Integrated Learning: Controlling Explanation

Michael Lebowitz
Department of Computer Science
Columbia University
New York, NY 10027

July, 1985

CUCS-201-85

To appear in *Cognitive Science*.

Integrated Learning: Controlling Explanation

Michael Lebowitz[1]
Department of Computer Science
Computer Science Building, Columbia University
New York, NY 10027

Running head: *Integrated Learning*

Abstract

Similarity-based learning, which involves largely structural comparisons of instances, and explanation-based learning, a knowledge-intensive method for analyzing instances to build generalized schemata, are two major inductive learning techniques in use in Artificial Intelligence. In this paper, we propose a combination of the two methods -- applying explanation-based techniques during the course of similarity-based learning. For domains lacking detailed explanatory rules, this combination can achieve the power of explanation-based learning without some of the computational problems that can otherwise arise. We show how the ideas of *predictability* and *interest* can be particularly valuable in this context. We include an example of the computer program UNIMEM applying explanation to a generalization formed using similarity-based methods.

1 Introduction

Current research in inductive machine learning includes two relatively disparate approaches: traditional *similarity-based learning* (SBL) that involves comparisons of instances[2] of a concept, e.g., (Winston, 1972; Winston, 1980; Michalski, 1980; Michalski, 1983; Dietterich and Michalski, 1983; Lebowitz, 1983a; Lebowitz, 1983b), among many others, and a newer line of research that involves intensive application of knowledge to single instances (at a time), including analysis of hypothetical generalizations of the example, which we will refer to as *explanation-based learning*[3] or EBL, e.g., (Carbonell, 1983; DeJong, 1983; Ellman, 1985; Minton, 1984; Mitchell, 1983; Mostow, 1983; Salzberg,

[1]This research was supported in part by the United States Army Research Institute under contract MDA903-85-0103 and in part by the United States Defense Advanced Research Projects Agency under contract N00039-84-C-0165. Comments by Kathy McKeown, David Waltz, and anonymous reviewers of an earlier draft of this paper were quite useful as were several discussions with Jerry DeJong and Tom Mitchell. This paper will appear in *Cognitive Science*, Volume 10, 1986.

[2]An instance is defined here as a single item of input -- event or object -- given to a learning program. We use this term to avoid confusion over the various meanings of the word example.

[3]Term due to Jerry DeJong.

1983; Silver, 1983). Little has been done to relate these two methods (although Michalski (1983) provides a framework for doing so), and yet the combination seems crucial for robust learning. In this paper, we will show how two ideas, *predictability* and *interest*, can help bridge the gap. Application of these ideas allows us to control the explanation process in situations where it might otherwise be computationally explosive.

Considerable research has been done involving *similarity-based learning*. While there are many varieties to such learning, the basic idea is that a program takes a number of instances, compares them in terms of similarities and differences, and creates a generalized description based on this structural analysis. Such learning has been studied for cases where the input is specially prepared by a teacher; for unprepared input; where there are only positive instances; where there are both positive and negative instances; for a few instances; for many instances; for determining only a single concept at a time; and for determining multiple concepts. Cohen and Feigenbaum (1982) and Michalski, Carbonell and Mitchell (1983) provide good summaries of this research. Practically, SBL programs have learned by comparing instances more or less syntactically, using little "high level" knowledge of their domains (other than in deciding how to represent each instance initially).

In the last few years, another approach has become popular in the machine learning field -- *explanation-based learning*. This line of research views learning as a knowledge-intensive activity, much like other tasks in AI. An EBL program takes a single instance, builds up an explanation of how the various components relate to each other using traditional, domain-dependent AI understanding or planning methods, and then generalizes the properties of various components of the instance as long as the explanation remains valid. What is left is then viewed as a generalized description of the instance that can be applied in understanding further instances. This kind of learning is tremendously useful, as it allows generalized concepts to be determined on the basis of a single instance. On the other hand, the building and analysis of explanations does require extremely detailed knowledge of the domain (which may minimize the need to learn). In addition, virtually all current EBL work is in the "perfect learner" paradigm that assumes that all input is noise-free and fits the correct final generalization.

Perhaps the easiest way to see the difference between these methods is to consider one of the

earliest examples of SBL research, Winston's blocks world arch learning program (Winston, 1972). Winston's program learned the concept of a simple blocks world concept such as an arch by comparing instances of arches and non-arches (carefully selected by a teacher) to determine the essential elements of the concept. The positive instances were used to loosen constraints in the concept (e.g., if one arch has a rectangular lintel and another a triangular lintel then "arch" might only require a polyhedral lintel). Conversely, the negative instances made the definition more specific (so, if in the previous example, the structure with the triangular lintel was *not* an arch, then the program would assume that arches *must* have rectangular lintels). A sequence of well-chosen instances led to an accurately defined arch.

EBL analysis in this domain would be very different. It would require detailed knowledge of the blocks world domain, perhaps including information about gravity, support requirements, and so forth. For EBL to make sense, there would have to be some information about the *purpose* of arches (otherwise they would just be arbitrary collections of blocks that could not be further analyzed). Suppose the purpose of an arch required it to support an object in the air and allow movement under it. EBL processing would begin with a single instance of an arch, say two red, rectangular supports and a blue, rectangular lintel. First, an EBL program would analyze the structure in terms of its domain knowledge -- deducing that the uprights support the lintel, perhaps.

After its initial analysis, the EBL program would further analyze its representation to determine which elements were crucial in achieving the desired purposes, and which elements could be generalized or were entirely superfluous and could be eliminated. In our example, it might determine that the existence of two supports, not touching each other, was crucial in allowing traffic underneath, but that the shape of the lintel could be generalized to any shape that the uprights could support, since the purpose would still be achieved. The colors of the bricks would be found to be totally superfluous, and should not be part of a generalized arch description.

We can see from this example the advantage of EBL, at least for cases where we have a substantial amount of domain knowledge, but not necessarily a large set of instances. We were able to determine a reasonable definition of an arch from a single instance. Furthermore, even if we did have a number of instances available, by looking at a detailed representation of how the parts of the arch

interrelate, we are somewhat less likely to generalize the kinds of coincidental information that often arises in SBL. In addition, the kinds of explanations needed for EBL may be useful for other aspects of processing, e.g., (Schank, 1975; Schank, 1984). On the other hand, to successfully carry out EBL processing, we had to have available a rather extensive amount of information. Further, if we had many applicable understanding rules, the explanation process could become very expensive computationally.

As we try to scale up EBL to domains larger than the blocks world, a major apparent problem is that it may be very difficult to develop the initial causal explanation that the process relies upon. This is particularly true for systems that lack detailed domain knowledge and only have available general explanatory rules. Additionally, since the EBL process is computationally complex, we will not want to apply it to all instances or to all elements of instances we do consider. We suggest here a model that combines SBL and EBL methods, one that does learning by noticing similarities when efficient (e.g., specific) understanding rules are not available, or when the payoff from EBL is not likely to be high, and applies EBL-type analysis at carefully selected times -- most likely when we have a number of generalizations based on similarities that we are fairly confident of. The concepts that are generalized in this manner can then be applied to the explanation process for later instances. We feel that this is a promising path to robust learning, that allows us to minimize the necessary initial domain information.

2 An EBL Example

DeJong (1983) used the following story, STORY1, to illustrate explanation-based learning. We will use it to show some of the problems that can arise in doing such processing.

> **STORY1** - Paris police disclosed Tuesday that a man who identified himself as Jean Maraneaux abducted the 12-year-old daughter of wealthy businessman Michel Boullard late last week. Boullard received a letter containing a snapshot of the kidnapped girl. The next day he received a telegram demanding that 1 million francs be left in a lobby waste basket of the crowded Pompidou Center in exchange for the girl. Asking that the police not intervene, Boullard arranged for the delivery of the money. His daughter was found wandering blindfolded with her hands bound near his downtown office on Monday.

DeJong's program first applies standard story understanding techniques to build up a detailed causal representation of the events in STORY1. This representation includes links that show how various aspects of the *deduction* of the causal links depend upon each other. Then, the program repeatedly

substitutes more and more general descriptions of entities in the story, as long as the causal explanation remains valid. So, for instance, it might discover that the 1 million francs could be replaced by any large amount of money and that the place where the money was transferred need not be the Pompidou Center, but could be any public place. The final representation, using the most generalized entities that allow the explanation to remain valid, constitutes a hypothesized "kidnap" schema.

The EBL method works very well for this example, primarily because DeJong's program has available a rich model of the domain, and so can build up a very detailed representation of the story. Further, EBL is applied efficiently because the program appears to have *only* relevant information. If this story was embedded in a system with a wider range of information that operated on a large range of instances, several potentially serious problems would arise including: 1) while looking at instances, deciding when to generalize; 2) forming the initial explanation of each instance in a computationally feasible manner; and 3) from all the possible explanations that could be derived from a story, and all the parts of a complex explanation, deciding which pieces to generalize.

DeJong does address the first question. He presents five heuristics for deciding when to generalize (whether the main goal of a character is achieved, whether the goal is general, whether the resources needed by the character are generally achievable, whether the method is at least as effective as known ones, and whether the method is not already known). These heuristics are certainly related to the interest-based proposal we will make, in some sense defining "interesting" for DeJong's system. However, note that these heuristics are, like the method itself, knowledge intensive in terms of the information needed about the domain. We will consider the case where considerably less information is available for deciding when and what to generalize, and, more specifically, what instances (or generalizations made using SBL), should be subjected to full EBL analysis.

Even given that a particular instance should be generalized, we may still have a problem in deciding what aspects of the instance should be subject to generalization, and indeed how to control the process that creates the initial explanation. DeJong does not address these problems directly. Due to the state of his knowledge-base, he is able to generalize everything and explain everything. Since he has a very complete domain model, he can make use of existing story understanding methods, as described above,

to derive the initial representation. So, for example, though his system must explain why the daughter of a wealthy businessman is a plausible kidnap target, it presumably does not try to explain why an event that involved a young girl was a kidnapping. I.e., it does not start with the concept of a young girl and try to explain why that implied she was likely to be a kidnap target, since that would violate its detailed knowledge of intentionality.

In the next section, we will show how one can deal with the problems of constructing an initial explanation and determining what parts of it to generalize, and then return to the issue of deciding when to generalize.

3 EBL with Less Information -- Predictability

As we have suggested, the main problems with EBL arise in domains where minimal domain knowledge is available. Such domains are typical in SBL. To show how these problems can be dealt with by integrating EBL with SBL, we will employ a domain used by a typical SBL program, UNIMEM (Lebowitz, 1983b). UNIMEM is a program that takes a stream of facts about objects in a domain and organizes them into a long-term generalization-based memory with specific instances stored in terms of generalized concepts (Lebowitz, 1980; Lebowitz, 1982; Schank, 1982; Lebowitz, 1983a; Lebowitz, 1983b). UNIMEM learns by observation, and is neither specifically provided with a set of concepts to learn nor with "teacher-prepared" sequences of instances. It creates a hierarchy of new concepts by noticing similar instances and assuming that the similarities represent regularities in the domain.

One domain that we have used UNIMEM on involves information about congressional voting records.[4] This domain consists of information about the voting records of United States congressmen and the states and districts they represent. The primary information is the voting record of congressmen on 15 major issues taken from the *1982 Almanac of American Politics*. This information is augmented with a variety of facts about the states and districts where the congressmen reside. The kinds of generalizations we would expect to find would relate votes with each other (e.g., congressmen who oppose cutting the MX missile also oppose general cuts in defense spending), or that relate votes to

[4]Other domains that we have used UNIMEM on include information about the states in the United States, descriptions of computer software, descriptions of spiders, and football plays.

features of districts (e.g., congressmen from high-income districts support tax cuts). The 15 votes we used are described in Figure 1, along with a voting summary for the congressmen in a random 50 district sample used for examples in this paper. The numbers in parentheses are the votes of the entire House of Representatives.

For	Against	Absent	Vote Name	Description
33	15	2	alaska-parks	Create parks in Alaska (268F-157A)
22	21	7	chrysler	Guarantee loan to Chrysler (252F-141A)
29	18	3	draft	Register males for draft (219F-180A)
28	21	1	education	Create dept of education (210F-206A)
23	25	2	fair-housing	Enforce fair housing (205F-204A)
20	28	2	food-stamp-cap	Cap food stamp money (146F-276A)
23	26	1	gas-cont-ban	Prohibit gasoline price control (189F-225A)
18	32	0	hosp-cost-cont	Hospital cost containment prog (166F-234A)
16	34	0	MX-cut	Reduce MX appropriations (152F-250A)
22	24	4	nicaragua-ban	Ban aid to Nicaraguan government (189F-221A)
17	30	3	nuc-power-halt	Stop new nuclear plants (135F-254A)
23	27	0	osha-cut	Cut OSHA money (177F-240A)
25	25	0	PAC-limit	Limit PAC contributions (217F-198A)
20	28	2	soc-fund-cut	Switch social funds to defense (164F-264A)
34	16	0	wind-tax-lim	Limit windfall profits tax (236F-183A)

Figure 1: Descriptions of the votes

Figure 2 shows two generalizations and the instances they describe taken from a run of UNIMEM on the information about the subset of 50 congressmen.

Instances and generalizations in UNIMEM are described in terms of sets of features. The first generalization in Figure 2, GEN5, was formed by noticing districts with similar features. It describes congressional districts with congressmen who voted "no" on the bills about a nuclear power halt and hospital cost containment, "yes" on bills about the Nicaragua ban and windfall profits for oil companies, where farm value is high (in the fifth of six categories; Lebowitz (1985) describes the categorization method), where population went up between 1970 and 1980, and where the minority population is relatively low. This generalization describes one district directly (the Pennsylvania 22nd) and a number of others indirectly under more specific versions of this generalization. GEN6 is one such more specific generalization. It describes moderately high income districts with Republican congressmen who voted in a particular way on several bills. GEN6 describes the Pennsylvania 15th directly, and several other districts indirectly. The numbers in parentheses in Figure 2 indicate UNIMEM's confidence in each

```
GEN5
NUC-POWER-HALT VOTE            NO          [1]     (13)
STATE          FARM-VAL        FAR5:6      [1]     (20)
NICARAGUA-BAN  VOTE            YES         [2]      (0)
HOSP-COST-CONT VOTE            NO          [4]     (14)
DISTRICT       POP-DIR         UP          [4]     (26)
WIND-TAX-LIM   VOTE            YES         [4]     (18)
STATE          MINORITY-PCT    MIN1:2      [8]     (40)
[PENNSYLVANIA22]
{ALABAMA2 CALIFORNIA14 CALIFORNIA26 CALIFORNIA34 FLORIDA6 GEORGIA1
 GEORGIA6 KENTUCKY1 KENTUCKY4 MISSISSIPPI5 NORTHCAROLINA10 PENNSYLVANIA15
 TEXAS6 TEXAS22 VIRGINIA1 VIRGINIA2 VIRGINIA4 VIRGINIA6}

   GEN6
   FOOD-STAMP-CAP VOTE            YES         [1]     (13)
   STATE          INCOME          INC3:4      [1]      (6)
   STATE          SEATS           GAINED      [1]      (4)
   CHRYSLER       VOTE            NO          [2]      (9)
   GAS-CONT-BAN   VOTE            NO          [2]      (9)
   SOC-FUND-CUT   VOTE            YES         [2]     (12)
   OSHA-CUT       VOTE            YES         [2]     (14)
   CANDIDATE      PARTY           R           [2]      (8)
   PAC-LIMIT      VOTE            NO          [2]     (16)
   FAIR-HOUSING   VOTE            NO          [3]     (13)
   [PENNSYLVANIA15]
   {CALIFORNIA14 CALIFORNIA26 CALIFORNIA20 CALIFORNIA34 FLORIDA6
    TEXAS6 VIRGINIA1 VIRGINIA4}

  Two UNIMEM generalizations -- GEN5, made up of seven features
 (NUC-POWER-HALT VOTE NO, STATE FARM-VAL FAR5:6, etc.) and GEN6, a
 sub-generalization that inherits all those features, and includes
  10 others (FOOD-STAMP-CAP VOTE YES, STATE INCOME INC3:4, etc.).
 Instances stored directly with the generalization are in brackets
(PENNSYLVANIA22 for GEN5) and those stored under a generalization's
 sub-generalizations are in braces. The instances stored under each
generalization also inherit these features. The numbers in brackets
  are predictability information, while the numbers in parenthesis
                    are confidence levels.
```

Figure 2: A piece of UNIMEM memory

feature (the numbers start at 0 and can rise or fall). The numbers in brackets are predictability information, which will be explained later.

The EBL programs in the literature would not approach this domain by looking at how voting records of various congressmen compare to each other, as UNIMEM does. Presumably they would start by looking in detail at the information from a given congressional district. The information available to UNIMEM for one such district, in the form of 32 features about the district, the state it is located in, and the votes of its congressman on a number of issues, is shown in Figure 3.

```
Features: PENNSYLVANIA22 (DISTRICT)
CANDIDATE        OCCUPATION    LAW             | DISTRICT        INCOME        INC2:4
DISTRICT         POP-DIR       UP              | CANDIDATE       PARTY         D
STATE            IS            PENNSYLVANIA    | DRAFT           VOTE          YES
NICARAGUA-BAN    VOTE          YES             | MX-CUT          VOTE          NO
NUC-POWER-HALT   VOTE          NO              | ALASKA-PARKS    VOTE          NO
FAIR-HOUSING     VOTE          YES             | PAC-LIMIT       VOTE          NO
FOOD-STAMP-CAP   VOTE          NO              | EDUCATION       VOTE          YES
OSHA-CUT         VOTE          NO              | SOC-FUND-CUT    VOTE          NO
HOSP-COST-CONT   VOTE          NO              | GAS-CONT-BAN    VOTE          YES
WIND-TAX-LIM     VOTE          YES             | CHRYSLER        VOTE          YES
STATE            SEATS         LOST            | STATE           REGION        MA
STATE            POPULATION    POP6:7          | STATE           URBAN-PCT     URB6:6
STATE            MINORITY-PCT  MIN1:2          | STATE           MIGRATION     MIG1:9
STATE            SIZE          SIZ3:6          | STATE           SCHOOL-EXP    SCH3:3
STATE            CRIME-RATE    CRI2:5          | STATE           STATE-DEBT    DEB6:7
STATE            MILITARY-$    MIL7:9          | STATE           INCOME        INC3:4
STATE            FARM-VAL      FAR5:6          | STATE           TAXES         TAX2:5
```

Figure 3: The Pennsylvania 22d

An EBL program, such as DeJong's, would first build up a causal analysis of the various features of the input, using whatever reasoning rules were available. Then, it would determine how properties of the features could be generalized such that the causal analysis would still hold up. So, for example, the program might decide the Pennsylvania 22nd's congressman voted against the MX-cut because military spending in the district was high. Then it would see just how high the military spending had to be for the causal explanation to hold.

This approach might be appropriate if we had very thorough information about the domain and could construct a detailed causal explanation, particularly if there were a only limited number of points to vary during the EBL generalization phase (as that would limit the analysis we would have to do with the explanation). However, if we have only very general rules to apply, as is often the case in a new domain, then the explanation process would not be combinatorially feasible, particularly as it must be applied to many modified versions of the instance as constraints are relaxed.

What we propose is, first, to apply EBL methods to inductively created *generalizations*, rather than individual instances or episodes. This means that we will wait for SBL methods to suggest generalizations that will then be analyzed by EBL methods (i.e., a causal explanation will be derived and constraint-loosening performed). At the very least, this will avoid analyzing instances that are totally atypical (as they will not take part in SBL generalizations). This is similar to the way that Lenat has used

the idea of "worth" to control learning by discovery in AM (Lenat, 1982) and EURISKO (Lenat, 1983) (see Section 5). There is a larger advantage, however.

Causal explanation involves determining why a given set of conditions (causes) leads to an observed behavior (results). In order to do this in a domain where we have limited knowledge, we must first identify which elements of an instance are causes, and which are results. Doing so will provide a focus for applying general rules to come up with an explanation of the instance. Most EBL systems determine the explanation in a fashion unrelated to the generalization phase, and need not deal with this problem. For example, DeJong bases his EBL on a causal explanation of the sort provided in (Carbonell, 1981; Schank and Abelson, 1977; Wilensky, 1983), based on detailed knowledge of human intentionality. The rules about human intentionality used in such methods imply the causes in a situation (e.g., human intentional actions).

If we look at generalization GEN5 in Figure 2, we see that identification of causes is not trivial. For example, it might be that districts with high farm property values are thought to have oil reserves and hence their congressmen would vote to limit any windfall profits tax. Conversely, it might be that voting to limit the windfall profits tax actually causes the farm value to be high, as potential investors would know oil profits would not be subject to high taxes.

Our solution to this problem is to use *predictability* (presented in (Lebowitz, 1980; Lebowitz, 1983a) for indexing and understanding purposes). The basic idea is that in a given context, features of a generalization that are most nearly unique to that generalization indicate its applicability.[5] These features are called *predictive*. Most importantly here, the predictive features are exactly those that are likely to be causes in a causal explanation. This follows from the observation that non-predictive features occur in many generalizations, and are associated with many different combinations of other features. Hence, they do not predict a single outcome. To take the simplest possible case, if a generalization was made up of two features, A and B, and A occurred in one generalization and B in many, B could not cause A. If it did, A would also appear in all the other generalizations that B was in.

---

[5]Predictability can be viewed as an operational definition similar to the concept of cue discriminability used in perceptual categorization, e.g., (Restle, 1962).

As a further example of predictability, if we noticed that all AI conferences were exciting, we would assume that a conference being about AI causes it to be exciting. We probably have few generalizations about AI conferences, as opposed to assuming that because an event was exciting it was probably an AI conference.[6] (See (Goodman, 1965; Hempel, 1943) for related philosophical discussion.)

If we return to Figure 2, we can see how predictability might be used. The numbers in brackets next to each feature indicate how many of the generalizations under the generalization's parent node involve that feature. So, in GEN6, the "yes" vote on the food stamp cap does not appear in any generalization under GEN5 other than GEN6. On the other hand, a "no" vote on fair housing appears in three generalizations under GEN5. The figures for the features in GEN5 reflect the generalizations under its parent node.

Using the predictability information from GEN5, we can see that an explanation should be formed that shows why features like a "no" vote on the nuclear power halt, high farm value and possibly a "yes" vote on the Nicaraguan ban issue imply the remaining features. The other features will certainly work less well as the causes in an explanation, since they are associated with a variety of different features in other generalizations. An EBL program trying to explain GEN6 should look for rules that indicate why a congressman from a state with fairly high income that gained congressional seats and who voted "yes" on a food stamp cap (the predictive features, which we assume to be causes) should be a Republican who voted against the Chrysler guarantee, against gas controls, etc.

In developing a causal analysis, *only* predictive features may be causes (or be indicative of causes), although not every predictive feature need be causes. Non-predictive features having causal impact would cause contradictions, as such features co-occur with a variety of other features (i.e., they cannot consistently cause one set of features). With the predictive features as a causal starting point, we can apply general plan/goal-based understanding methods such as those in (Carbonell, 1981; Schank and Abelson, 1977; Wilensky, 1983), or whatever explanation methods seem appropriate.

---

[6]Note that if we knew a number of things about AI conferences, that they are usually in the summer, in interesting locations, have papers in a number of areas, perhaps, then these facts would form a single conjunctive generalization.

Predictability can also be applied in analyzing specific instances (should we wish to do so). If we wanted to apply EBL to the Pennsylvania 22nd district, for example, in the manner of current EBL programs, then we could use the predictability of the generalization it is stored with (GEN5, in this case) to provide a starting point for the analysis. While such analysis may still be difficult, as any single instance might be anomalous in some way, at least the search will not be totally unconstrained. Note that if the instance is stored in several places in memory (which is possible, since generalizations in UNIMEM are not viewed as being mutually exclusive), then several possible explanations might be generated.

The point here is simply that in any given situation there are variety of different features or effects we could try to explain. Predictability provides a focus for application of general explanatory rules, especially for domains with limited amounts of available world knowledge. Many problems remain in applying predictability, most notably how to deal with combinations of features that are predictive even when none of the individual features are, but predictability appears to provide useful clues in building up knowledge of a domain.

4 Applying Predictability -- An Example

The use of predictability in the EBL process can best be seen with an example. We will use for illustrative purposes a simple "backward chaining" explanation mechanism that we have implemented which applies simple rules to the generalizations that are made by UNIMEM in the congressional voting domain. Our initial explanation implementation focuses on how predictability can be used to help construct an initial causal explanation of a generalization. Further work is needed to show how this explanation can then be used during later SBL processing.

To apply explanation-based methods, we must supply rules that capture our initial understanding of the domain. The obvious way to do this for UNIMEM is with implications that capture hypothesized low-level causal connections among features. We have rules that indicate that the presence of one feature(s) causes the presence of another feature(s), i.e., F_c -> F_r. Such rules can be used in understanding the causality underlying a set of features in one of two ways: 1) from the presence of one feature (F_c) we "explain" the presence of another (F_r); 2) from the known absence of one feature (F_r) "explain" the absence of another feature (F_c) whose presence would have forced the presence of the

first.

The second usage of our rules is particularly important in "closed-world" domains, like the congressional votes in our example. Absences are easy to detect; a "yes" vote by a given congressman indicates conclusively that a "no" vote did not occur. For example, one of our rules is that "a congressman from a state with a major defense industry will vote against the MX-cut". Using this rule, we can explain a "no" MX-cut vote from a defense industry in the congressman's state. We can also "explain" the lack of a defense industry from a "yes" MX-cut vote, although the complete underlying causality is, of course, more complicated. We cannot, though explain a "yes" MX-cut vote from the absence of a defense industry (the rules are one-directional).

Figure 4 illustrates a set of rules used in the experiments described here. It is important to recognize that in this paper we are primarily trying to indicate how causal explanation rules can be applied to SBL learning. While we tried to make the rules plausible, their details are not critical to this presentation. Although the rules were not specifically created to explain just this one generalization, they were created with an eye on a small number of examples. We will discuss briefly below how a more robust set of such rules might be created.

The rules in Figure 4 are quite simple from an implementation standpoint. Each rule indicates that if the features on the left of the "=>" describe a Congressional District then they can be used to plausibly explain the features on the right. The first rule, for example, indicates that we believe that a vote for parks in Alaska can be explained by the presence of pro-wildlife voters in the congressman's state. Multiple features on either side of the implication indicate conjunction. (Disjunction is handled with multiple rules.)

The rules in Figure 4 indicate a believed direction of causality for relations among features. They reflect an informal level of explanation that people often use. The rules may hide a number of steps in the true underlying causal mechanism. This is quite important when we are using the rules in a contrapositive sense, since a full explanation then involves what would have happened in other cases.

Representing rules as simple causal implications seems reasonable as a first approximation. It

```
(STATE VOTERS PRO-WILDLIFE) => (ALASKA-PARKS VOTE F)
(STATE TYPE RURAL) (STATE INDUSTRY LOWTECH) => (STATE VOTERS PRO-WILDLIFE)
(STATE TYPE RURAL) => (STATE SIZE (< SIZ4:6))
(STATE INDUSTRY DEFENSE) => (MX-CUT VOTE A)
(DISTRICT TYPE BOOMDIST) => (DISTRICT POP-DIR UP)
(DISTRICT TYPE BOOMDIST) (STATE INCOME (> INC2:4)) =>
   (STATE FARM-VAL-PER-ACRE (> FAR4:6))
(DISTRICT TYPE BOOMDIST) (STATE DEBT (> DEB4:7)) => (STATE INCOME (> INC2:4))
(DISTRICT TYPE BOOMDIST) => (STATE MIDDLECLASS YUPPIE)
(STATE MIDDLECLASS YUPPIE) => (SOC-FUND-CUT VOTE A) (OSHA-CUT VOTE A)
(STATE VOTERS PRO-EDUCATION) => (STATE VOTERS PRO-WILDLIFE)
(STATE TYPE URBAN) => (STATE TAXES-PERCAP (> TAX2:5))
(STATE INCOME (<= INC2:4)) => (STATE TAXES-PERCAP (> TAX2:5))
(NOT (STATE INDUSTRY DEFENSE)) => (STATE TAXES-PERCAP (> TAX2:5))
(STATE INDUSTRY HIGHTECH) => (STATE SEATS GAIN)
(STATE VOTERS ANTI-EDUCATION) => (EDUCATION VOTE A)
(STATE INCOME (< INC3:4)) => (NOT (STATE MIDDLECLASS YUPPIE))
(DISTRICT TYPE STATIC) => (PAC-LIMIT VOTE A)
(STATE INCOME (< INC3:4)) => (STATE SCHOOL-EXP (< SCH3:3))
(STATE SIZE (> SIZ3:6)) => (STATE SCHOOL-EXP (< SCH3:3))
(STATE VOTERS ANTI-EDUCATION) => (STATE SCHOOL-EXP (< SCH3:3))
(NOT (STATE MIDDLECLASS YUPPIE)) => (FOOD-STAMP-CAP VOTE F)
(NOT (DISTRICT PHILOSOPHY HIGH-TECH)) => (WIND-TAX-LIM VOTE A)
(NOT (DISTRICT PHILOSOPHY FREE-ENT)) => (HOSP-COST-CONT VOTE F)
(DISTRICT PHILOSOPHY FREE-ENT) (DISTRICT PHILOSOPHY HIGH-TECH) =>
   (DISTRICT TYPE BOOMDIST)
=> (STATE MIN-PCT MIN1:2)
```

Figure 4: Rules about the CD voting record domain

corresponds to production system methodology (Newell, 1973; Waterman and Hayes-Roth, 1978). Knowledge about the domain is represented in the form of small chunks that can be combined in various ways. The primary goal of the explanation task then becomes to combine these simple rules to build up causal representations that describe larger conceptual situations that can be generalized.

Note that we do *not* view the rules as being guaranteed correct. Rather, they are viewed as heuristic and tentative, representing our best current view of the domain. This would be the best we could do for many domains. In the long run, we expect to introduce a strong feedback mechanism that would use observed generalizations to help evaluate the rules, even while we use the rules to help explain the generalizations. Obviously, this would introduce a strong element of subjective understanding in the sense of Abelson (1973) or Carbonell (1981).

With initial rules in hand, UNIMEM can engage in EBL. For the moment, we ignore the problem of deciding when during SBL a generalization should be analyzed using EBL, and simply show how we would process a typical generalization. Figure 5 shows one of the generalizations made by UNIMEM in

the same run of the program mentioned in Section 3. It describes congressional districts that gained population since the last census, that are located in medium-sized states with low tax rates, high farm value, and low minority population and where the congressman voted "yes" on the windfall profits and draft votes and "no" on the hospital cost containment and MX-cut votes. As before, the numbers in brackets are predictability information (how many generalizations the feature appears in) and the numbers in parentheses are confidence values.

```
GEN1
WIND-TAX-LIM    VOTE          F          [1]    (16)
DRAFT           VOTE          F          [1]     (9)
HOSP-COST-CONT  VOTE          A          [2]    (12)
MX-CUT          VOTE          A          [2]    (16)
STATE           TAXES         TAX2:5     [2]    (18)
DISTRICT        POP-DIR       UP         [3]    (26)
STATE           FARM-VAL      FAR5:6     [4]    (20)
STATE           SIZE          SIZ3:6     [4]    (10)
STATE           MIN-PCT       MIN1:2     [6]    (44)
```

Figure 5: Another typical generalization -- GEN1

As mentioned earlier, an EBL analysis of GEN1 would begin by developing a plausible causal explanation of the various features. Even if we restrict ourselves to the rules in Figure 4, finding such an explanation would be non-trivial, as many different rules would apply at each stage of the analysis. Indeed, in some cases, the rules may be mutually contradictory, due to their heuristic nature. For this reason, plus efficiency considerations, we must control the explanation process. As presented in Section 3, we use predictability to provide this control.

We begin our analysis by assuming that all the features less than or equal to a given threshold (two, here) are predictive (the four votes and low tax rate in GEN1), and hence potential causes in our explanation. We assume that all the rest of the features of GEN1 (the remaining state features and the decrease in population) are non-predictive, and should be explainable from the predictive features. UNIMEM then uses simple backwards chaining methods to find causal chains that connect the predictive features to the non-predictive ones. Figure 6 shows the output from this analysis. The features marked with "c" are the assumed causes (determined by predictability); the results (non-predictive features) are marked with "r". The rules marked with *'s are the contrapositive forms of the original rules. That is, the absence of the right side of the rule (usually the presence of a contradictory feature, often an opposite

vote) is being used to explain the negation of the left side.

```
Nothing left to prove
Final rule chain:
=> (STATE MIN-PCT MIN1:2)r
(STATE TYPE RURAL) => (STATE SIZE (< SIZ4:6))r
(DISTRICT TYPE BOOMDIST) => (DISTRICT POP-DIR UP)r
(STATE TAXES-PERCAP (<= TAX2:5))c => (NOT (STATE TYPE URBAN))   [*]
(DISTRICT TYPE BOOMDIST) (STATE INCOME (> INC2:4)) =>
   (STATE FARM-VAL-PER-ACRE (> FAR4:6))r
(STATE TAXES-PERCAP (<= TAX2:5))c => (STATE INCOME (> INC2:4))   [*]
(DISTRICT PHILOSOPHY FREE-ENT) (DISTRICT PHILOSOPHY HIGH-TECH) =>
   (DISTRICT TYPE BOOMDIST)
(WIND-TAX-LIM VOTE F)c => (DISTRICT PHILOSOPHY HIGH-TECH)   [*]
(HOSP-COST-CONT VOTE A)c => (DISTRICT PHILOSOPHY FREE-ENT)   [*]
Unused causes -- (MX-CUT VOTE A) (DRAFT VOTE F)
```

Figure 6: Causal analysis of GEN1

The output in Figure 6 shows all the applicable rules needed to explain the non-predictive features in GEN1 from the predictive ones. Note that neither the MX-cut nor the draft votes had to be used in constructing the explanation. We will return to this in a moment. The chaining process involves the construction of a number of explanatory causal chains simultaneously. In order to make the explanations in Figure 6 clearer we have displayed them graphically in Figure 7. (Again, "c" and "r" mark causes and results.) The links marked with "*" are those where the contrapositive of rules has been used, so that the full underlying causality would involve prevention of certain states from occurring.

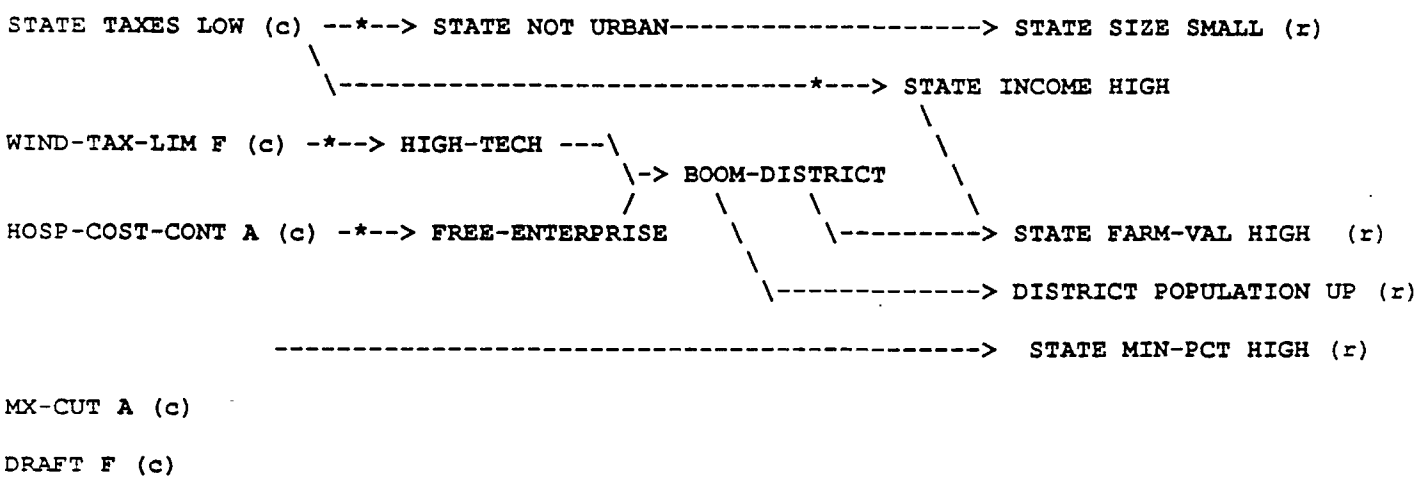

Figure 7: The analysis of GEN1, graphically

Figure 7 makes clear that we have found the kinds of relationships we are looking for. We see that

since GEN1 describes low tax states, the states are probably rural, small and high-income. The windfall and hospital cost containment votes would only be cast by congressmen from districts that are high-tech-conscious and pro-free-enterprise. Such districts would normally be "boom" areas. This, in turn, implies that the districts are likely to have high farm value and rising population (due to the "high-tech" boom). The low minority level is simply a default (46 of the fifty states fall in this class). These cascaded rules then explain all the non-predictive features of GEN1. If it seems a little strange that the explanation has several votes at the beginning of the causal chains (as if they directly caused properties of the states), notice that in each case the relevant rules are being used in contrapositive form, meaning that there are really underlying factors inhibiting the opposite vote.

As noted earlier, the MX-cut and draft votes do not appear in the explanations of any other features. There are two possible reasons for this: these features may be irrelevant to the generalization, the result of a coincidence that has not been identified by UNIMEM's confidence evaluation methods (Lebowitz, 1982; Lebowitz, 1983a; Lebowitz, 1983b), or the features may be explainable by the other predictive features. Their appearance in only a small number of generalizations would then be due to lack of data -- i.e., they are not really predictive. To test this second possibility, we have UNIMEM add the unused features to the set of potential results, and redo the explanation process. The results of this re-application of the rules is shown in Figure 8 and illustrated graphically in Figure 9.

We can see from Figure 9 that the MX-cut vote is indeed explainable from the other predictive features (low taxes imply a pro-defense state which explains the MX-cut vote). On the other hand, the draft vote still remains causally unconnected, which means either that it does not belong in the generalization, or should be connected by a rule unknown to the system.

The current version of UNIMEM does not follow up on the causal explanation that it has built. If we were to continue the EBL process, we would use the explanation built for the specific generalization and see if we could abstract it and determine the essential features of the explanation. So, for example, we might infer that the windfall tax limit vote is not necessary to conclude that a district is interested in high-tech, but instead any positive vote on limiting taxes is satisfactory. We would, of course, need further

```
Reprocessing with unused causes as results

No more valid rules
Final rule chain:
=> (STATE MIN-PCT MIN1:2)r
(STATE INDUSTRY DEFENSE) => (MX-CUT VOTE A)r
(STATE TYPE RURAL) => (STATE SIZE (< SIZ4:6))r
(DISTRICT TYPE BOOMDIST) => (DISTRICT POP-DIR UP)r
(STATE TAXES-PERCAP (<= TAX2:5))c => (NOT (STATE TYPE URBAN))   [*]
(DISTRICT TYPE BOOMDIST) (STATE INCOME (> INC2:4)) =>
   (STATE FARM-VAL-PER-ACRE (> FAR4:6))r
(STATE TAXES-PERCAP (<= TAX2:5))c => (STATE INCOME (> INC2:4))   [*]
(STATE TAXES-PERCAP (<= TAX2:5))c => (STATE INDUSTRY DEFENSE)   [*]
(DISTRICT PHILOSOPHY FREE-ENT) (DISTRICT PHILOSOPHY HIGH-TECH) =>
   (DISTRICT TYPE BOOMDIST)
(WIND-TAX-LIM VOTE F)c => (DISTRICT PHILOSOPHY HIGH-TECH)   [*]
(HOSP-COST-CONT VOTE A)c => (DISTRICT PHILOSOPHY FREE-ENT)   [*]

Unexplained results: (DRAFT VOTE F)
Unused causes -- none
```

Figure 8: Reanalyzing GEN1

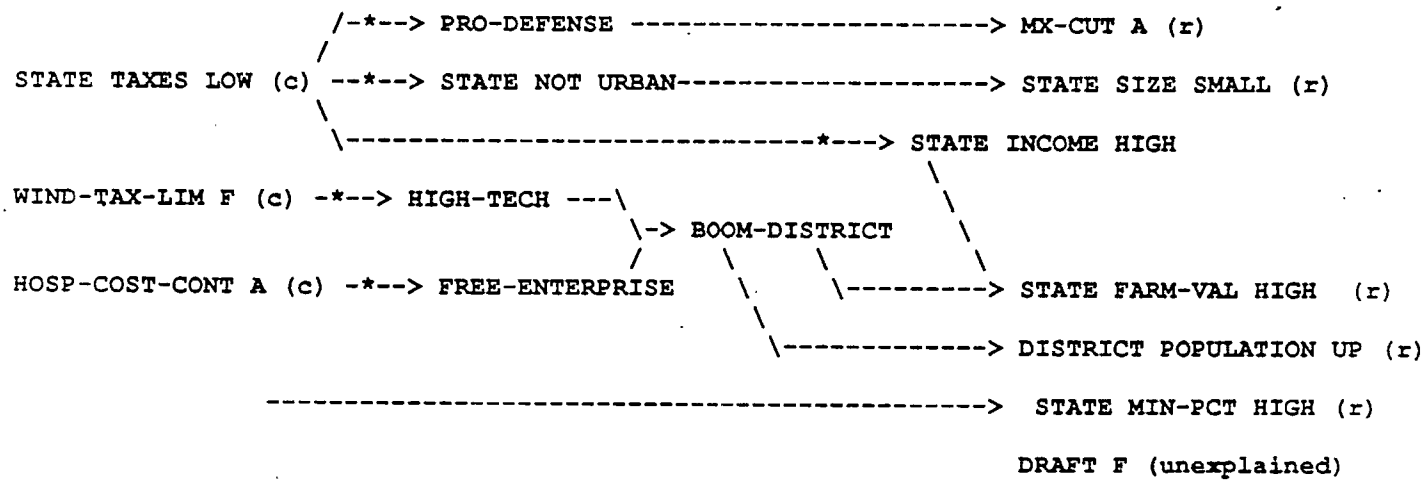

Figure 9: The re-analysis of GEN1, graphically

pre-supplied knowledge to allow us analyze in this way (although our existing rules can be used to some degree; for example, the rules that require only a range of values, e.g., less that than the third category out of 6, would allow us to relax a specific category to a range).

The main point here is that we have constructed an explanation adequate to apply all the EBL techniques discussed in Section 2 (at least as far as our rules allow).

There are several ways we could use our explanatory analysis within UNIMEM, or indeed SBL in general. The most obvious is to drop from a generalization any features that were involved in our

analyses (e.g., the draft vote in our example). This adds a new way to detect coincidence in generalizations. In addition, features that we thought were predictive, but did not explain anything else, and were explainable themselves (e.g., the MX-cut vote in our example), could be marked as non-predictive, even though they appear to be predictive from their frequency in different generalizations. This has a significant effect on the application of generalizations to new instances, as only predictive features can indicate a generalization's potential relevance. In this case, a "no" MX-cut vote would no longer be used to indicate the applicability of GEN1. We can also imagine splitting up generalizations based on the explanations, and other uses involving the details of the analysis. To properly implement these ideas would require building a full set of explanatory rules covering all the features in the domain and developing an algorithm for deciding when to apply explanatory analysis.

A final way to use our explanation was mentioned earlier -- we could evaluate the reliability of our initial, tentative rules. Our view is that the initial rules for a domain would be hand-coded, as was done for the example above. Later rules could abstracted from generalizations in which we have high confidence. Then, if we used a confidence scheme for rules similar to UNIMEM's confidence levels for generalizations, we would increase our confidence in the rules used to build up explanations, but decrease confidence in rules that might have been applied but were not, particularly if they would have given wrong results (as opposed to just being irrelevant). More complex schemes involving analysis of exactly what went wrong in applying each rule would also be possible.

There are several important points to be gleaned from the example in this section. Predictability provided significant control on the explanation process. We did not have to use brute force and try all the possible explanatory rule sequences. If we had a more detailed knowledge of the domain with very specific explanatory rules, this would not be so important, but in a new domain, where rules are very general and perhaps contradictory, it is crucial. In addition, we can see how the SBL and EBL processes naturally complement each other. SBL gives us generalizations to explain and help control the explanation. The explanation, in addition to the main EBL purpose of detailed understanding, can be used to make further SBL processing more efficient. Among the many problems that need to be solved to make full use of this synergy are deciding when to apply EBL, how the interaction works when the explanations are more detailed than the one we used in our example, and how to use the internal

features of a detailed explanation.

5 Further Control -- Interest

Even having taken predictability into account, an EBL system will still have a large amount of work to do. We have the problem of deciding when to generalize, and the explanation process could still use further control. One way that people deal with both of these problems is to focus on instances that seem *interesting* to them, and the parts of the instances that are interesting. As pointed out in (Lebowitz, 1981), the interesting instances are exactly those that are likely to lead to successful learning. While we do not plan in this paper to present an entire theory of what makes something interesting, we will 1) define interest in a way that is useful for the task at hand; 2) describe in more detail how interest can be applied to our combination of SBL and EBL; and 3) indicate the plausibility of determining interest. By necessity, our presentation will be somewhat general, hopefully stimulating further research in this important area.

5.1 Defining interest

Saying that interesting instances and interesting parts of instances are useful in learning appears almost tautological. Some researchers have actually defined interest in terms of what helps in learning. Davis (1971) in a comparative philosophy of science study of what constitutes interesting sociological theories did just this. However, such an approach would not help us, as we would have to carry out the learning process before being able to apply interestingness. If we wish interest to be an active part of a computational model, we will have to assume that interest is a *heuristic* measure of what is likely to help in terms of learning. This is opposed to simply treating interest as a post hoc property of a memory structure.

We will, then, make use of an intuitive feel for what makes something interesting. We will ultimately develop this into one or more heuristics for use in learning. This replaces an attempt to look for a guaranteed metric of what makes a good learning instance.

5.2 Using interest

If we look back at STORY1, the DeJong kidnapping example, and at the various UNIMEM examples presented in this paper, we can see how interest can provide useful control. DeJong, for his kidnapping example, has already applied a set of heuristics which include a form of interest, to decide that the story as a whole is interesting. Nonetheless, we could still apply the ideas of interest further. Specifically, to help further control the search process, we would want to limit the number of features in the story that we actively look at when generalizing features. This is particularly important if there is significant interaction among the various features we might generalize. So, while we certainly want to worry about the amount of money being extorted by the kidnapper from the businessman (it is expected to be large, but not exactly 1 million francs), we might not worry about the details of the communication between the kidnapper and his victim (of course, some people might -- interest being idiosyncratic). It is not that we would then assume the communication must be by telegram, but rather we would generalize the form of communication without doing a detailed feasibility analysis. This is because our heuristics presumably show that the amount of money is interesting, while the form of communication is not. If the form of communication was of interest, we might analyze further to discover that it is important that the communication not be face-to-face.

For the UNIMEM example in Figure 2, we have two potential ways to apply interest. Unlike DeJong's program, but common to the state of many learning domains, we do not have a straightforward set of heuristics to tell us when to apply EBL. So, we will want to make use of interest. Actually, the fact that we are looking at generalizations instead of instances is one application of interest -- we are assuming that generalizations are more interesting (because they are more reliable) than individual instances. The second application of interest would be, as in the DeJong example, to decide how to focus the EBL process.

5.3 Determining interest

It clearly makes little sense to discuss the heuristic use of interest if we cannot hope to measure it in a computationally feasible way. While we will not look formally at the components of a heuristic measure of interest in this paper, we will indicate why we consider the computation of such a measure plausible.

In the work done on the use of interest in learning, probably the most complete description of an interest measure is that of Lenat for two programs that learn by discovery, AM (Lenat, 1982) and Eurisko (Lenat, 1983). Associated with each concept in the programs (both initial and derived) is a "worth" level -- a number that specifies how likely it is that further exploration will find more useful concepts. These values can change as the universe of concepts change. Each concept also has an "interest" slot that indicates how to determine the worth of new concepts formed using the given concept. (E.g., the "compose" concept's interest slot shows how to find the worth of concepts formed by composing functions.)

Lenat's interest heuristics are rather specialized for the domains at hand, mathematics in AM's case. While the work on Eurisko involving heuristics that modify heuristics may help in this regard, we prefer to look for simpler, more general heuristics that depend for their power on the richness of our memory structures. This will be particularly important for EBL systems that make use of complex knowledge bases.

Schank (1979) and Lebowitz (1981) have discussed the applicability of interest in relation to complex memory structures. In (Lebowitz, 1981) we indicated that various properties such as relevance and novelty do make the determination of interest computationally feasible (in particular, by focusing on heuristics that indicate when a concept is *not* interesting). Several interest heuristics based on the ideas of relevance and novelty used in (Lebowitz, 1981) would be applicable for deciding which generalizations to analyze. We would want to concentrate on generalizations that describe a number of instances, rather than just a few, and perhaps those that involve an unusual set of features. In addition, we would prefer generalizations that organize other generalizations, as they have wider applicability. So, looking back at Figure 2, we would be more interested in GEN5 than GEN6 since it describes more instances (as well as having a number of more specific generalizations). Should it turn out that GEN6 is the only generalization involving congressmen voting "yes" on the food stamp cap bill and "no" on the Chrysler bailout, then it would be more interesting, because it is novel. Note that this is just what we want, since new combinations of features are likely to lead to new concepts.

Interest rules for deciding which features of a generalization to focus the explanation process on

would be similar. We would tend to focus on explaining features that are novel, but not too novel. Novel, since otherwise we can presumably just use existing explanations, but not too novel, since we want to relate the explanations and generalizations that we derive to other parts of our knowledge base.

Note that the interest heuristics described here, as well as most of the others one can think of (at least those that do not use pre-existing domain knowledge), crucially depend on having a sizable number of instances in memory, and hence indicate a connection between SBL and EBL. If we were developing a learning system with user-imposed outside interests (e.g., "be interested in votes about defense"), we could combine these interests with the more general heuristics to develop a system that makes generalizations that are relevant to a specific user.

To recapitulate, interest is a very intuitive idea that leads to many useful processing heuristics. If we apply these heuristics to the learning process, they will help focus processing on the items that accelerate learning most efficiently. We need not have a detailed understanding of why each heuristic helps the learning process to make use of interest. Although we are only proposing methods of applying interest at this point in time, we feel that the use of robust interest heuristics will be crucial in building large learning systems that combine SBL and EBL methods.

6 Conclusion

EBL methods hold the promise of developing learning systems that can make full use of the knowledge they already possess. However, it is necessary to relate these methods to SBL techniques so that our systems can not only make use of a priori knowledge, but also use similarities noticed among large numbers instances. This is particularly important in domains lacking detailed domain knowledge. We have described in this paper a three step plan involving:

- Applying EBL to generalizations derived by noticing similarities, instead of to individual instances.
- Using *interest* to determine when to learn.
- Using *predictability* to help control an otherwise unmanageable explanation process.

The integration of EBL and SBL methods can lead to robust learning systems that can both make use of existing knowledge and process large numbers of instances. This will help our systems deal with realistic, noisy data (Lebowitz, 1982; Lebowitz, 1983b). There are many issues to be addressed, some of

which we have suggested in this paper, on the road to learning systems that approach the power of human learners.

References

Abelson, R. P. (1973). The structure of belief systems. In R. C. Schank and K. Colby, (Ed.), *Computer Models of Thought and Language*, (pp. 287 - 340). San Francisco: W. H. Freeman Co.

Carbonell, J. G. (1981). *Subjective Understanding: Computer Models of Belief Systems.* Ann Arbor, Michigan: UMI Research Press.

Carbonell, J. G. (1983). Derivational analogy in problem solving and knowledge acquisition. *Proceedings of the 1983 International Machine Learning Workshop*, Champaign-Urbana, Illinois, pp. 12 - 18.

Cohen, P. R. and Feigenbaum, E. A. (Eds.). (1982). *The Handbook of Artificial Intelligence, Volume 3.* Los Altos, California: William Kaufmann, Inc.

Davis, M. S. (1971). That's interesting! Towards a phenomenology of sociology and a sociology of phenomenology. *Philosophy of the Social Sciences, 1*, 309 - 344.

DeJong, G. F. (1983). An approach to learning from observation. *Proceedings of the 1983 International Machine Learning Workshop*, Champaign-Urbana, Illinois, pp. 171 - 176.

Dietterich, T. G. and Michalski, R. S. (1983). Discovering patterns in sequences of objects. *Proceedings of the 1983 International Machine Learning Workshop*, Champaign-Urbana, Illinois, pp. 41 - 57.

Ellman, T. (1985). Generalizing logic circuit designs by analyzing proofs of correctness. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, pp. 643 - 646.

Goodman, N. (1965). *Fact, Fiction, and Forecast.* Indianapolis, Ind.: The Bobbs Merrill Company, Inc.

Hempel, C. G. (1943). A purely syntactic definition of confirmation. *Journal of Symbolic Logic, 8*, 122 - 143.

Lebowitz, M. (1980). *Generalization and memory in an integrated understanding system* (Tech. Rep. No. 186). New Haven, CT: Yale University Department of Computer Science.

Lebowitz, M. (1981). Cancelled due to lack of interest. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada, pp. 13 - 15.

Lebowitz, M. (1982). Correcting erroneous generalizations. *Cognition and Brain Theory, 5*, 367 - 381.

Lebowitz, M. (1983). Generalization from natural language text. *Cognitive Science, 7*, 1 - 40.

Lebowitz, M. (1983). Concept learning in a rich input domain. *Proceedings of the 1983 International Machine Learning Workshop*, Champaign-Urbana, Illinois, pp. 177 - 182.

Lebowitz, M. (1985). Classifying numeric information for generalization. *Cognitive Science, 9*, 285 - 308.

Lenat, D. B. (1982). AM: Discovery in mathematics as heuristic search. In R. Davis and D. B. Lenat, (Ed.), *Knowledge-Based Systems in Artificial Intelligence*, (pp. 1 - 225). New York: McGraw-Hill.

Lenat, D. B. (1983). EURISKO: A program that learns new heuristics and domain concepts. *Artificial Intelligence, 21*, 61 - 98.

Michalski, R. S. (1980). Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2*, 349 - 361.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence, 20*, 111 - 161.

Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.). (1983). *Machine Learning, An Artificial Intelligence Approach.* Los Altos, CA: Morgan Kaufmann.

Minton, S. (1984). Constraint-based generalization. *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 251 - 254.

Mitchell, T. M. (1983). Learning and problem solving. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, pp. 1139 - 1151.

Mostow, J. (1983). Operationalizing advice: A problem-solving model. *Proceedings of the 1983 International Machine Learning Workshop*, Champaign-Urbana, Illinois, pp. 110 - 116.

Newell, A. (1973). Production systems: Models of control systems. In W. G. Chase, (Ed.), *Visual Information Processing*, (pp. 463 - 526). New York: Academic Press.

Restle, F. (1962). The selection of strategies in cue learning. *Psychological Review, 69*, 329 - 343.

Salzberg, S. (1983). Generating hypotheses to explain prediction failures. *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 352 - 355.

Schank, R. C. (1975). The structure of episodes in memory. In D. Bobrow and A. Collins, (Ed.), *Representation and Understanding: Studies in Cognitive Science*, (pp. 237 - 272). New York: Academic Press.

Schank, R. C. (1979). Interestingness: Controlling inference. *Artificial Intelligence, 12*, 273 - 297.

Schank, R. C. (1982). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* New York: Cambridge University Press.

Schank, R. C. (1984). *The Explanation Game* (Tech. Rep. No. 307). New Haven, CT: Yale University Department of Computer Science.

Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Silver B. (1983). Learning equation solving methods from worked examples. *Proceedings of the 1983 International Machine Learning Workshop*, Champaign-Urbana, Illinois, pp. 99 - 104.

Waterman, D. A. and Hayes-Roth, F. (1978). *Pattern-Directed Inference.* New York: Academic Press.

Wilensky, R. (1983). *Planning and Understanding.* Reading, MA: Addison-Wesley.

Winston, P. H. (1972). Learning structural descriptions from examples. In P. H. Winston, (Ed.), *The Psychology of Computer Vision*, (pp. 157 - 209). New York: McGraw-Hill.

Winston, P. H. (1980). Learning and reasoning by analogy. *Communications of the ACM, 23*, 689 - 702.