporting start-to-finish development of knowledge bases, *Mach. Learn.* **4** (3–4) (1989) 259–284.

[2] In: P. Gärdenfors, ed., *Belief Revision* (Cambridge University Press, Cambridge, England, 1992).

[3] F. Bergadano and A. Giordana, A knowledge intensive approach to concept induction, in: *Proceedings Fifth International Conference on Machine Learning*, Ann Arbor, MI (1988) 305–317.

[4] L.A. Birnbaum and G.C. Collins, eds., *Proceedings of the Eighth International Workshop on Machine Learning: Section on Learning From Theory and Data*, Evanston, IL (1991).

[5] T. Cain, The Ductor: a theory revision system for propositional domains, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 485–489.

[6] E. Charniak and D. McDermott, *Introduction to AI* (Addison-Wesley, Reading, MA, 1985).

[7] W.W. Cohen, Learning from textbook knowledge: a case study, in: *Proceedings AAAI-90*, Boston, MA (1990) 743–748.

[8] A.P. Danyluk, Finding new rules for incomplete theories: explicit biases for induction with contextual information, in: *Proceedings Sixth International Conference on Machine Learning*, Ithaca, NY (1989) 34–36.

[9] G.F. DeJong and R.J. Mooney, Explanation-based learning: an alternative view, *Mach. Learn.* **1** (2) (1986) 145–176.

[10] W.F. Dowling and J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *J. Logic Programm.* **3** (1984) 267–284.

[11] J. Doyle, A truth maintenance system, *Artif. Intell.* **12** (1979) 231–272.

[12] G. Drastal, G. Czako and S. Raatz, Induction in an abstraction space: a form of constructive induction, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 708–712.

[13] R. Feldman, A. Segre and M. Koppel, Incremental refinement of approximate domain theories, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 500–504.

[14] D.H. Fisher and K.B. McKusick, An empirical comparison of ID3 and backpropagation, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 788–793.

[15] N.S. Flann and T.G. Dietterich, A study of explanation-based methods for inductive learning, *Mach. Learn.* **4** (2) (1989) 187–226.

[16] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, New York, NY, 1979).

[17] A. Ginsberg, Theory reduction, theory revision, and retranslation, in: *Proceedings AAAI-90*, Boston, MA (1990) 777–782.

[18] A. Ginsberg, S.M. Weiss and P. Politakis, Automatic knowledge based refinement for classification systems, *Artif. Intell.* **35** (1988) 197–226.

[19] A.R. Golding and P.S. Rosenbloom, Improving rule-based systems through case-based reasoning, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 22–27.

[20] D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artif. Intell.* **26** (1988) 177–221.

[21] H. Hirsh, Incremental version-space merging: a general framework for concept learning, Ph.D. Thesis, Stanford University, Palo Alto, CA (1989).

[22] D.S. Johnson, Approximation algorithms for combinatorial problems, *J. Comput. Syst. Sci.* **9** (1974) 256–278.

[23] H.J. Levesque, A knowledge-level account of abduction, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1061–1067.

[24] X. Ling and M. Valtorta, Revision of reduced theories, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 519–523.

[25] J.J. Mahoney and R.J. Mooney, Combining neural and symbolic learning to revise probabilistic rule bases, in: *Advances in Neural Information Processing Systems* **5** (Morgan Kaufmann, San Mateo, CA, to appear).

[26] R.S. Michalski, Synthesis of optimal and quasi-optimal variable-valued logic formulas, in: *Proceedings 1975 International Symposium on Multiple-Valued Logic*, Bloomington, IN (1975) 76–87.

[27] R.S. Michalski, A theory and methodology of inductive learning, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 83–134.

[28] R.S. Michalski and S. Chilausky, Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis, *J. Policy Anal. Inf. Syst.* **4** (2) (1980) 126–161.

[29] R.S. Michalski, I. Mozetic, J. Hong and N. Lavrac, The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 1041–1045.

[30] T.M. Mitchell, R.M. Keller and S. Kedar-Cabelli, Explanation-based generalization: a unifying view, *Mach. Learn.* **1** (1) (1986) 47–80.

[31] R.J. Mooney and D. Ourston, Induction over the unexplained: integrated learning of concepts with both explainable and conventional aspects, in: *Proceedings Sixth International Conference on Machine Learning*, Ithaca, NY (1989) 5–7.

[32] R.J. Mooney and D. Ourston, Theory refinement with noisy data, Tech. Report AI91-153, Artificial Intelligence Laboratory, University of Texas, Austin, TX (1991).

[33] S. Muggleton, Duce, an oracle based approach to constructive induction, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 287–292.

[34] S. Muggleton and W. Buntine, Machine invention of first-order predicates by inverting resolution, in: *Proceedings Fifth International Conference on Machine Learning*, Ann Arbor, MI (1988) 339–352.

[35] H.T. Ng and R.J. Mooney, Abductive explanations for text understanding: some problems and solutions, Tech. Report AI89-116, Artificial Intelligence Laboratory, University of Texas, Austin, TX (1989).

[36] H.T. Ng and R.J. Mooney, An efficient first-order abduction system based on the ATMS, in: *Proceedings AAAI-91*, Anaheim, CA (1991).

[37] D. Ourston, Using explanation-based and empirical methods in theory revision, Ph.D. Thesis, Tech. Report AI 91-164, Artificial Intelligence Laboratory, University of Texas at Austin, TX (1991).

[38] D. Ourston and R.J. Mooney, Improving shared rules in multiple category domain theories, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 534–538.

[39] M. Pazzani and C. Brunk, Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning, in: *Proceedings Fifth Knowledge Acquisition for Knowledge-Based Systems Workshop* Banff, Alta. (1990).

[40] M. Pazzani, C. Brunk and G. Silverstein, A knowledge-intensive approach to learning relational concepts, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 432–436.

[41] J.R. Quinlan, Induction of decision trees, *Mach. Learn.* **1** (1) (1986) 81–106.

[42] J.R. Quinlan, Generating production rules from decision trees, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 304–307.

[43] J.R. Quinlan, Learning logical definitions from relations, *Mach. Learn.* **5** (3) (1990) 239–266.

[44] S.A. Rajamoney, A computational approach to theory revision, in: J. Shrager and P. Langley, eds., *Computational Models of Scientific Discovery and Theory Formation* (Morgan Kaufmann, San Mateo, CA, 1990) 225–254.

[45] B. Richards and R.J. Mooney, First-order theory revision, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 447–451.

[46] D.E. Rumelhart, G.E. Hinton and J.R. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, eds., *Parallel Distributed Processing, Vol. I* (MIT Press, Cambridge, MA, 1986) 318–362.

[47] A. Segre, ed., *Proceedings of the Sixth International Workshop on Machine Learning: Section on Combining Empirical and Explanation-Based Learning*, Ithaca, NY (1989).

[48] B. Selman and H.J. Levesque, Abductive and default reasoning: a computational core, in: *Proceedings AAAI-90*, Boston, MA (1990) 343–348.

[49] M.E. Stickel, A prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation, Tech. Report 451, SRI International, Menlo Park, CA (1988).

[50] G.D. Tecuci and R.S. Michalski, A method for multistrategy task-adaptive learning based on plausible justifications, in: *Proceedings Eighth International Workshop on Machine Learning*, Evanston, IL (1991) 549–553.

[51] G.G. Towell and J.W. Shavlik, Refining symbolic knowledge using neural networks, in: *Proceedings International Workshop on Multistrategy Learning*, Harper's Ferry, WV (1991) 257–272.

[52] G.G. Towell, J.W. Shavlik and M.O. Noordewier, Refinement of approximate domain theories by knowledge-based artificial neural networks, in: *Proceedings AAAI-90*, Boston, MA (1990) 861–866.

[53] B.L. Whitehall, Knowledge-based learning: an integration of deductive and inductive learning for knowledge base completion, Ph.D. Thesis, Tech. Report UILU-ENG-90-1776, University of Illinois, Urbana, IL (1990).

[54] D.C. Wilkins, Knowlege base refinement using apprenticeship learning techniques, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 646–651.

[55] P.H. Winston, T.O. Binford, B. Katz and M. Lowry, Learning physical descriptions from functional definitions, examples, and precedents, in: *Proceedings AAAI-83*, Washington, DC (1983) 433–439.