A Survey of Machine Learning Systems Integrating Explanation-Based and Similarity-Based Methods

Andrea Pohoreckyj Danyluk
Department of Computer Science
Columbia University
New York, NY 10027

(212) 854-2736

andrea@CS.COLUMBIA.EDU

12 October 1989

CUCS-467-89

Abstract

Two disparate machine learning approaches have received considerable attention. These are explanation-based and similarity-based learning. The basic goal of an explanation-based learning system is to more efficiently recognize concepts that it is already capable of recognizing. The learning process involves a knowledge-intensive analysis of an environment-provided example of a concept in order to extract its characteristic features. The basic goal of a similarity-based system, on the other hand, is to acquire descriptions that allow the system to recognize concepts it does not yet know. Although they have been applied with some success to problems in a variety of domains, both methods have clear deficiencies. Explanation-based learning assumes that a system will be provided with an explicit domain theory that is complete, correct, and tractable. This assumption is unrealistic for many complex, real-world domains. Similarity-based learning suffers because of its lack of an explicit theory. Since the two methods are complementary in nature, an obvious solution is to augment systems using one approach with techniques from the other. This survey discusses machine learning systems that integrate explanation-based and similarity-based learning methods such that one is incorporated primarily to handle a deficiency of the other. Although sufficient background material is provided that the reader need not be familiar with machine learning, general knowledge of AI is assumed.

Table of Contents

List of Figures

1 Introduction

Recently two disparate machine learning approaches have received considerable attention. These are explanation-based and similarity-based learning. The basic goal of an explanation-based learning system is to more efficiently recognize concepts that it is already capable of recognizing. The learning process involves a knowledge-intensive analysis of an environment-provided example of a concept in order to extract its characteristic features. The analysis is in the form of an explanation describing why the particular instance is a member of the concept to be learned. The basic goal of a similarity-based learning system, on the other hand, is to acquire descriptions that will allow the system to recognize concepts it does not yet know. Many examples of a concept are given to the system. These are compared against each other in order to find shared features that are assumed to define the concept.

Both explanation-based and similarity-based learning methods have been applied with some success to problems in a variety of domains. Both methods have clear problems, however. Explanation-based learning assumes that a system will be supplied with an explicit theory of the domain of application that is complete, correct, and tractable. This assumption is clearly unrealistic for most complex, real-world domains. Similarity-based learning suffers because of its lack of an explicit theory of the domain. The method requires that human intervention play a large role in tailoring input examples and the language describing them in such a way as to allow a system to choose a set of features to define a concept. Less tailoring of the examples leaves a system open to the possibility of not converging on the best definition for a concept, or any at all, due to the computational complexity.

Since the two methods are complementary, a natural solution to the problems encountered in explanation-based and similarity-based learning is to augment systems using one approach with techniques from the other. This survey discusses machine learning systems that integrate explanation-based and similarity-based learning methods such that one is incorporated primarily to handle a deficiency of the other. Specifically, it describes the use of similarity-based learning to handle inadequacies in explanation-based learning systems' domain theories, and the use of explanation derivation in giving similarity-based learning systems more freedom to select feature sets and appropriate representations.

Sections 2.1 and 2.2 introduce explanation-based and similarity-based learning methods. Section 2.3 compares the methods, discussing benefits and deficiencies of each. The remainder of the paper is devoted to the discussion of problems with the approaches, and systems that have been designed to address them. Section 3 discusses the augmentation of explanation-based learning systems with similarity-based learning in order to compensate for incompleteness (3.1), incorrectness (3.2), or intractability (3.3) of the relevant domain theories. Section 4 discusses the use of explanation derivation to direct the search of similarity-based learning for more appropriate representations and for the concepts they are to describe.

2 An Overview of Explanation-Based and Similarity-Based Learning

2.1 Explanation-Based Learning

Explanation-based learning (EBL)[1] (e.g., [DeJong 81; DeJong 83; DeJong and Mooney 86; Mitchell et al. 86; Silver 86; Winston et al. 83]) is a deductive machine learning approach in which a definition of a concept is derived, usually after observing only a single example of that concept. To understand this method on an intuitive level, consider a robot operating in a room containing household objects. In order to manipulate the objects in its world, the robot must have a mechanism for recognizing them. For example, it might require the ability to recognize cups.

Consider a robot that has been provided with a knowledge base of rules describing objects in its world and their characteristics. The rule base would be the domain theory as it constitutes the robot's representation of its domain of application. Some rules from this domain theory might be:

$$\text{LIGHT}(X) \wedge \text{PART-OF}(X,Y) \wedge \text{ISA}(Y,\text{HANDLE}) \Rightarrow \text{LIFTABLE}(X)$$

that states that if an object is light and has a handle then it is liftable, and:

$$\text{ISA}(X,\text{OPEN-VESSEL}) \wedge \text{STABLE}(X) \wedge \text{LIFTABLE}(X) \Rightarrow \text{ISA}(X,\text{CUP})$$

that states that if an object is an open vessel that is liftable and stable then it is a cup.[2] Now assume that the objects in the world are represented by predicates that describe basic structural features of the objects, such as FLAT and LIGHT. Because the domain theory relates these primitive predicates to higher level concepts such as CUP, it can be used by the robot to deduce that a particular object is or is not a cup. The deductive process, generally implemented by a theorem-prover, is computationally expensive, however. A major goal of explanation-based learning is to improve the performance of such systems.

Input to EBL is the name of the concept to be learned, an instance of it, and a domain theory to be used to prove that the input instance is an example of the given concept. In the robot domain described above, the concept to be learned might be ISA(X,CUP). The example of a cup might be the representation of a particular blue ceramic cup: a conjunction of primitives such as COLOR(obj1,BLUE), MADE-OF(obj1,CERAMIC), etc. The domain theory would include rules such as those given above.

---

[1] EBL is sometimes referred to in the machine learning literature as explanation-based generalization (EBG).

[2] These examples are modified from Mitchell *et al.*'s paper on a unifying framework for the method of explanation-based learning [Mitchell et al. 86]. All examples from this domain presented here are based upon examples from their paper.

The goal of EBL is to create a general concept description based upon the predicates describing the particular example, such that the general description is justified by the proof derived to show that the example was a member of the concept to be learned. The first step of the learning process is to use the domain theory to generate such a proof. The proof is sometimes called an explanation. The explanation for the input described in the preceding paragraph is shown in Figure 1.[3] Note that not all predicates used to describe the input example are actually found in the explanation. It is assumed that these, for instance the color of the cup, are not relevant to the definition of a cup.

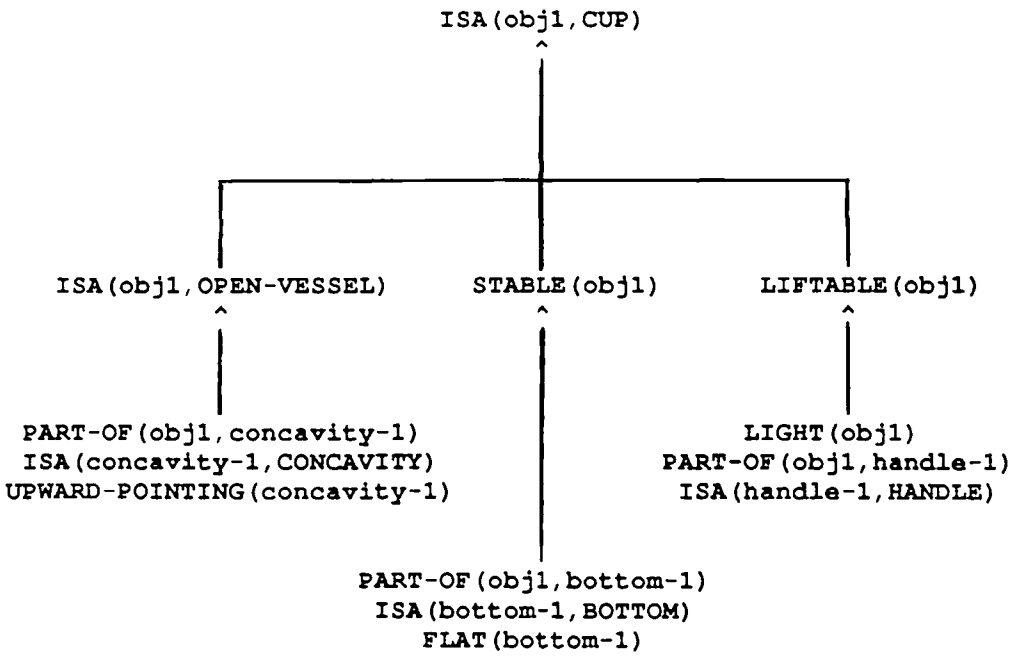

Figure 1: Explanation of Cup

The next step of EBL is to maximally generalize the description of the input example with the constraint that the derived proof structure still holds. Many constants appearing in the proof can be made variables as shown in Figure 2. In some cases the generalization must be constrained. For example, OPEN-VESSEL was not generalized.

The general description of a cup, deductively validated by the proof derived, is shown in Figure 3. It can be used by the robot to efficiently recognize cups, because the robot can now simply apply a single rule rather than having to construct a proof. (It has more recently been shown by [Minton 88] that there are cases in which the application of a single rule is less efficient than proof construction due to the complexity of determining that the rule applies.) In

[3]Modified from [Mitchell et al. 86], page 59.

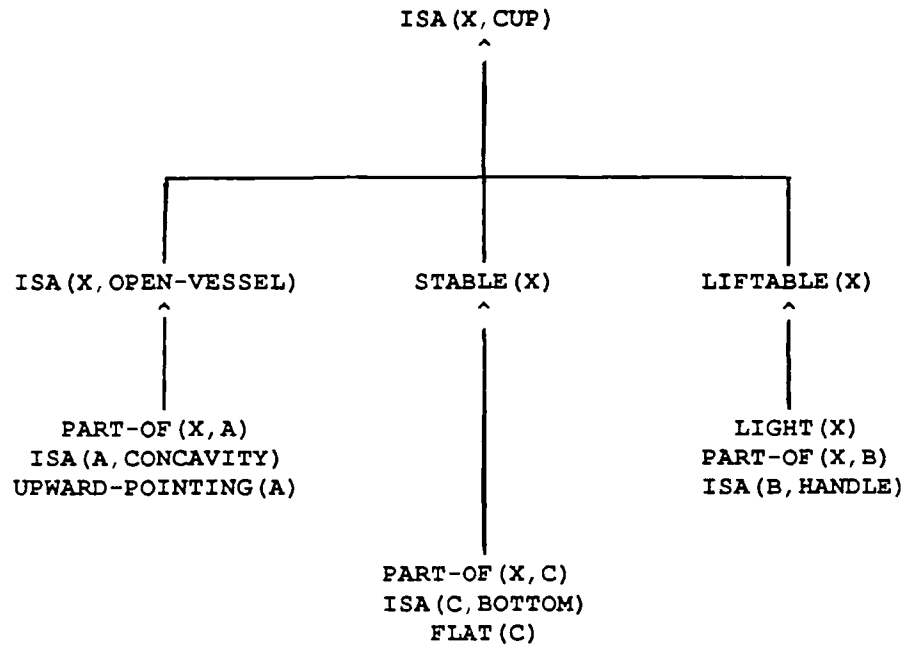

Figure 2: Generalized Explanation of Cup

domains such as that of solving algebraic equations, where proofs can resemble traces of problem solving steps, the generalized proofs may be stored to be applied automatically to analogous examples later. The result of explanation-based learning as described here is not to have learned a concept that could not have been recognized, but to learn a more efficient definition of such a concept.

$$\text{PART-OF}(X,A) \wedge \text{ISA}(A,\text{CONCAVITY}) \wedge \text{UPWARD-POINTING}(A)$$

$$\wedge$$

$$\text{PART-OF}(X,C) \wedge \text{ISA}(C,\text{BOTTOM}) \wedge \text{FLAT}(C)$$

$$\wedge$$

$$\text{LIGHT}(X) \wedge \text{PART-OF}(X,B) \wedge \text{ISA}(B,\text{HANDLE})$$

$$\Rightarrow \text{ISA}(X,\text{CUP}).$$

Figure 3: General Definition of Cup

The discussion above is intended to give an intuitive view of explanation-based learning. A more formal discussion is beyond the scope of this paper. (But see [DeJong and Mooney 86; Mitchell et al. 86]; a survey of EBL systems is presented in [Ellman 89].) The key ideas of EBL are: Explanation-based learning is a method in which a single example is analyzed and then generalized by a system. In order to perform the analysis, a system must possess a domain

theory that is complete and correct, and the process of analysis must be tractable. The analysis, or derivation of an explanation, will often be performed by a deductive inference mechanism such as a theorem-prover. Depending upon the domain, an explanation may be a deduction of the flavor in the example above, or, more generally, any transformation of an initial problem state to a goal state. Generalized explanations, in addition to learned general definitions, may be saved so that they need not be re-derived. A summary of EBL is given in Figure 4.

```
INPUT:   name of a concept to be learned;
         specific example of the concept;
         domain theory.

OUTPUT:  general (efficient) definition of the concept;
         generalized explanation.

METHOD:  1. derive a proof that the example is an instance of
            the concept.
         2. maximally generalize the proof while maintaining
            its correctness.
         3. extract a general definition from the generalized
            explanation.
```

Figure 4: Summary of Explanation-Based Learning

2.2 Similarity-Based Learning

Similarity-based learning (SBL) (e.g., [Fisher 87; Lebowitz 87; Michalski and Stepp 83; Mitchell 78; Quinlan 86; Winston 72]) is an empirical technique that involves comparisons among large numbers of input examples. The input examples are compared in order to find similarities and differences among them. Similarities are generally assumed to define a useful concept.

To get a flavor of the method, consider the robot world described in the preceding section. There is a robot operating in a room containing household objects. Again the robot must be able to recognize objects in this room, such as cups. Unlike in the EBL setting, this robot does not have a domain theory relating predicates describing basic structural properties of the objects to higher level concepts. It must learn the descriptions of the objects "from scratch".

Input to the learning algorithm is the name of the concept to be learned, for instance CUP, and examples of that concept. Additional input to the algorithm might be examples of objects that are not members of the concept. Sample input examples to an SBL system that is to learn the concept CUP are shown in Figure 5. Structural properties that are the same as those used in the preceding section are used to describe the inputs here. The goal of SBL is to create a general description of the concept that, so far as is possible, includes all the positive

examples and excludes all the negative examples. The most obvious algorithm for doing so is to compare the input and to find the largest set of descriptive features shared by all of the positive examples but not by any of the negative examples. A general description for the CUP concept based upon the input shown in Figure 5 is given in Figure 6.

FEATURE:	CUP-1	CUP-2	NON-CUP-1
COLOR	blue	red	blue
OWNER	Andrea	Anne	Harry
MADE-OF	ceramic	ceramic	ceramic
FLAT-BOTTOM?	yes	yes	yes
CONCAVITY?	yes	yes	yes
HANDLE?	yes	yes	no

Figure 5: Sample Input to SBL

	CUP
MADE-OF	ceramic
FLAT-BOTTOM?	yes
CONCAVITY?	yes
HANDLE?	yes

Figure 6: General Definition of Cup Found by SBL

The intuition behind the method is that the features essential for defining "CUP-ness" must appear in all examples of cups. The features that are not relevant to defining the concept, such as color, will vary among the examples and thus not be included in the general description. The version of SBL presented here is a simple-minded one. Many SBL systems deal with more complex representations, disjunctive definitions, and other generalizations of that presented above.

Though they do not possess the explicit domain theories of EBL systems, SBL programs do have an implicit bias built into them that allows them to arrive at particular general concept definitions [Mitchell 80; Utgoff 86]. This bias includes, among other things:

- the language in which input examples and the learned concepts are represented.
- concept hierarchies that define relationships among the values that can be taken on by features.

Bias built into SBL systems is called inductive bias, because the generalization performed by such systems is analogous to that of inductive reasoning, or the reasoning from particular instances to a general conclusion.

The example of similarity-based learning presented here belongs to just one variety of the methods called SBL. Although these methods vary, they share the intuition that the discovery of commonalities among members of a class may lead to a description for the class that will have predictive power in analyzing future examples. Most generally, SBL systems may be divided into two categories based upon whether or not examples provided to the system have been classified by an external source.[4]

Those methods that receive input classified by an external source are termed **learning from examples**. Work in this area includes that of [Mitchell 78; Winston 72] among others. These systems may be divided on the basis of the types of examples input. Some are given both positive and negative examples of a concept, while others receive only positive examples. Another distinction may be made on the basis of whether all examples are seen simultaneously or whether a description is built incrementally by considering one example at a time.

Those methods that receive unclassified input are referred to as **conceptual clustering**. Typically, many examples of a number of concepts are presented to a conceptual clustering system simultaneously. The system creates sets from the input examples such that the elements of each set share a number of descriptive features not shared by the members of any other set. Work in this area includes that of [Fisher 87; Lebowitz 87; Michalski and Stepp 83].

As with the overview of explanation-based learning above, this survey does not present a formal treatment of similarity-based learning. A summary of the learning from examples style of SBL is given in Figure 7.

```
INPUT:   name of the concept to be learned;
         specific examples (positive and/or negative)
           of the concept.

OUTPUT:  general definition of the concept.

METHOD:  compare examples to find similarities among positive
         examples and differences of positive examples from
         negative.
```

Figure 7: Summary of Learning from Examples

[4]This description of SBL has been modified from a taxonomy of learning methods given in [Machine Learning, Volume I 83].

2.3 A Comparison of Explanation-Based and Similarity-Based Learning

Explanation-based and similarity-based learning methods are approaches that can be placed on opposite ends of a spectrum describing purely deductive to purely inductive techniques [Mitchell 84]. In explanation-based learning, analysis of a single example guides the generalization of knowledge possessed by a system in order to make itself more efficient. Its power derives from having extensive domain knowledge. The dependence upon the domain theory is also the weakness of the method. The theory is assumed to be complete and correct and the explanation derivation process to be tractable. These assumptions are unrealistic in many complex, real-world domains. Similarity-based learning systems do not possess the extensive, explicit knowledge bases of EBL. They must instead have extensive explicit example bases which guide them in a search through all possible concepts representable in a given language. Problems may arise if the inductive bias is not sufficiently strong guiding the search or if the input data is noisy. A more fundamental problem for both learning methods is the possibility that the concept to be learned cannot be adequately represented in the language of the system.

A natural solution to the problems of incomplete, incorrect, and intractable domain theories for EBL is to augment such systems with SBL, which works in the absence of a strong domain theory. Similarly, the deductive reasoning from an explicit domain theory of EBL may be added to SBL systems to prune the search space for interesting similarities or to transform one representation language into another. The following sections describe in more detail potential problems of EBL and SBL systems. Each section contains case studies of approaches that involve the integration of explanation-based and similarity-based learning methods to address the problems. Examples illustrating each approach are taken from the domain to which it has been applied. Some of the methods address multiple deficiencies of EBL and SBL. Each of these is discussed in detail with respect to only one of the problems it addresses.

3 Using Similarity-Based Learning to Overcome Unrealistic Assumptions of Explanation-Based Learning

3.1 The Problem of Incomplete Theories

Rajamoney and DeJong [Rajamoney and DeJong 87] identify two ways in which domain theories used by EBL systems can lack knowledge. In the first, an explanation, or proof, cannot be completed by the EBL system because knowledge, usually in the form of a rule, is missing from the domain theory as illustrated in Figure 8. In order to complete the explanation, a rule or fact must be added to the domain theory. In the second case, proofs can be constructed leading to a conclusion, but they lack the detail required for a particular application. This problem may be seen as one of a wrong choice of granularity of the knowledge represented. Here Rajamoney and DeJong's second description of incompleteness is treated as incorrectness of the domain theory. This section discusses four approaches to the

problem of missing rules.

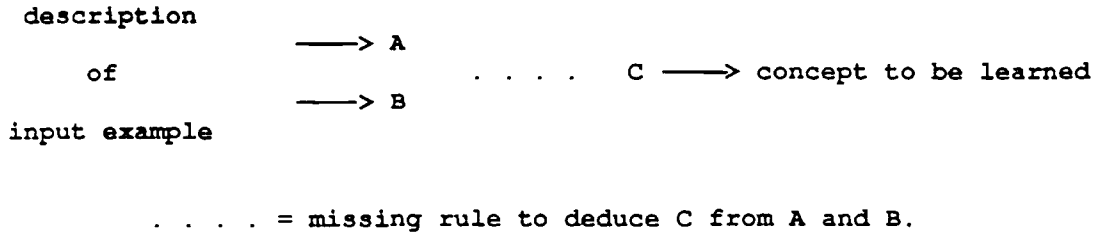

Figure 8: The Problem of Incompleteness

3.1.1 Rajamoney

Rajamoney [Rajamoney et al. 85; Rajamoney 88; Rajamoney 89] proposes a method called **experimentation-based theory revision**, implemented in the ADEPT system, as a solution to the problem of incompleteness in a domain theory. Recently, Michalski and Ko [Michalski and Ko 88] have also discussed the use of experimentation in addressing this problem. The discussion here focusses on Rajamoney's work.

ADEPT's domain theory is represented using Forbus's Qualitative Process Theory [Forbus 84]. According to this theory, changes in the world such as boiling or evaporation are due to *processes*, which in reasoning may be used analogously to rules. Processes are composed of three parts:

1. individuals - a set of objects that participate in the process,
2. preconditions and quantity conditions - a set of conditions that must be satisfied if the process is active,
3. relations and influences - a set of statements about the world that must be true if a process is active.

Preconditions and quantity conditions are analogous to the *if* part of an *if-then* rule, while relations and influences are analogous to the *then* part. ADEPT's goal is to use processes in explaining viewed changes in the world.

Rajamoney presents three ways in which incompleteness can become evident in a domain theory of processes:

1. an active process, i.e., one whose preconditions are satisfied, has flawed influences and is, in fact, causing the phenomenon that ADEPT is trying to explain. Essentially, this means that the phenomenon should be in the *then* part of an existing active rule (or should be deducible from a chain of active rules), but is not.
2. an inactive process that could explain the phenomenon if active has flawed preconditions or quantity conditions and should be active. In other words, a rule in which the phenomenon appears in the *then* part should have fired, but did not. Its preconditions, or *if* part, must be modified so that the rule may become

active.

3. a new process is causing the phenomenon. No modification of an existing rule is sufficient to complete the explanation.

ADEPT proposes new rules when it is unable to complete an explanation for a viewed change in the world. The types of rules proposed reflect the first two of the descriptions of inadequacy just discussed. That is, an existing active rule can be modified to include the viewed change in its *then* part, or an existing inactive rule can be modified so that its preconditions are satisfied. These constrain the number of rules that are proposed. Proposed rules are empirically tested for validity. The testing performed by ADEPT is not SBL per se, but rather empirical validation. It is included, however, because the underlying premise of SBL is empirical validation. That is, SBL makes the assumption that patterns observed over many examples will continue to be observed. ADEPT assumes that if the proposed rules accurately explain other observed changes in the world then they will continue to apply.

In order to minimize dependence upon a user, ADEPT tests its proposed rules by designing experiments so that the results of some of them will be inconsistent with a number of the proposed rules. These rules are eliminated from consideration. In an example taken from [Michalski and Ko 88], a system is given the task of explaining why a wine bottle placed in a freezer shattered. In the absence of more specific information, the system might propose that cold causes glass to contract while the volume of the contents of the glass remains the same. Alternately the liquid might expand. An experiment could be devised in which a glass container of water was placed in a freezer. If the container did not break, the first proposed rule would be invalidated.

Presumably ADEPT must possess a theory of experiment design for the domain to which it is applied. Given that good experiment design is a non-trivial problem, such a theory must be complex and would therefore be subject to the problem it is supposed to address, that of incompleteness. Furthermore, since the experiments would not exhaustively test all situations, a rule accepted by the system as appearing to hold in the experimental case might be incorrect. For instance in the example given above, the container might break despite the fact that the proposed rule is wrong.

A positive aspect of the work is that the number of newly proposed rules is constrained. This is important because it would be computationally infeasible to test exponential numbers of rules for validity. It is not clear, however, that the rule proposition mechanism is constrained sufficiently.

3.1.2 Hall

Hall describes a method for learning new rules called **Learning by Failing to Explain** [Hall 86]. In this method, a system that fails to find an explanation for an input example is given a new, analogous example by a teacher. The system learns by analyzing the analogue and comparing its explanation using similarity-based methods to the incomplete explanation.

Hall's system works in the domain of logic circuit design. Given as input a function name and an implementation of that function as a logic circuit, the system's goal is to prove that the implementation given is correct. Rules in the domain theory of the system have the form LHS ⇒ RHS, where LHS denotes a function, such as *PLUS*, and RHS is the description of an implementation for the LHS. The RHS may refer to other functions as well as to specific circuit implementations. For example, as shown in Figure 9, a function F0 might be decomposed into three subfunctions F1, F2, and F3, which might be implemented by circuits x, y, and z, respectively. The tree given in the figure is taken to be a proof that the implementation is correct. Rules used to construct the proof might be:

```
F0 ⇒ F1 F2 F3
F1 ⇒ x | q | r | s , where | indicates disjunction
F2 ⇒ y
F3 ⇒ z
```

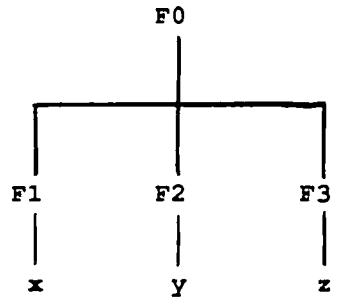

Figure 9: A Proof Tree for Logic Circuits

In Hall's system an explanation failure arises when the system is unable to prove that an input structure implements a given function. In the above example, this would occur if the system were missing the rule F1 ⇒ x | q | r | s as depicted in Figure 10. When the system signals a failure, a teacher provides an example from which the system can learn the rule it needs. It is assumed that in the failed proof the system is unable to link at most one subfunction to structures within the given implementation. That is, at most one rule is missing from the proof and it directly links the unexplained subfunction to features of the implementation.

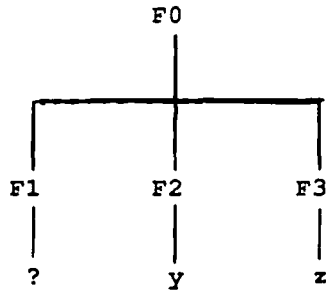

Figure 10: A Failed Proof for Logic Circuits

The teacher gives the system a new input structure that:

1. implements the same function as that given to the system initially and
2. contains as a subfunction the one for which the proof could not be completed.

A teacher-provided analogue to the example given above might be one for which a proof is given in Figure 11. The system derives as complete an explanation as possible for the new input. Assuming that all rules linking a specific circuit implementation to subfunction F1 are missing, a partial proof for the analogue would look like that in Figure 12. Next, corresponding parts of the two implementations are matched, where matching is defined as functional equivalence - in essence, the proofs are matched. In this example, the system might determine that F2 and F4 are functionally equivalent, as well as F3 and F5. The system assumes that corresponding unmatched parts are equivalent to each other, i.e., that they both implement the same function. In the example above, a and y match since F2 and F4 are functionally equivalent; similarly, b and z match. The yet unmatched parts, q and x, are assumed to both implement the unproved function F1. Rules corresponding to the separate implementations are created and generalized. For the example above, the system would learn the rule: F1 $\Rightarrow$ x | q.

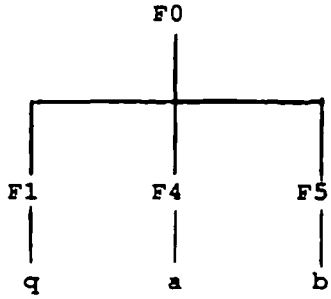

Figure 11: Proof for a Teacher-Provided Analogue

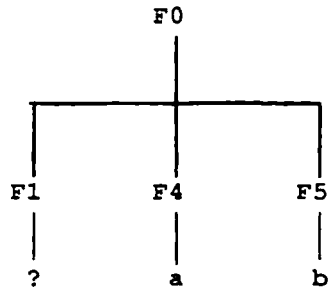

Figure 12: Partial Proof for Teacher-Provided Analogue

Hall's system works under the optimistic assumption that its domain theory will be very nearly complete. This must be the case if at most one subfunction may be unproved in the explanation of an entailing function. Hall's method is dependent upon a user not only for this nearly complete theory, but for training examples from which to learn, which must be specially tailored to the failure situation. Fortunately the input provided by the teacher does place strong constraints on the number of new rules proposed by the system. Recall that in Rajamoney's work the number of new rules proposed, although somewhat constrained, could be potentially large.

3.1.3 Pazzani

Pazzani, in his system OCCAM [Pazzani et al. 86; Pazzani 87; Pazzani et al. 87; Pazzani 88], which predicts and explains the outcome of events, uses general knowledge about causality to propose cause/effect rules to be added to an incomplete domain theory. Pazzani states a clear preference for knowledge-based methods over empirical ones. He believes that EBL is always to be preferred to SBL and that a system should only fall back on SBL as a last resort. In addition to the domain theory used by EBL, he provides OCCAM with a base of *generalization rules* that function as templates for new rules to be created.

Input to OCCAM is the description of an event. OCCAM's goal is to predict an outcome for it. The chain of reasoning from the description of the event to the predicted outcome is a proof that the outcome will occur. A failure occurs if a proof cannot be derived that links aspects of the event to a predicted result. When this happens, OCCAM instantiates a generalization rule that will complete an explanation. Generalization rules encode information reflecting a theory of causality, but are otherwise independent of a domain. In theory they could be used by a performance system that predicted the outcome of meteorological events as well as by another that predicted the outcome of chemistry experiments. A typical example of a generalization rule is: "If an action on an object precedes a state change for the object, then the action may cause the state change." Given two examples, one in which a balloon could not be blown up and one in which it was first stretched and then blown up, OCCAM would propose a rule that stated that stretching a balloon causes it to be in a state from which it can be blown

up. Rules are tentatively proposed by OCCAM as the generalization rules are not guaranteed to be instantiated correctly in all cases. For example, if a balloon were placed in water before being blown up, OCCAM would propose a rule that stated that dipping the balloon in water caused it to be blown up later. Rules are validated empirically as more examples are seen by the system. Thus, at any time, OCCAM's rule base may be incorrect.

A major strength of OCCAM is that the rules proposed by the system are constrained by the base of generalization rules which are applicable across a number of domains. However, they are specific to domains involving causality. It is not clear that an analogous base of rule templates could be devised for, say, Hall's logic circuit domain. Unlike Hall's system, however, Pazzani's is not dependent upon the presence of a teacher.

3.1.4 Kodratoff and Tecuci

DISCIPLE [Kodratoff and Tecuci 87; Kodratoff 87] uses an incomplete domain theory to pose questions to a user who then teaches the system missing rules. DISCIPLE has been applied to the domain of designing technologies for the manufacture of loudspeakers. A typical explanation is a plan for the manufacture of a loudspeaker. When the system is unable to complete a plan because knowledge is missing from the domain theory, a user supplies a rule that will complete the plan. DISCIPLE uses a combination of explanation-based and similarity-based learning to generalize that rule, thereby making it more widely applicable.

DISCIPLE begins the process of rule generalization in explanation-based mode, trying to explain why the rule given by the user is valid. Suppose that during the phase of constructing a plan, DISCIPLE had needed to *ATTACH sectors ON chassis-membrane-assembly*. Suppose that a user provided the following as a method to achieve that goal: *APPLY mowicoll ON sectors, PRESS sectors ON chassis-membrane-assembly*.[5] It is not assumed that the domain theory can provide a complete explanation, but only that it contains enough information to propose partial explanations. A best partial explanation is selected by the teacher. In the example described above, a user selected the explanation labelled A in Figure 13 as the best explanation proposed by DISCIPLE. Next DISCIPLE enters an SBL phase in order to generalize the rule. It searches the domain theory for information that matches the graph edges of the explanation selected. For example, the partial explanation labelled B in Figure 13 was extracted from DISCIPLE's domain theory as being similar to the first partial explanation. Before the matching and generalization of SBL can proceed, the user must validate all examples as being similar and relevant in the current context.

Kodratoff and Tecuci's system relies heavily upon interaction with a teacher, who gives the system the information it is missing. Rather than counting on the teacher to provide a

[5]From [Kodratoff 87], page 271. *Mowicoll* is roughly translated as *Romanian glue*.

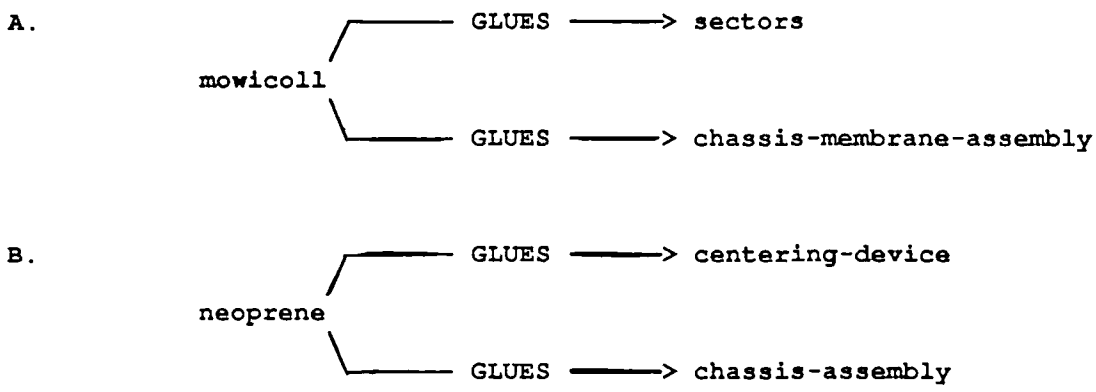

Figure 13: Partial Explanations in DISCIPLE

general rule, however, it asks for one appropriate for a particular situation and uses explanation-based and similarity-based techniques to guide the teacher in providing it with information sufficient for generalization of the rule. Constant supervision by the teacher can stop DISCIPLE from wasting computational resources considering irrelevant examples and assures correctness of the learned rules. This, however, places a heavy burden on the teacher.

3.1.5 Summary

This section described four approaches to the problem of finding new rules for an incomplete domain theory. Kodratoff and Tecuci take a conservative approach in DISCIPLE, which relies heavily on a user both for proposing and verifying new rules. Hall likewise relies on a user to input an analogue to a failure situation from which the system can learn. But his system is left to analyze the analogue as well as to generate a new rule from it. Both Pazzani's OCCAM and Rajamoney's ADEPT are independent of a teacher during the learning process, but they rely on the existence of auxiliary theories to guide them in creating new rules. ADEPT needs a theory of experimentation, while OCCAM has one of causality.

3.2 The Problem of Incorrect Theories

One manifestation of the incorrect theory problem of explanation-based learning was introduced in the previous section on incompleteness. Incorrectness may be the result of insufficient detail in the domain theory. This occurs when the designer of the theory makes assumptions about knowledge that may be approximated. The use of such a theory may result in mutually inconsistent explanations as depicted in Figure 14. A second type of incorrectness, probably the more basic one, is that in which wrong knowledge exists in the theory and must be retracted.

The following sections discuss three approaches to the problem of incorrectness. Araya and Silver each address the first manifestation of incorrectness; Dietterich and Flann address

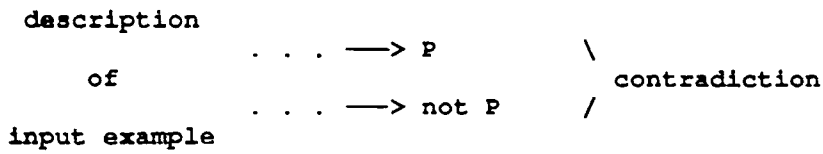

Figure 14: The Problem of Incorrectness

the second.

3.2.1 Araya

[Araya 84], in a system that learns in order to solve elementary physics problems, considers the task of starting with a domain theory that is incorrect in the sense of encoding knowledge at the wrong level of detail. The knowledge is sufficient to generate solution traces, or explanations, for all problems, but may not generate solutions that are optimal in the number of steps. The system has an auxiliary theory that allows it to simplify a solution trace. Simplification here is defined as the elimination of one or more equations employed by the general method that are not needed. The learning objective is to find a subclass of problems for which the simplified solution holds.

Using an idea similar to Rajamoney's, Araya employs an empirical approach in which the system tries to determine descriptions of problem classes by performing experiments and comparing their results. Variations of the original problem are proposed. A description of a problem class in Araya's system is represented in first-order logic. Variations on a problem are created by generating false variations of predicates that are true for the original problem solved. This is done using background knowledge associated with each predicate. To constrain the number of variations proposed, a fixed priority is used to determine the order in which specific kinds of predicates are considered. In the elementary physics domain, the priority specifies first looking at predicates referring to relations between positions of objects, then at predicates referring to relations between attributes of objects, and finally to those referring to attributes of objects.

Next each problem variation is solved. That is, a proof is derived linking the description of the problem to the solution. The variation is classified as a positive or negative instance for input to SBL. A problem variation is classified as a positive instance if its solution has the same extraneous steps as the original solution trace; otherwise it is classified as a negative instance. SBL is used to induce a general subclass of problems from the positive and negative instances that describes situations in which extraneous problem solving steps can be eliminated.

A number of possible problems exist with the approach proposed by Araya. First, an auxiliary theory is needed in order to devise simplified versions of problem solving traces. It is questionable that this auxiliary theory can be assumed to be correct when the main domain

theory is not. Second, information must be associated with each possible predicate in order for problem variations to be proposed. This requirement places an extra burden on the programmer of the system, especially when large numbers of predicates exist. The SBL component of the system searches for similarities in extraneous steps found in solution traces. Although this appears to be generally applicable to all equation solving domains, not just that of elementary physics problems, it is not clear that it can be generalized to any other type of domain. Finally, the method described here is very similar to that used in LEX [Mitchell 83].[6] It is not clear that it extends the work beyond the earlier LEX system.

3.2.2 Silver

Silver's LP [Silver 88] uses empirical methods to help a performance system work as well as possible with potentially incorrect information provided to it by EBL. LP's performance domain is mathematical equation solving. The system uses EBL to learn general problem solving techniques. Essentially, it develops problem solving schemata for general classes of equations. The system's domain theory is not assumed to be correct. The schemata learned will be correctly applicable in many situations, but may occasionally fail. The approach devised by Silver to address this problem is to retain the domain theory in its incorrect form, but to allow problem solving experience to override instantiations of the learned schemata.

The basis of the empirical process is a simple voting system. Votes are updated when LP finishes an attempt to solve a new equation. If a schema is successfully used to solve an equation, LP increases the future vote for the use of that schema on equations of the same type as that one just seen. If a schema is used unsuccessfully, i.e., if it is applied but does not lead LP to a solution, LP decreases the vote. Such a mechanism allows LP to eventually stop applying learned schemata to situations in which they probably will not work.

The other approaches to solving the incorrect theory problem discussed here involve using auxiliary knowledge to detect problems and to correct the theory. Silver takes a more optimistic approach to the problem. As he assumes that the theory will in most cases be correct, it might not be worthwhile to expend computational resources to change it. Instead, a simple empirical technique lowers the credibility of learned schemata if they are found to fail in situations where the theory implies that they should be applicable. If a schema fails a sufficient number of times, it will be considered unusable and will, therefore, be essentially unlearned.

Although the simplicity of Silver's approach is appealing, it cannot be applied efficiently with domain theories that are not largely correct. It is not desirable to have to unlearn most of the schemata proposed by EBL, especially since Silver assumes that LP has a default mechanism for solving all equations seen anyway. Although schemata are learned to save LP

---

[6] LEX is discussed in more detail in Section 4.1.3.

problem solving time, they might be better ignored entirely if they don't generally work.

3.2.3 Dietterich and Flann

Dietterich and Flann [Dietterich and Flann 88] identify a method, called **Induction Over Explanations (IOE)**, that allows incorrect knowledge to be removed from a system's domain theory. IOE requires that several training examples be provided to a system by a teacher. For each training example, the domain theory is used to construct all possible explanations. The derived explanations are input to SBL. For example, consider the domain of medical diagnosis in which a system is asked to determine what causes lung cancer. A training example might be a patient with lung cancer, including information about that person's general health and substances to which their lungs have been exposed -- such as the scent of flowers. The system creates all explanations linking potential causes to the patient's having cancer. Some of these explanations may be incorrect, say in the case that the domain theory states that smelling flowers causes lung cancer.

When all explanations have been derived, IOE prunes the domain theory such that at least one explanation exists for every positive example and no explanation may be derived for a negative example. In the case of the lung cancer examples, the system should not be able to conclude that a healthy person has cancer. This is accomplished by comparing explanations. Rules from the domain theory that are found in explanations of positive examples are retained. Rules shared by the explanations of negative examples but not found in the positive are deleted from the theory. When rules required to prove the negative instances have been deleted, the negative cases can no longer be proved.

Flann and Dietterich's IOE approach is commendable in that it attempts to make changes to a domain theory. A number of potential problems exist with the method of IOE, however. It requires that all explanations for a given input example be generated. It may be the case, in practice, that the number of explanations is small, yet this is not necessarily true for all domains. Furthermore, the matching of explanations is not a straightforward process and could prove to be computationally prohibitive.

Although Dietterich and Flann discuss the use of both positive and negative examples, their current implementation of IOE learns from positive examples only. It is possible, given that the algorithm requires only that one explanation hold for each input example, that correct rules will be eliminated from the domain theory. For example, consider a case in which ten examples are input to IOE. Say that all are examples of patients with lung cancer and that all patients have been exposed to cigarette smoke. Also assume that one of these patients was additionally exposed to toxic fumes. Rules might exist in the domain theory that would describe both cigarette smoke and toxic fumes as potential causes of lung cancer. Since only enough rules need to be retained to explain the input in one way, the rule specifying cigarette smoking might be kept while that referring to toxic fumes might be eliminated. This would

clearly be incorrect. If negative examples were seen in which the rule about toxic fumes did not play a role, the system could more accurately assume that its existence in only a single positive example was not problematic. Alternatively, the system could wait for more examples, presumably encountering one in which there was no reference to cigarette smoke but only a reference to toxic fumes.

3.2.4 Summary

This section described three approaches to the problem of detecting and compensating for incorrectness in a domain theory. Araya and Silver consider working with a domain theory that has been encoded at the wrong level of granularity. Araya uses an auxiliary theory to convert an explanation (or problem trace, in his case) into a more efficient version. He then uses empirical methods to find cases to which the simplifications can be applied more generally. Silver conjectures that if general schemata formed by EBL are rarely incorrect then the theory that generated them need not be corrected. The schemata can simply be ignored (or unlearned) if they are not found to work well.

Dietterich and Flann consider the problem of using a theory that contains incorrect information. They remove the incorrect information, or rules, by treating them as features in an inductive learning problem.

3.3 The Problem of Intractable Theories

A problem arises in explanation-based learning when the deductions that the system might make are computationally prohibitive and cannot be completed [Mitchell et al. 86]. While constructing an explanation the system exceeds the resources (time, memory, etc.) allotted to it. This is called the problem of intractability of the domain theory.

The knowledge represented in the domain theory of an EBL system may be viewed as a collection of *if-then* rules. The process of constructing the proof that the input example is a member of the concept to be learned is one of chaining together these rules such that the description of the example is linked to the concept. Given this, two different kinds of intractability can be encountered [Rajamoney and DeJong 87]:

- Large Search Space Problem - This occurs when the number of rules in the domain theory that *appear* to be relevant in deriving a proof is large. Finding the appropriate subset of rules that actually link the input example to the concept might involve exploring paths that appear promising but that do not lead to the creation of a proof. If explanation generation is viewed as search, then in the worst case the size of the space explored is D^B, where D is the maximum length of an explanation and B is the maximum number of rules applicable at any point. The large search space problem occurs when D^B is large, and is characterized by large values for B.
- Small Links Problem - This occurs when the number of rules required to form a single proof is too many for the system to construct the complete proof within the allotted resources. This problem is independent of the large search space problem and may occur even when no search is involved. Viewing explanation generation

as search, it is characterized by large values of D.

This section describes three approaches to distinct types of intractability. Lebowitz addresses the large search space problem. Ellman additionally considers the problem of small links. Utgoff and Saxena focus on the efficient use of definitions learned by EBL. This type of intractability will be described in more detail in the discussion of their work.

3.3.1 Lebowitz

In a variation of his system UNIMEM [Lebowitz 86a], Lebowitz uses SBL as a front-end to an explanation-based learning system to address the large search space problem defined above. Rather than generating an explanation for every instance input to the system, UNIMEM first applies a conceptual clustering style of similarity-based learning to input examples, which are represented as sets of features, essentially property/value pairs. The input examples are placed by the SBL algorithm into groups such that the members of each group share a subset of descriptive features. The shared features form a general definition for each group. When SBL has been completed, EBL is applied to the generalizations that have been formed.

One reason for first applying similarity-based learning methods is that by looking for similarities, a system is able to bring some regularity to a domain for which there does not exist a complete theory. More significantly, Lebowitz points out that similarities between examples often occur because of some underlying causal mechanism. To express this, he defines a notion called *predictability*. Features that occur in fewer generalizations are said to be *predictive* because they are most likely to be causes in a causal description, or proof. In [Lebowitz 86b] he explains this by pointing out that a feature present in many different generalizations cannot by itself imply the existence of a feature that occurs in only one of those generalizations, or it would imply its existence in all the other generalizations. He gives the following example from the domain of meteorology. If SBL has created many generalizations that have as a feature DROUGHT, but only one of these has the feature WARM-CURRENTS, then the presence of DROUGHT cannot imply the presence of WARM-CURRENTS or that feature would be present in all the generalizations containing the DROUGHT feature. It is possible, though not guaranteed, that the presence of WARM-CURRENTS implies the presence of DROUGHT. The presence of WARM-CURRENTS in only the single generalization makes it a predictive feature. In traditional EBL, a proof is derived to show that features of the input example imply membership in a concept to be learned. In UNIMEM, a proof is derived to show that predictive features imply the existence of non-predictive features. From there EBL proceeds normally.

By using an SBL component that first classifies and generalizes input examples, UNIMEM attacks the problem of intractable domains in two ways:

1. Rather than explaining all input instances, only generalizations are explained, resulting in fewer explanations to be derived.
2. Predictability can be used to choose relevant features to appear in an

explanation. A large part of the rule space used in deriving an explanation can be pruned in the search for the causal description.

Lebowitz's approach to the problem of intractability constrasts in interesting ways with Ellman's, described next. Further discussion of UNIMEM is therefore deferred until Ellman's work has been discussed.

3.3.2 Ellman

Ellman, in his program POLLYANA [Ellman 87; Ellman 88a; Ellman 88b], approaches the problem of intractability not by pruning the search space for explanations, as Lebowitz does, but by replacing the intractable theory by an approximate theory requiring fewer computational resources. The intractable theory is approximated by adding simplifying assumptions that, although not true in all situations, will be correct in most cases.

The domain to which POLLYANA has been applied is the card game "hearts". An explanation is the computation of an evaluation function yielding the expected final game score for a player who plays a certain card choice. That is, a proof is generated that the particular card choice would lead to the computed final game score. While this could be accomplished by performing a complete minimax search, the process would be intractable.

POLLYANA begins its search for simplifying assumptions in the hearts domain by generating a set of candidate assumptions. It does so by instantiating schemata from a set of predefined generic simplifying assumptions. The system then selects sets of candidate assumptions and integrates them into the initial intractable theory. This produces a collection of approximate theories, organized in a lattice-structured search space. In POLLYANA, functions used to evaluate the expected score are defined in multiple ways, each version implementing a different simplifying assumption. Representing assumptions in this way allows the system to easily determine when a set of assumptions is complete, that is, when there is a definition for each function. Finally, the system conducts an empirical search through the approximate theory space in order to select one of the approximate theories.

Input to POLLYANA is a set of examples where each input instance is a card choice for a given player in a certain trick. For each example, POLLYANA evaluates the expected final game score with an approximate theory, beginning with the simplest of the approximate theories. As discussed above, evaluation is analogous to the proof derivation process. If the scores evaluated meet a pre-specified error threshold, then POLLYANA adopts the approximate theory. If not, the procedure is repeated with the most simple of the remaining approximate theories. Given one set of training examples POLLYANA was led to a theory that ignored the probability of winning the current trick while concentrating on the trick's expected point value. By introducing simplified versions of evaluation functions in POLLYANA, it becomes unnecessary to fully expand minimax game trees, or even subtrees, in order to do evaluation. Thus the method applies to both the large search space and small links problems.

Both Ellman and Lebowitz integrate explanation-based and empirical learning methods in an effort to overcome the intractable theory problem in explanation-based learning. A major advantage of Ellman's approach is that it considers both the large search space and small links problems, whereas Lebowitz looks only at the large search space problem.

Ellman's method is not without its shortcomings, however. It requires a great deal of knowledge beyond that needed for EBL alone. For example, it requires a typology of generic simplifying assumptions. Ellman lists a number of these that are intended to be generally applicable to situations in which the explanation or proof is an evaluation of a function. For instance, in order to simplify an evaluation function, one can assume independence of random variables. That is, Prob(x given y) = Prob(x). It is not clear, however, that such a typology can be specified for cases where the explanation is a more general proof that an example is an instance of a particular concept. Even if this were possible and one could create abstract rules corresponding to approximate functions, the system would still have to be provided with a relation partially ordering the approximations. Another potential problem of Ellman's method is that it introduces incorrectness into a correct domain theory. If the extent of intractability of the domain theory were such that it was not usable for any input example, it could reasonably be argued that the ability to process examples outweighed any inaccuracy introduced. However, this is clearly an extreme case. At the other extreme one could imagine a theory that was tractable for all but a few input examples. It might not be worthwhile to gain explanatory power for a few examples at the expense of sacrificing correctness for the majority of cases.

In contrast to Ellman's approach, Lebowitz's does not require that the system have any knowledge beyond that necessary for EBL in general. Unfortunately, there are problems with the notion of predictability [Lebowitz 86b]. In the meteorology example presented above, UNIMEM would attempt to prove that the presence of the predictive feature, WARM-CURRENTS, implied the presence of DROUGHT. It would be possible, however, that in fact the presence of DROUGHT in conjunction with some other predictive feature (or even a non-predictive one) implied the presence of WARM-CURRENTS. This may be called a problem of *conjunctive predictability*. Considering what appear to be non-predictive features in conjunction with each predictive feature may cause new intractability problems. Not doing so may result in missing explanations.

Finally, Lebowitz and Ellman's approaches can be compared on the basis of their independence from a teacher. In POLLYANA, Ellman's system, teacher-prepared examples are input to the system. UNIMEM can operate without a teacher.

Lebowitz and Ellman approach the intractable domain problem from two very different perspectives. Lebowitz uses generalizations to restrict the search for explanations. Ellman introduces new knowledge that allows a system to find an approximate but tractable theory.

3.3.3 Utgoff and Saxena

Utgoff and Saxena [Utgoff and Saxena 88] consider the problem of intractability in a slightly different sense from Lebowitz and Ellman. They assume that the actual derivation of an explanation is not expensive but that evaluating whether a new instance is covered by an explanation is. Although intended to derive general definitions that are efficiently matched, EBL sometimes produces definitions that are, in fact, inefficient to use. Consider an EBL system that is given as input a graph and a path that is the shortest between two particular nodes of the graph. Finding other graphs that would match the general class for which the path was shortest would involve a solution of the (sub)-graph isomorphism problem, which is NP-complete. (A variant of this is discussed in [Tambe and Newell 88]. Further discussion on the problem of efficiency in using definitions derived by EBL may be found in [Minton 88]).

Utgoff and Saxena essentially apply EBL and SBL in parallel until it is determined that sufficient confidence can be placed in a given definition inferred by SBL. Utgoff and Saxena's learning mechanism is embedded in a larger problem solving system that uses a domain theory and a successfully solved problem to infer a set of prioritized subgoals. A subgoal is defined by a predicate that describes the set of states that satisfy it. Within the predicate is a decision procedure that manages two definitions of the concept: a, presumably inefficient to match, definition learned by EBL, and a, presumably efficient, definition learned by SBL. For a given subgoal predicate, an argument is to be classified as a positive instance if it satisfies the subgoal, and a negative instance if it does not. The problem state is first classified according to the fast definition. If the fast definition determines the answer with high probability, then it is returned. Otherwise, the argument is classified according to the slow definition. Classification according to the slow definition tells the system whether it is a positive or negative instance of the concept for incrementally updating the fast definition using SBL. As the fast definition becomes more reliable, the slow definition is used less often.

Utgoff and Saxena's approach is simple and apparently applicable across domains. It uses EBL to classify inputs as positive or negative rather than relying on an outside source for that information. However, is seems to carry with it some of the weaknesses of both explanation-based and similarity-based learning. The use of definitions produced by EBL requires a complete and correct domain theory; it is assumed that the theory is not intractable in either of the senses described at the beginning of this section. Furthermore, Utgoff and Saxena's system performs pure SBL, causing it to suffer from the shortcomings discussed in Section 4. Finally, it is not clear how their method determines that the concept definition learned by SBL is "good enough" to allow the slow but correct definition to be abandoned.

3.3.4 Summary

This section discussed approaches to three types of intractability. Lebowitz uses predictability information extracted from SBL to prune the space of rules that need to be considered in deriving an explanation. Ellman addresses the small links problem by using empirical methods to select assumptions that simplify a theory by removing detail. Utgoff and Saxena use EBL only until a fast definition learned by SBL is considered accurate.

4 Using Explanation-Based Learning to Adjust the Biases of Similarity-Based Learning

4.1 The Problem of Representational Inadequacy

The entities that can be described by the input and output for a learning system are determined by their knowledge representation. Thus the choice of a particular representation language is said to bias SBL in that it constrains the concepts output from - and therefore learned by - the method.

Preferably the creator of a learning system should not be required to anticipate all input that the system will encounter in a given domain. If that were necessary, learning systems would be limited to functioning in small, non-complex domains. Furthermore, it is desirable that a given learning system be applicable across a wide range of domains. But often a representation suited to one domain does not adequately capture the objects in another. If the burden of anticipating all future processing of a learning system is to be removed from the initial design, a learning system must be able to modify, or at least add to, its body of representable objects.

This section describes three approaches to modifying knowledge representations for SBL. The first two address the issue of modifying a representation in order to use the best for a particular context. The third addresses the problem of learning new vocabularies.

4.1.1 Flann and Dietterich

Flann and Dietterich's SBL system Wyl [Flann and Dietterich 86] pays special attention to the issue of the appropriateness of a knowledge representation. Illustrating their points with examples from the game of checkers, Flann and Dietterich point out that entities similar in one way are often very different in another. For example, many different board configurations might correspond to the concept *trap*. Difficulties can arise when the representation of a concept required for the performance task of a system is not a natural representation for learning that concept. In the checkers domain, an appropriate representation for the task of playing the game is a board configuration. However, providing an SBL system only with examples of board configurations in order to have it learn the concept *trap* would not be satisfactory as static board configurations do not capture the essence of a trap situation.

In order to handle this problem, Flann and Dietterich designed Wyl to accept training instances in a *performance representation* that is explicitly converted into a *learning representation* for SBL. The learned definition is then mapped back into the performance representation. In the game of checkers, for example, input to Wyl is in the form of board configurations, which are translated into functional descriptions for SBL. Conversion from one representation to the other is a proof process in the sense of the explanation construction of EBL. The generalization step of EBL is not performed. The translation of one representation into another more useful to a system is termed **operationalization** [Mostow 81].

Wyl is similar to Buchanan and Mitchell's earlier Meta-DENDRAL [Buchanan and Mitchell 78]. In Meta-DENDRAL training instances were presented in a structural representation consisting of molecular structures and associated mass spectra. These were translated into a learning representation of behavioral descriptions called cleavage processes, sequences of cleavage steps. The cleavage steps were then inductively generalized to obtain general cleavage rules.[7].

Flann and Dietterich's work is an important first step in addressing the problem of the inadequacy of fixed representations. They point out that the choice of a best representation might differ, not only across domains, but within a single domain in different contexts. The ability of a learning system to translate the representation of the performance system in which the learner is embedded to improve the learning process is desirable. The method of translating between representations is generally applicable, but it is not clear that it can be realistically implemented short of devising a separate translation procedure for every different pair of representations. The method does not remove the burden of choosing an appropriate representation from the programmer. The creator of the system is required to choose the representation into which all input is to be translated.

4.1.2 Stepp and Michalski

Stepp and Michalski [Stepp and Michalski 86] address an issue similar to that looked at by Flann and Dietterich. They consider the use of different representations of identical objects for different contexts. More specifically, Stepp and Michalski point out that a set of objects might be classified in a number of ways depending upon the performance task for which the classifications are being used. Thus they consider different contexts to be defined by different performance tasks, while Flann and Dietterich focus on a difference in context between performance and learning.

In Stepp and Michalski's work a single feature set might be adequate to describe all objects in a domain, but different subsets of those features could be more or less important for different tasks. Consider the domain of classifying objects on a restaurant table. Features for

---

[7]This is taken from the clear description of Meta-DENDRAL given in [Flann and Dietterich 86]

this domain might include the material out of which an object is made - such as ceramic for dishes and crystal for glasses. If an object were a container, another feature might be its contents. Say one wanted a system to classify a crystal salt shaker. It would be reasonable to classify it together with the crystal glasses as they share the feature describing the material out of which they are made. It would also be reasonable to classify it with food as the salt shaker contains a seasoning. One or the other classification would be better in the context of a particular performance task - say creating attractive table settings as opposed to making seasoned foods.

Features deemed relevant in a particular situation might furthermore be described most informatively by a new feature name that describes the property they define. For example, rather than describing salt using a feature giving its type as a seasoning and describing lettuce as being of type vegetable, one might want to create a new feature that would describe both as being edible. The process of creating new, more appropriately descriptive features is termed **constructive induction** [Michalski 83]. In order to create general definitions of objects using the conceptual clustering variation of SBL, constructive induction is first performed on the objects. The process begins by selecting the set of features important to the context. This requires a theory that encodes relevance information. Then using another theory, the system chains forward from the relevant features to arrive at new, more descriptive predicates. Conceptual clustering is performed on the newly constructed features.

Stepp and Michalski's method of constructive induction provides for the use of the most descriptive features in a given context. Aside from making the learned concepts more understandable to a human observer, their method avoids problems that might arise if a system were to place significance on coincidental similarities. The method does not allow similarities among irrelevant features to be discovered because it possesses a theory that defines the set of relevant features for any learning context. Features not included in this set need not be considered.

Michalski [Michalski 87], has recently extended his earlier work with Stepp. In this work he notes that learned representations of concepts are inflexible in the boundaries they define for entities that are contained within them, while most concepts in the real world have imprecise meanings. His approach to learning such imprecise concept definitions is based upon the idea of a two-tiered concept representation. The meaning of a concept is defined by two components: the base concept representation (BCR) and the inferential concept interpretation (ICI). The BCR is learned by SBL and may include representative examples and counter-examples of the concept in addition to the general description. The ICI applies a deductive, i.e., explanatory, process using domain knowledge that represents such information as the importance of concept attributes and the context of discourse. Thus an EBL-like component is appended to what is otherwise an SBL system. Preliminary work on this learning mechanism has been applied by Michalski to the domain of lymphography.

Watanabe and Elio [Watanabe and Elio 87] consider constructive induction in the context of incremental inductive learning, i.e., in a situation in which not all examples are presented to the learning mechanism at once. Their system, LAIR, has been applied to the problem of learning the definition of a cup.

4.1.3 Utgoff

Utgoff, with STABB [Utgoff 86], addresses a more fundamental issue in representational inadequacy than those discussed above. In many SBL systems, general names are defined to describe subsets of the values that a particular feature might take. However, these names might not fully describe every learning situation that a system might encounter. For example, in the LEX system that solves problems in the domain of integral calculus [Mitchell 83], there is encoded in the system that sin, cos, tan, sec, csc, and cot are trigonometric functions. In creating a general concept description for which examples had specific trigonometric functions as values of a particular feature, LEX might generalize the value of the feature to be "any trigonometric function". A problem would arise if, say, the values appearing for a single feature in all examples were either sin or cos, and the system had no way to describe this subset of the trigonometric functions. The system would have to choose between making the generalization to "any trigonometric function" and representing the disjunction of these values explicitly. Making the generalization results in a more succinct representation of the feature. On the other hand, it might be incorrect. More formally, a problem exists because the hypothesis space of concept descriptions does not include a succinct definition that is consistent with the training instances.

STABB is an extension to LEX; thus its domain of application is integration. The process of integration may be viewed as the transformation of an equation containing an integral sign into one that does not include the integral sign. Each legal rule of transformation is an operator. The learning goal is to create a general description for a class of equations to which the application of a particular operator leads to a solved integral. Input to STABB includes an operator and an example of a full solution trace in which the application of the particular operator was useful. STABB uses constraint propagation to identify new concepts that should be represented in the description language. This is a mechanism that can be used to generalize a proof in EBL. If constraint propagation fails, STABB falls back on using the least disjunction of the values observed in multiple training instances. Thus, in order to identify new general values to be described in the representation language, STABB uses a solution trace, or explanation, and constraint propagation, which together are the mechanisms of EBL.

Utgoff considers a basic issue in representational inadequacy, that of constructing new terms that do not yet exist in the concept language. Although such a mechanism is necessary for autonomous learning, it is not clear that his technique is the appropriate method to use. It is based almost exactly upon an explanation-based learning algorithm and thus will have associated with it the problems caused by unrealistic assumptions about the domain theory.

The one example of a learned term shown by Utgoff is that describing odd numbers. It is not clear how extensible the method is.

4.1.4 Summary

This section described three approaches to the problem of representational inadequacy. Flann and Dietterich use deduction to transform one representation into another that is better suited for a particular context. Stepp and Michalski use a similar approach. Utgoff applies EBL for learning new terms that better describe generalizations of feature values.

4.2 The Problem of Searching a Large Hypothesis Space

Similarity-based learning may be viewed as search through the hypothesis space of representable concepts for one that is consistent with all positive input examples and inconsistent with negative input examples. The hypothesis space is potentially huge. A problem to be addressed is how search may be directed to prune large parts of the hypothesis space.

[Mitchell 84] was among the first to propose that explanation-based learning be used to focus the attention of SBL, enabling it to make larger leaps through its hypothesis space of concept definitions. Mitchell's proposed solution is an extension of the version space algorithm [Mitchell 78] for SBL. A version space is a lattice of representable concepts. Representations of specific instances are at one end of the lattice, while the most general concept definition, of which all entities are members, is at the other end. The version space algorithm maintains two sets of concepts. One set, S, contains the most specific concept definitions consistent with all the positive examples seen by the system. The other, G, contains the most general definitions not inconsistent with any of the negative examples seen. The algorithm learns incrementally, i.e., concept definitions are modified after seeing each input. Positive input examples may cause the S set to be modified by making its members more general. Negative input examples cause the G set to be modified, making it more specific so as to exclude the negative examples. S and G sets are pictured in Figure 15. After seeing enough input, the S and G sets should grow toward each other and become equivalent. This is the learned concept definition.

Mitchell suggested that performing EBL on each positive input example provides a set of sufficient conditions for being an instance of the concept to be learned. This set of sufficient conditions provides a basis for refining the S set of the version space. Negative instances are still used to refine the G set. This has recently been implemented by Hirsh [Hirsh 89].

A number of researchers have more recently investigated the idea of using EBL to provide extra information to SBL in order that it might intelligently prune its search space. The following sections present the approaches of Salzberg, Swaminathan, Danyluk, and Van Lehn.

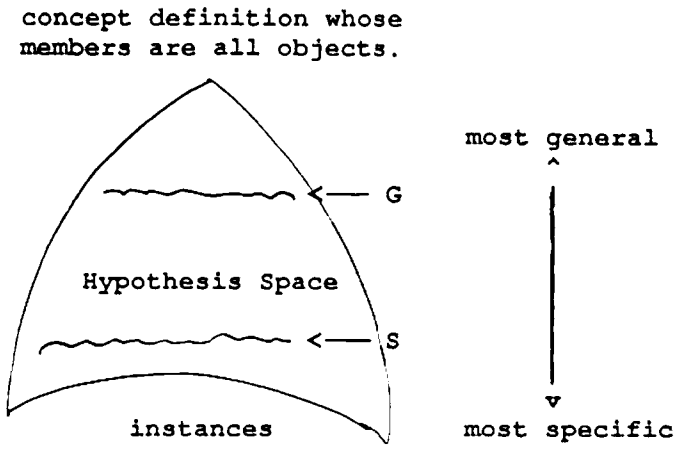

Figure 15: Boundary Sets of the Version Space

4.2.1 Salzberg

Salzberg, with HANDICAPPER [Salzberg 83; Salzberg 85], derives explanations to identify potentially important features for revising the definition of a concept. HANDICAPPER's domain of application is horse racing, where its performance task is to predict the horse that will win a race. The learning task is to identify sets of features that define the concepts "winning horse" and "losing horse". Input to HANDICAPPER are feature sets describing the horses in a race. HANDICAPPER chooses a winner by selecting the horse with features most similar to those describing a winner in general. If the prediction is incorrect, then the concept definitions for winning and losing horses are modified.

To modify the feature sets defining the two concepts, HANDICAPPER performs SBL, looking for *differences* between the horse predicted to win and the horse that actually won. This provides the system with a choice of features to be modified in the concept definitions for winning and losing horses respectively. Next explanations are derived that enable the system to identify those features most likely to require modification. In HANDICAPPER explanations relate features of a single horse to each other. The domain knowledge contains rules linking the presence of certain features to others. Concentrating on those features found to be different in the SBL phase, the EBL phase looks for those that contradict rules in the domain knowledge. No generalization of the explanation is done. The features identified by the EBL phase are modified in the concept description.

Salzberg's approach is similar to those taken by Swaminathan and Danyluk. Further discussion of his method is therefore deferred until theirs have been introduced.

4.2.2 Swaminathan

Swaminathan, whose system has been applied to the domain of epidemiology [Swaminathan 88], uses similarity-based learning to identify a set of features common to the victims of an illness in a manner similar to Salzberg's for horse racing. Because the feature set describing victims of an illness can potentially be huge, Swaminathan uses EBL to identify a set of features on which to focus. The domain theory used by the explanation-based component of Swaminathan's system is a partial causal theory of disease. The parts of the domain theory that are missing are those that refer to specific values of features describing input examples. For instance, given a set of hepatitis victims for which the system needs to find the cause of the illness, the domain theory would contain general information about contaminated water sources possibly causing hepatitis. It would not contain more specific information stating what water sources were contaminated. Explanations are generated using the partial theory linking general features of the input examples to the illness. As an explanation does not refer to specific feature values, it is not a proof that those features referred to are *actual* causes. It is a proof that those features are *possible* causes. SBL concentrates on these, presumably relevant, attributes of the inputs.

4.2.3 Danyluk

Danyluk, in Gemini, uses explanations in three distinct ways to aid in conceptual clustering [Danyluk 87; Danyluk 88]. All three follow from deriving an explanation of each input example before performing similarity-based learning. Gemini has been applied to the domains of network fault diagnosis, radio fault diagnosis, and terrorist event news stories.

The first way that explanations are used in Gemini to focus SBL is similar to the approaches of Salzberg and Swaminathan described above. Features used in the explanation of an input example are deemed to be most relevant. Features considered to be less relevant are not discarded, but are not considered to be important in the definition of a concept and thus are not generalized by SBL. Second, explanations provide a context for generalization of feature values. Unlike earlier SBL systems, Gemini is not biased to choose a single generalization for each subset of possible feature values. Multiple choices exist, one of which will be more or less appropriate in different contexts. Context is encoded in the domain knowledge used by EBL. Finally, explanations are used as a filter to safeguard against classifying an entity as a member of a particular concept if it agrees with that concept definition only in the syntactic similarity of its feature values. Explanation structures of input objects must at least partially match if they are to be included as members of the same concept.

Danyluk, Salzberg, and Swaminathan all use explanations to focus on a subset of features for matching and generalization in SBL. Also falling into the class of methods that use deduction to focus on feature subsets is constructive induction, discussed earlier in Section 4.1.2. None of these explicitly requires that the domain theory with which it works be

complete, yet each essentially treats it as if it were. This clearly can be problematic. In the cases of Salzberg and Swaminathan, as well as in constructive induction, features not identified by an explanation as important are not even considered by SBL. Danyluk looks at features not appearing in an explanation, but places stringent requirements on the ways in which they can be generalized. Danyluk further requires that auxiliary context knowledge be explicitly provided to SBL for better generalizations, an extra burden for the system designer.

4.2.4 Van Lehn

In Van Lehn's SIERRA system [Van Lehn 87], guidance as to the relative importance of features and the way in which to generalize them using SBL is provided by a teacher. Certain conventions, or **felicity conditions**, are understood by both the similarity-based learning system and the teacher. For example, in SIERRA's performance domain, arithmetic procedures, it is understood that the teacher will introduce at most one new disjunct to a concept per lesson. This drastically cuts the hypothesis space to be searched by SIERRA in performing inductive inference. Because of its dependence upon a teacher and felicity conditions between the teacher and learning mechanism, this work is of a somewhat different nature than the systems above. However, it is included here because of an explanatory step that SIERRA performs before doing induction. Upon receiving an input example in the form of a sequence of actions that solve an arithmetic problem, SIERRA attempts to parse the action sequence. The parse tree constitutes an explanation for the action sequence. The inability to complete a parse indicates the new disjunct, in SIERRA's case a subprocedure, to be learned. Due to the nature of the learned disjunct - i.e., that it is a new arithmetic rule - this may also be viewed as a use of induction for the incomplete theory problem of EBL.

4.2.5 Summary

This section discussed three basic approaches to cutting the search of SBL. Mitchell proposed that a positive example generalized by EBL be input to the version space algorithm. The single generalized example is essentially equivalent to a set of ungeneralized positive examples. Salzberg, Swaminathan, and Danyluk use explanations to focus the attention of SBL on features that are presumably most relevant. Van Lehn relies on felicity conditions between a teacher and learning system in order to focus incremental inductive learning. Although the above methods guide the search, an extra cost is incurred by generating the explanations that provide the focus for SBL.

5 Related Work in the Integration of Deduction and Induction

Other work in machine learning might be viewed as integrating inductive, or similarity-based, and deductive, or explanation-based, mechanisms. A number of researchers have investigated the use of **analogy** as a learning mechanism. Analogy, the inference that if two or more things agree in some respects they will probably agree in others, involves the matching of one entity with another. Thus the matching techniques of SBL are often applied here.

Furthermore, since similarities might be found at a different level from the purely superficial one of entities' observable characteristics, the entities are often analyzed using deductive, or explanation-based, mechanisms as well. Examples of work in analogy include [Burstein 86; Carbonell 83a; Carbonell 83b; Kedar-Cabelli 85; Russell 86].

Exemplar-based learning is another mechanism that combines inductive and deductive techniques. In exemplar-based learning a concept is represented by a stereotypical example. Deduction, or explanation derivation, is used to determine whether an input might be an instance of the concept even if it does not exactly match the definition of the exemplar for the class. An example of work in this area is [Bareiss and Porter 87]. More generally, work has been done in **Case-Based Reasoning (CBR)**. The philosophy of CBR is that solutions to problems need not always be derived from scratch. A new problem solving situation might be similar to one encountered earlier. It might be useful to modify the solution to the old problem. In CBR cases of previously seen problems and their solutions are stored in memory. When a new problem arises an old case is retrieved. Mechanisms similar to those described above for analogy may be used to retrieve appropriate cases for modification. Work in CBR includes [Hammond 88; Kolodner 87].

Finally, work in expert systems knowledge acquisition is not discussed here due to the fact that, although the lack of necessary knowledge is often detected by expert systems, it is generally then provided directly by an outside user. Although deductive mechanisms are often used by expert systems, they are most often used as mechanisms for reasoning and problem solving, and not directly for knowledge acquisition purposes.

6 Conclusion

This paper has surveyed machine learning systems that integrate explanation-based and similarity-based learning. EBL is an analytical approach that has as a major goal the re-structuring of an explicit domain theory to improve performance efficiency. SBL has the goal of acquiring concept definitions not yet known to a system. It does not require the use of an explicit domain theory as does EBL. Both methods have been applied with some success to a variety of domains. However, the two techniques make assumptions that are not guaranteed to hold. In EBL it is assumed that the explicit domain theory is complete, correct, and tractable; in SBL it is assumed that the knowledge representation is adequate to describe the concepts to be learned and that the process of searching for the appropriate concept definition is tractable. Since the two methods are complementary in nature, researchers have found it natural to use one as a solution to a problem with the other. The systems discussed above integrate EBL and SBL in this way.

A discussion and critique of each system has been presented. A major criticism of all the systems, not yet addressed, is that in augmenting one system with another, entirely new problems are necessarily introduced. Specifically, when SBL is introduced into an EBL system,

it cannot be assumed that SBL will work perfectly. The same is true when EBL is incorporated to solve a problem of SBL. The problems of potential domain theory incompleteness, incorrectness, and intractability are introduced into the composite system. Although the methods are complementary, one might conjecture that the pieces alone are not enough. Rather, they need to be expanded into a mechanism of mutual interaction between the parts.

I. A Summary of Selected Systems Addressing Problems of Explanation-Based Learning

PROBLEM	SYSTEM	APPROACH	SAMPLE DOMAIN	NEW THEORY FOR EBL
INCOMPLETENESS				
	Rajamoney's ADEPT	Auxiliary theory for experimentation	Physical processes	Yes
	Hall	User-provided analogue for system's analysis	Logic circuits	Yes
	Pazzani's OCCAM	Auxiliary theory of causality	Economic sanctions	Yes
	Kodratoff & Tecuci's DISCIPLE	User proposes and verifies rules	Equipment manufacture	Yes
INCORRECTNESS				
Wrong granularity	Araya	Auxiliary theory for generation of better explanations	Physics problem solving	Yes
Wrong granularity	Silver's LP	Vote on use of learned knowledge	Mathematics problem solving	No
Wrong information	Dietterich & Flann's IOE	Explanations as input to induction	Disease causes	Yes
INTRACTABILITY				
Large search space	Lebowitz's UNIMEM	SBL generalizations focus explanation construction	Meteorology	No
Large search space and Small links	Ellman's POLLYANA	Empirical selection of assumptions to simplify theory	Hearts	Yes
Efficient definition match	Utgoff & Saxena	EBL and SBL in parallel until SBL good enough	State descriptions for subgoals	Yes

II. A Summary of Selected Systems Addressing Problems of Similarity-Based Learning

PROBLEM	SYSTEM	APPROACH	SAMPLE DOMAIN	NEW THEORY FOR EBL
REPRESENTATION INADEQUACY				
Context-sensitive representation	Flann & Dietterich's Wyl	EBL translates to operational representation	Checkers	No
Context-sensitive representation	Stepp & Michalski	EBL to select and convert features to better terms		No
Creating new terms	Utgoff's STABB	EBL to generate new terms	Symbolic integration	Yes
CUTTING SEARCH				
Feature focus	Salzberg's HANDICAPPER	Find features that contradict theory	Horse racing	No
Feature focus	Swaminathan	Theory gives set of features for consideration	Epidemiology	No
Feature focus	Danyluk's Gemini	Relevance, context, and matching of explanations as input to SBL	Network fault diagnosis	No
Feature focus	Van Lehn's SIERRA	Explanation, or parse, indicates new disjunct	Arithmetic procedures	Yes

References

[Araya 84] Araya, A. A.
Learning Problem Classes by Means of Experimentation and Generalization.
In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 11 - 15. Austin, Texas, 1984.

[Bareiss and Porter 87]
Bareiss, E. R. and Porter, B. W.
Protos: An Exemplar-Based Learning Apprentice.
In *Proceedings of the Fourth International Machine Learning Workshop*, pages 12 - 23. Irvine, California, 1987.

[Buchanan and Mitchell 78]
Buchanan, B. G. and Mitchell, T. M.
Model-Directed Learning of Production Rules.
Pattern-Directed Inference Systems.
Academic Press, New York, 1978.

[Burstein 86] Burstein, M. H.
Concept Formation by Incremental Analogical Reasoning and Debugging.
Machine Learning: An Artificial Intelligence Approach, Volume II.
Morgan Kaufmann, Los Altos, California, 1986.

[Carbonell 83a] Carbonell, J. G.
Derivational Analogy in Problem Solving and Knowledge Acquisition.
In *Proceedings of the Second International Machine Learning Workshop*, pages 12 - 18. Champaign-Urbana, Illinois, 1983.

[Carbonell 83b] Carbonell, J. C.
Learning by Analogy: Formulating and Generalizing Plans from Past Experience.
Machine Learning: An Artificial Intelligence Approach.
Morgan Kaufmann, Los Altos, California, 1983.

[Danyluk 87] Danyluk, A. P.
The Use of Explanations for Similarity-Based Learning.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 274 - 276. Milan, Italy, 1987.

[Danyluk 88] Danyluk, A. P.
An Integrated Learning Method for Purpose-Guided Clustering.
Technical Report, Columbia University Department of Computer Science, 1988.

[DeJong 81] DeJong, G. F.
Generalizations Based on Explanations.
In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 67 - 69. Vancouver, B. C., Canada, 1981.

[DeJong 83] DeJong, G. F.
Acquiring Schemata through Understanding and Generalizing Plans.
In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 462 - 464. Karlsruhe, West Germany, 1983.

[DeJong and Mooney 86]
DeJong, G. and Mooney, R.
Explanation-Based Learning: An Alternative View.
Machine Learning 1(2):145 - 176, 1986.

[Dietterich and Flann 88]
Dietterich, T. G. and Flann, N. S.
An Inductive Approach to Solving the Imperfect Theory Problem.
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 42 - 46. Stanford University, 1988.

[Ellman 87] Ellman, T.
Explanation-Based Methods for Simplifying Intractable Theories.
Technical Report CUCS-265-87, Columbia University Department of Computer Science, 1987.

[Ellman 88a] Ellman, T.
Explanation-Directed Search for Simplifying Assumptions.
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 95 - 99. Stanford University, 1988.

[Ellman 88b] Ellman, T.
Approximate Theory Formation: An Explanation-Based Approach.
In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 570 - 574. St. Paul, Minnesota, 1988.

[Ellman 89] Ellman, T.
Explanation-Based Learning: A Survey of Programs and Perspectives.
Computing Surveys 21(2):163 = 221, 1989.

[Fisher 87] Fisher, D. H.
Conceptual Clustering, Learning From Examples, and Inference.
In *Proceedings of the Fourth International Machine Learning Workshop*, pages 38 - 49. Irvine, California, 1987.

[Flann and Dietterich 86]
Flann, N. S. and Dietterich, T. G.
Selecting Appropriate Representations for Learning from Examples.
In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 460 - 466. Philadelphia, Pennsylvania, 1986.

[Forbus 84] Forbus, K.
Qualitative Process Theory.
Artificial Intelligence 24:85 - 168, 1984.

[Hall 86] Hall, R. J.
Learning by Failing to Explain.
In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 568 - 572. Philadelphia, Pennsylvania, 1986.

[Hammond 88] Hammond, K.
Case-Based Planning: Viewing Planning as a Memory Task.
In *Proceedings of the Case-Based Reasoning Workshop*, pages 17 - 20. Clearwater Beach, Florida, 1988.

[Hirsh 89] Hirsh, H.
Incremental Version-Space Merging: A General Framework for Concept Learning.
PhD thesis, Stanford University Department of Computer Science, 1989.

[Kedar-Cabelli 85] Kedar-Cabelli, S.
Purpose-Directed Analogy.
In *Proceedings of the Cognitive Science Society Conference.* Irvine, California, 1985.

[Kodratoff 87] Kodratoff, Y. and Tecuci, G.
DISCIPLE-1: Interactive Apprentice System in Weak Theory Fields.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 271 - 273. Milan, Italy, 1987.

[Kodratoff and Tecuci 87] Kodratoff, Y. and Tecuci, G.
What is an Explanation in DISCIPLE?
In *Proceedings of the Fourth International Machine Learning Workshop*, pages 160 - 166. Irvine, California, 1987.

[Kolodner 87] Kolodner, J. L.
Extending Problem Solver Capabilities Through Case-Based Inference.
In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 167 - 178. Irvine, California, 1987.

[Lebowitz 86a] Lebowitz, M.
Integrated Learning: Controlling Explanation.
Cognitive Science 10:219 - 240, 1986.

[Lebowitz 86b] Lebowitz, M.
Not the Path to Perdition: The Utility of Similarity-Based Learning.
In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 533 - 537. Philadelphia, Pennsylvania, 1986.

[Lebowitz 87] Lebowitz, M.
Experiments with Incremental Concept Formation: UNIMEM.
Machine Learning 2(2):103 - 138, 1987.

[Machine Learning, Volume I 83] Michalski, R. S., Carbonell, J. G., Mitchell, T. M., eds.
Machine Learning: An Artificial Intelligence Approach, Volume I.
Morgan Kaufmann publishers, Inc., Los Altos, California, 1983.

[Michalski 83] Michalski, R. S.
A Theory and Methodology of Inductive Learning.
Machine Learning: An Artificial Intelligence Approach.
Morgan Kaufmann, Los Altos, California, 1983.

[Michalski 87] Michalski, R. S.
How to Learn Imprecise Concepts: A Method for Employing a Two-Tiered Representation in Learning.
In *Proceedings of the Fourth International Machine Learning Workshop*, pages 50 - 58. Irvine, California, 1987.

[Michalski and Ko 88] Michalski, R. S. and Ko, H.
On the Nature of Explanation or Why Did the Bottle Shatter?
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 12 - 16. Stanford University, 1988.

[Michalski and Stepp 83] Michalski, R. S. and Stepp, R. E.
Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy.
IEEE Transactions on Pattern Analysis and Machine Intelligence 5(4):396 - 409, 1983.

[Minton 88] Minton, S.
Learning Effective Search Control Knowledge: An Explanation-Based Approach.
PhD thesis, Carnegie-Mellon University Department of Computer Science, 1988.

[Mitchell 78] Mitchell, T. M.
Version Spaces: An Approach to Concept Learning.
PhD thesis, Stanford University Department of Computer Science, 1978.

[Mitchell 80] Mitchell, T. M.
The Need for Biases in Learning Generalizations.
Technical Report CBM-TR-117, Rutgers University Department of Computer Science, 1980.

[Mitchell 83] Mitchell, T.
Learning and Problem Solving.
In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 1139 - 1151. Karlsruhe, West Germany, 1983.

[Mitchell 84] Mitchell, T. M.
Toward Combining Empirical and Analytical Methods for Inferring Heuristics.
Human and Artificial Intelligence.
North Holland Publishing Company, Amsterdam, 1984.

[Mitchell et al. 86] Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S.
Explanation-Based Generalization: A Unifying View.
Machine Learning 1(1):47 - 80, 1986.

[Mostow 81] Mostow, D. J.
Mechanical Transformation of Task Heuristics into Operational Procedures.
PhD thesis, Carnegie-Mellon University Department of Computer Science, 1981.

[Pazzani 87] Pazzani, M. J.
Inducing Causal and Social Theories: A Prerequisite for Explanation-Based Learning.
In *Proceedings of the Fourth International Machine Learning Workshop*, pages 230 - 241. Irvine, California, 1987.

[Pazzani 88] Pazzani, M. J.
Learning Causal Relationships: An Integration of Empirical and Explanation-Based Learning Methods.
PhD thesis, UCLA Department of Computer Science, 1988.

[Pazzani et al. 86] Pazzani, M., Dyer, M., and Flowers, M.
The Role of Prior Causal Theories in Generalization.
In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 545 - 550. Philadelphia, Pennsylvania, 1986.

[Pazzani et al. 87] Pazzani, M., Dyer, M., and Flowers, M.
Using Prior Learning to Facilitate the Learning of New Causal Theories.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 277 - 279. Milan, Italy, 1987.

[Quinlan 86] Quinlan, J. R.
Induction of Decision Trees.
Machine Learning 1(1):81 - 106, 1986.

[Rajamoney 88] Rajamoney, S. A.
Experimentation-Based Theory Revision.
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 7 - 11. Stanford University, 1988.

[Rajamoney 89] Rajamoney, S. A.
Explanation-based theory revision: An approach to the problems of incomplete and incorrect theories.
PhD thesis, University of Illinois Department of Computer Science, 1989.

[Rajamoney and DeJong 87]
Rajamoney, S. and DeJong, G.
The Classification, Detection and Handling of Imperfect Theory Problems.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 205 - 207. Milan, Italy, 1987.

[Rajamoney et al. 85]
Rajamoney, S., DeJong, G., and Faltings, B.
Towards a Model of Conceptual Knowledge Acquisition Through Directed Experimentation.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 688 - 690. Los Angeles, California, 1985.

[Russell 86] Russell, S. J.
Analogical and Inductive Reasoning.
PhD thesis, Stanford University Department of Computer Science, 1986.

[Salzberg 83] Salzberg, S.
Generating Hypotheses to Explain Prediction Failures.
In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 352 - 355. Washington, DC, 1983.

[Salzberg 85] Salzberg, S.
Heuristics for Inductive Learning.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 603 - 609. Los Angeles, California, 1985.

[Silver 86] Silver, B.
Precondition Analysis: Learning Control Information.
Machine Learning: An Artificial Intelligence Approach, Volume II.
Morgan Kaufmann, Los Altos, California, 1986.

[Silver 88] Silver, B.
A Hybrid Approach in an Imperfect Domain.
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 52 - 56. Stanford University, 1988.

[Stepp and Michalski 86]
Stepp, R. E. and Michalski, R. S.
Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects.
Machine Learning: An Artificial Intelligence Approach, Volume II.
Morgan Kaufmann, Los Altos, California, 1986.

[Swaminathan 88] Swaminathan, K.
Integrated Learning with an Incomplete and Intractable Domain Theory: The Problem of Epidemiological Diagnosis.
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 62 - 66. Stanford University, 1988.

[Tambe and Newell 88] Tambe, M. and Newell, A.
Some Chunks Are Expensive.
In *Proceedings of the Fifth International Conference on Machine Learning*, pages 451 - 458. Ann Arbor, Michigan, 1988.

[Utgoff 86] Utgoff, P. E.
Shift of Bias for Inductive Concept Learning.
Machine Learning: An Artificial Intelligence Approach, Volume II.
Morgan Kaufmann, Los Altos, California, 1986.

[Utgoff and Saxena 88] Utgoff, P. E., and Saxena, S.
Obtaining Efficient Classifiers from Explanations.
In *Proceedings of the AAAI Symposium on Explanation-Based Learning*, pages 47 - 51. Stanford University, 1988.

[Van Lehn 87] Van Lehn, K.
Learning One Subprocedure per Lesson.
Artificial Intelligence 31:1 - 40, 1987.

[Watanabe and Elio 87] Watanabe, L. and Elio, R.
Guiding Constructive Induction for Incremental Learning from Examples.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 293 - 296. Milan, Italy, 1987.

[Winston 72] Winston, P. H.
Learning Structural Descriptions from Examples.
The Psychology of Computer Vision.
McGraw-Hill, New York, 1972.

[Winston et al. 83] Winston, P. H., Binford, T. O., Katz, B., and Lowry, M.
Learning Physical Descriptions from Functional Definitions, Examples, and Precedents.
In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 433 - 439. Washington, DC, 1983.