CUCS-111-84

Concept Learning in a Rich Input Domain: Generalization-Based Memory

Michael Lebowitz

May, 1984

The research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165.

Concept Learning in a Rich Input Domain: Generalization-Based Memory[1]

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

Abstract

Automatic concept learning from large amounts of complex input data is an interesting and difficult process. In this paper we discuss how the use of a permanent, generalization-based, memory can serve as an important tool in developing programs that learn in rich input domains. The use of Generalization-Based Memory (GBM) allows programs to determine what concepts to learn, as well as definitions of the concepts. We present in this paper a characterization of our research, describe our use of Generalization-Based Memory in two programs under development at Columbia, UNIMEM and RESEARCHER, and describe how they perform concept evaluation and generalization of complex structural descriptions, problems typical of those we are concerned with.

Key Terms: Learning, automatic concept formation, generalization, Generalization-Based Memory, intelligent information systems, artificial intelligence, cognitive science.

1 Introduction

Automatic concept learning in the form of generalization has been shown to be useful in interpreting and organizing large amounts of information about a domain [Lebowitz 80; Schank 82; Lebowitz 83a], as well as being an interesting task in its own right. Recently, we have been concerned with the development of new

[1]This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165. Comments by Kathy McKeown and anonymous reviewers on an earlier draft of this paper were most helpful. Work on RESEARCHER and UNIMEM has been greatly advanced by graduate students at Columbia including Tom Ellman, Larry Hirsch, Laila Moussa, Cecile Paris, Kenneth Wasserman and Ursula Wolz.

methods of concept formation that employ a permanent memory of previously determined concepts along with the examples that led to their creation. These methods involve the determination of what concepts to learn, as well as the definitions of the concepts. In particular, we have concentrated on the problems of concept formation from a stream of input that is complex in any of several different ways. In this paper, we detail the class of problems we are addressing, present the basic learning technique that we use, known as Generalization-Based Memory (GBM), and indicate solutions to some of the specific problems that are involved.

Much of the concept learning research that has been done in Artificial Intelligence has consisted of either supplying programs with examples, and possibly counter-examples, of specified concepts and having these programs determine definitions of those concepts ([Winston 72; Mitchell 82; Dietterich and Michalski 83], for example) or of using largely analytic techniques to classify input (e.g., [Michalski 80; Langley 81]). In "real-world" settings, the crucial concepts to be learned -- those that best help explain and organize information about a domain -- are not pre-supplied; rather, it is necessary to determine these concepts from a stream of very complex input data. Consequently, our research concentrates not just on how to compare examples, but also on methods for determining what examples to compare, which largely determines the concepts to create.

Taking examples from various programs we have worked on, we look here at how intelligent systems could extract from complex input streams generalizations such as: "States that have high school expenditures have high per capita incomes" (from information about the states of the United States); "A large class of disk drives use flexible (floppy) discs" (from patent abstracts about disk drives); or "Terrorist attacks in Northern Ireland are frequently carried out by the IRA" (from news stories about terrorism), to the same extent as human learners.

We describe here a powerful memory organization and concept learning technique, Generalization-Based Memory GBM was developed for IPP, a computer program that read, remembered and generalized from news stories [Lebowitz 80; Lebowitz 83a; Lebowitz 83b], based on intuitions about how complex human

episodes might be stored in memory in a manner analogous to Schank's MOPs [Schank 80; Riesbeck 81; Schank 82] and Kolodner's E-MOPs [Kolodner 84]. We believe it is advantageous to use the same techniques in more traditional concept learning environments and for intelligent information systems that make use of complex streams of input. Our presentation of the problems of concept learning from complex input focuses on two intelligent information systems being developed at Columbia, UNIMEM and RESEARCHER, both of which use GBM.

UNIMEM is a program that can accept a large quantity of relatively unstructured facts about a domain, use generalization techniques to determine important concepts, and use these concepts to organize the information in a fashion that allows further generalization and intelligent question answering. For example, if information about the states in the U.S. is given to such a program (a domain used in prototype testing), the program might determine that New England states, or states with large education budgets are useful concepts. UNIMEM is being used to study problems that can arise when the individual items used for learning are not highly structured, each consisting simply of a set of descriptive features.

The problems in forming concepts from complex input data involved in our research with UNIMEM include: the impact of domain-dependent knowledge on concept learning; categorizing numeric input information so that generalization is possible; concept evaluation and refinement from further examples; using concepts that very slightly contradict new input items (those like Winston's "near misses" [Winston 72], but not pre-identified as such); dealing with concepts that change over time; and question answering based on Generalization-Based Memory. In this paper, we present the basic techniques for using GBM and for evaluating concepts in the context of UNIMEM.

RESEARCHER [Lebowitz 83c; Lebowitz 83d], in contrast with UNIMEM, deals with highly structured, physical descriptions of devices. RESEARCHER reads patent abstracts in natural language form, and then remembers and generalizes information from these texts, automatically creating appropriate object classes. Complete understanding (and generalization) of patent abstracts requires many kinds of analysis. To date, we have concentrated on the complex physical descriptions of

the objects described (i.e., part x is on top of part y), as opposed to, for example, functional characteristics. In this paper, we use RESEARCHER as a context in which to discuss the problems of comparing complex, highly structured representations.

Figure 1 shows some typical concepts generalized by each of the Generalization-Based Memory programs mentioned here. The IPP and UNIMEM generalizations were actually made by the programs (although the English was generated by hand), and the RESEARCHER examples are target concepts which can currently be learned from simplified input.

IPP Concepts:
Bombings in El Salvador cause damage, but do not often hurt anyone.
Urban terrorists in Italy frequently use silencer equipped pistols.

UNIMEM Concepts:
State class -- High urban percentage, low minority percentage, moderate income, low taxes, manufacturing important [RI, NJ, TX, MI, FLA, OH]

State class -- High value of farmland, fairly high population, manufacturing, agriculture, tourism important [NC, ARK, TENN, MINN, WISC, VA, MO]

RESEARCHER Concepts:
Floppy disk drive

Double density disk drive

Fully enclosed disk drive

Figure 1: GBM Concept Examples

In the remainder of this paper, we describe how our research relates to other work in concept formation, and present an overview of our concept learning methods, concentrating on our use of Generalization-Based Memory. Finally, we describe the way we handle concept evaluation and generalization of complex structural descriptions, problems typical of those we are concerned with.

2 Complex Input Domains

The intelligent information systems we are developing basically engage in what is called ***multiple concept learning from observation (descriptive generalization)*** in [Michalski 83]. These programs are given large number of examples, with no pre-specification of the concepts to generalize, and they acquire sets of concepts by deciding what instances to compare and how such examples are similar. The concepts derived are often overlapping, in that many concepts can describe the same example.

The tasks of our programs also involve aspects of Michalski's ***concept acquisition***. In addition to determining the properties of instances in the classes that they create, they fit objects to those classes. There are elements in our programs of both observing patterns in data and developing discriminant descriptions of the classes thereby derived.

Our research is characterized by several other properties, all somewhat novel for working systems (particularly in combination), but, we feel, crucial to the development of useful, dynamic, information systems. The first parameter that characterizes all our work is that we are dealing with ***"pragmatic" generalizations***. That is, we are concerned with concepts that describe what is ***usually***, but ***not necessarily always***, true. This means, crucially, that methods that invalidate generalized concepts on the basis of a single example are not acceptable In the same vein, we do not require that every concept that could legitimately be generalized be found. The class of pragmatic generalizations provides more power and flexibility in representing what it is possible to learn about a rich domain.

The pragmatic nature of our generalizations is in sharp contrast with most other learning methods. While there has been work dealing with noisy input data (e.g., [Quinlan 83], and to some extent [Mitchell 82]), it has always been assumed that the generalizations themselves perfectly described the world, although they were perhaps obscured in the input data. The need to deal with pragmatic generalizations strongly affects all aspects of our work.

Secondly, we look at learning that is ***incremental***. It is not possible in systems

that are continually receiving input to wait for all examples to be available for inspection before creating concepts. We require that after every example is processed, our systems have made the best possible generalizations based on the input that has been processed. While it is possible to imagine many other methods being applied incrementally, most other learning research has assumed that all the input is available at once to the learning process, and that the process is rerun from scratch if new information is added. A notable exception is [Winston 72], which incrementally develops a concept (although it only learns a single concept from specially selected inputs).

Finally, we expect that our systems will ultimately deal with *large numbers of examples*. It is the ability to deal with many examples and many concepts simultaneously that gives human learning the power we would like our systems to have. No method that requires comparison of a new instance with all, or a large portion of, previous examples will be acceptable, for computational reasons. Even comparison with generalized concepts must be done in a principled way. Furthermore, our systems must deal with whatever examples they are given, not specially prepared (as by a teacher) input. We are, in addition, sometimes concerned with cases where the individual items to be generalized are themselves complex, as in RESEARCHER.

While there has been learning research that involves large numbers of examples (e.g., [Quinlan 79]), much of it has been statistically oriented (see [Cohen and Feigenbaum 82]), and little of it has dealt with pragmatic generalizations (with the exception of [Schank 82], and related research). The fact that all concepts are not guaranteed to be logically correct turns out to have a major effect on the learning process.

We feel that methods for dealing with the type of input described here will be necessary in developing systems that take full advantage of the large quantities of complex information. One area that we have not addressed, but feel will be important in our future work, is the use of explanation-based generalization, of the sort discussed in [DeJong 83; Mitchell 83; Mostow 83; Riesbeck 83].

3 Generalization-Based Memory

In this section, we provide an overview of the techniques used to form concepts as part of maintaining a Generalization-Based Memory. For clarity, we describe the way the process works in UNIMEM, but the main techniques are identical in IPP and RESEARCHER.

The basic idea of Generalization-Based Memory is that a generalization system begins to create a hierarchy of concepts that describe a situation from a small number of examples, and then records in memory specific items, both those examples from which the concepts are generalized and others, in terms of the generalized concepts. More specific generalizations are recorded along with specific examples under the more general cases. GBM involves identifying and defining multiple concepts, as opposed to maintaining a single model of a concept.

In order to standardize our terminology, we refer to the objects stored in memory which are used to build generalizations, i.e., the input examples, as *instances*. In UNIMEM these are descriptions of objects in a domain. An instance is described in UNIMEM terms of a set of *features* (essentially property/value pairs). As we will see, RESEARCHER uses more complex descriptions of instances. The combinations of generalizations, themselves sets of features, and the events and sub-generalizations they organize are called *GEN-NODEs*.[2] GEN-NODEs form the basis of GBM. The structure of a typical GEN-NODE is shown in Figure 2. The manner in which GEN-NODEs are combined to form a concept hierarchy is illustrated in Figure 3.

Generalization-Based Memory consists basically of one or more hierarchies of GEN-NODEs that describe concepts of increasing specificity.[3] As shown in Figures 2 and 3, instances and sub-GEN-NODEs are stored under each GEN-NODE using

[2]GEN-NODEs were called S-MOPs in IPP, as they are, in some sense, specialized versions of Schank's Memory Organization Packets [Schank 82].

[3]Technically, through methods not described in this paper, the set of GEN-NODEs may not actually form a tree, but rather a directed acyclic graph.

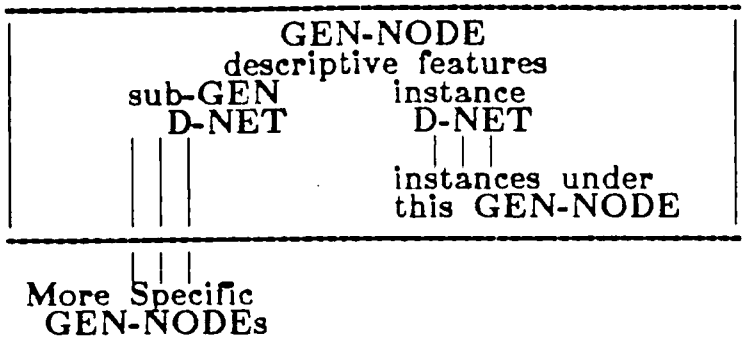

Figure 2: GEN-NODE Structure

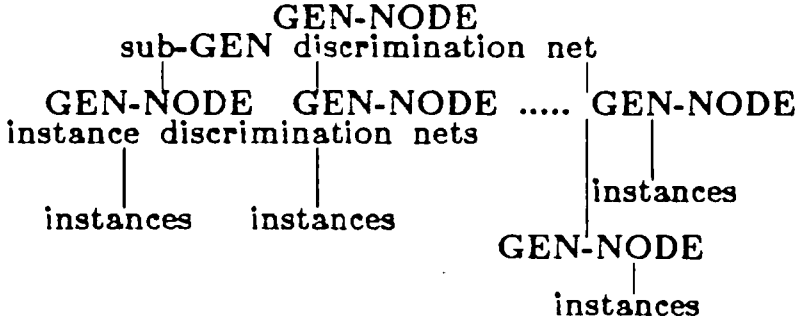

Figure 3: Schematic Structure of GBM

discrimination networks (D-NETs) [Charniak et al. 80]. (Note that a GEN-NODE can organize both instances and more specific GEN-NODEs.) D-NETs provide an efficient way to retrieve any object stored with a given set of indices. In the GBM model, every feature of an instance or sub-GEN-NODE is initially used as an index, resulting in shallow, bushy D-NETs that allow retrieval of an object given any one of its features. The resulting plethora of indices is pruned by ceasing to use as indices features that pertain to a large number of objects in a given D-NET.

The use of a hierarchy of GEN-NODEs with D-NETs as a method of memory organization allows efficient storage of information, since information in a generalization does not have to be repeated for each instance that it describes. In addition, it allows relevant generalizations and instances -- and *only* relevant generalizations and instances -- to be found efficiently in memory during processing, allowing further generalizations. This property of GBM is largely independent of the specific knowledge representation being used.

The use of concept hierarchies to intelligently and efficiently organize information about concepts is not a new one. Semantic networks [Quillian 78], frame systems [Minsky 75], MOPs [Schank 80; Schank 82], among many other formalisms all include this property. A primary feature of the representation language KRL [Bobrow and Winograd 77] is its ability to allow inheritance to be implemented easily. [Wasserman and Lebowitz 83] shows how frame-based schemes can be applied to physical object descriptions. What is new here is the dynamically changing nature of the concept hierarchy, and its use to guide the development of further concepts. Only a limited amount of work has been done on automatically generalizing concept hierarchies, including [Hayes 77; Michalski and Stepp 83; Sammut and Banerji 83], and this work has not dealt with pragmatic generalizations or particularly large numbers of examples.

The process of maintaining GBM, which is the learning process we are considering here, is a relatively simple one, once the memory organization method has been defined. As each new instance is processed, the most specific GEN-NODE that describes it is found. This is done, easily and efficiently, using the discrimination nets that index the GEN-NODEs in memory, starting with a very general node that covers the whole range of instances in the domain. Then, before the instance is actually indexed under that GEN-NODE, a check is made for instances already stored there that have additional features in common with the new instance, which can be found using the instance D-NET. If there are enough such features (one of many adjustable parameters of GBM[4]), a new concept is generalized, and the contributing instances indexed there. Otherwise, the new instance is simply stored under the existing GEN-NODE.[5]

Two further important features characterize GBM. Since concepts are generalized on the basis of few instances, they must be evaluated to eliminate over-generalization (including the elimination of whole concepts). This is discussed in

---

[4]Future research may look at how the parameters of GBM could be adjusted automatically.

[5]The process is actually a bit more complex, as a given instance can be stored in multiple spots in memory for two different reasons. An instance can either be classified initially in several different ways, each of which would indicate a place to store it, or several different "most specific" GEN-NODEs might be found, each of which would lead to the processing described.

Section 4. The second feature is the use of an idea known as *predictability*. While space does not permit a discussion of predictability here (see [Lebowitz 83a]), the basic idea is that only the presence of some features of a concept in an instance indicate the relevance of the concept, and that these features can be identified quite easily using GBM.

Further details of the algorithm used to maintain GBM are shown in Figures 4, 5 and 6. Figure 4 shows how the addition of a new instance to GBM consists of finding the GEN-NODE (or GEN-NODEs) that best describe the instance (updating feature confidence factors as this is done), followed by indexing the new instance (which includes a check for new generalizations). Figure 5 shows the process that searches for the GEN-NODE that best describes the new instance (essentially a depth first search heuristically guided by features of the new instance that have not been explained), and Figure 6 shows how the new instance is actually added to memory, possibly causing new concepts to be generalized.

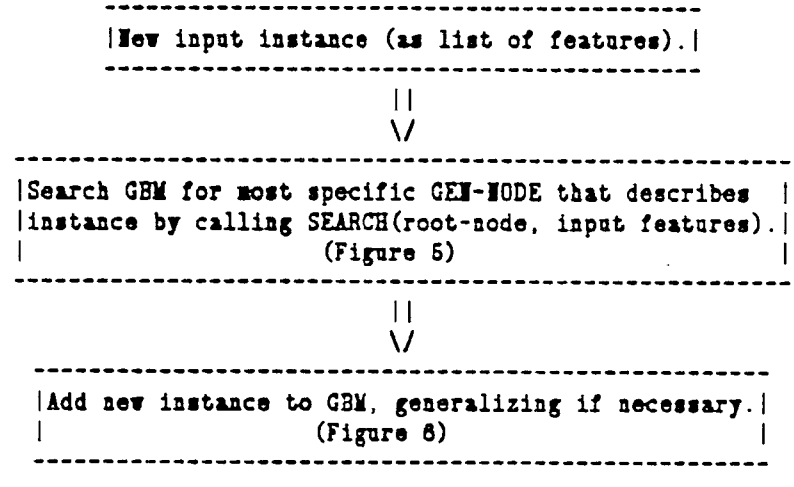

Figure 4: GBM Update Algorithm

We believe the use of GBM as described in this section can successfully satisfy the domain characteristics described in Section 2. In particular:

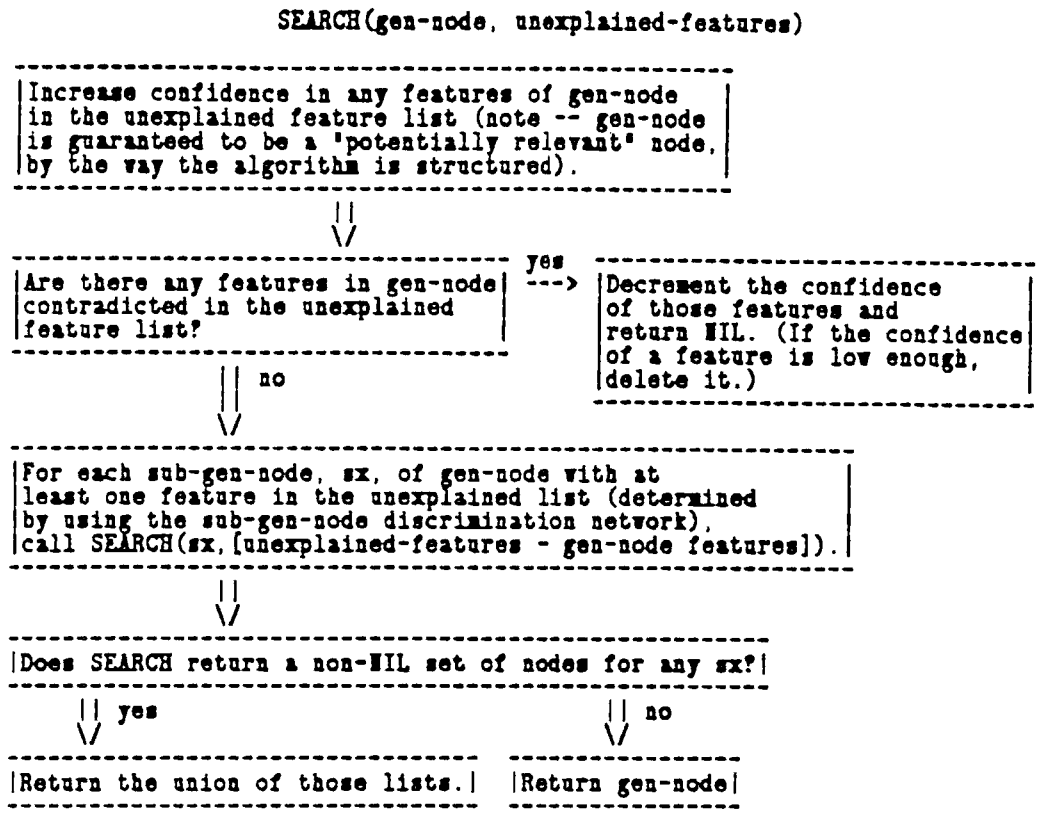

Figure 5: Searching GBM for Most Specific GEN-NODE

1) All concepts generalized in GBM are *"pragmatic"*. No concept is removed by a single counter-example, but instead, the process described in the next section is used to evaluate all concepts. The generalization process is also pragmatic because it can sometimes miss concepts that could be found by comparing instances that were stored in widely different parts of memory, but this seems a reasonable trade-off to avoid combinatoric numbers of comparisons.

2) GBM is inherently *incremental*. As each instance is added to GBM, the best possible concepts that can be generalized so far are made.

3) GBM is ideal for learning from *large numbers of examples*. The use of a hierarchy of concepts that organize specific instances allows only instances that might lead to generalizations to be compared to each other. Relevant concepts are easily found. It is also an efficient way to store the concepts.

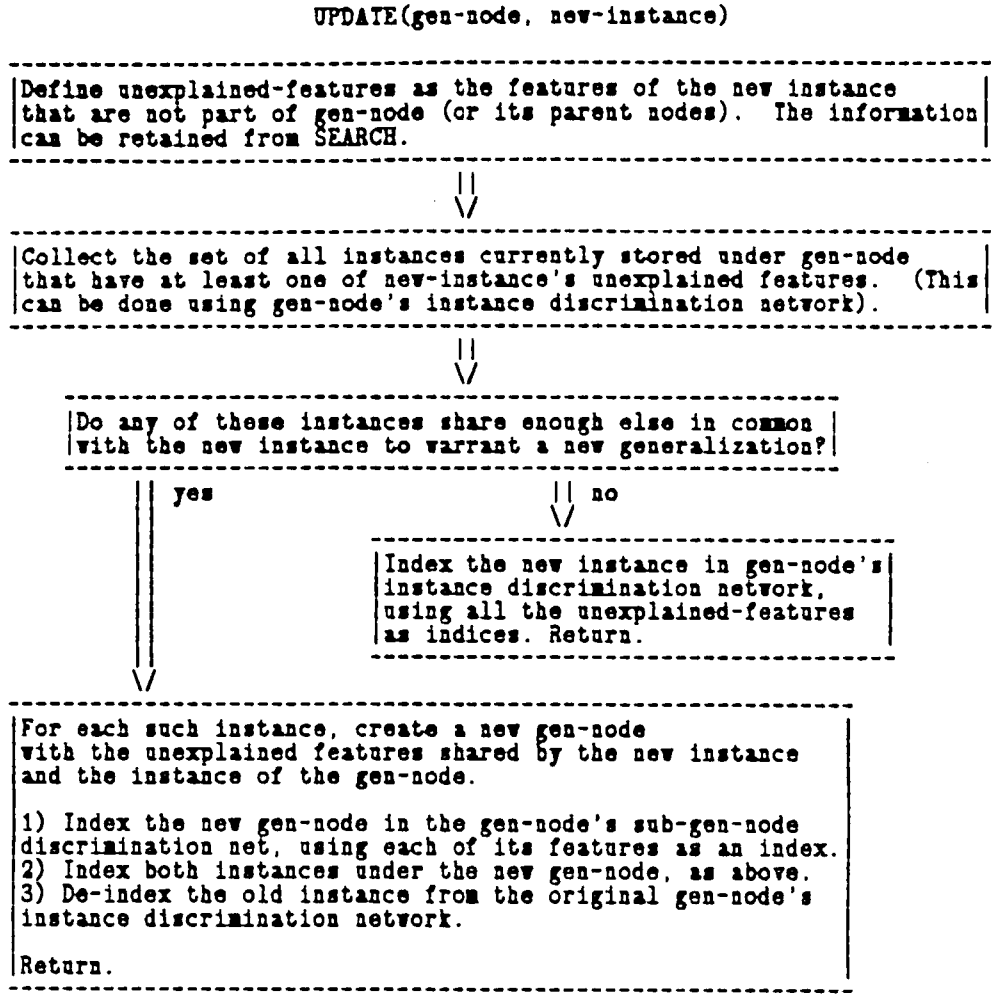

Figure 6: Updating GBM

We further illustrate the details of updating GBM with an example in Section 5 that follows a discussion of concept evaluation.

4 Concept Evaluation

As mentioned in the previous section, the concept learning process we have described inherently leads to over-generalization, particularly in a domain where there is a large amount of information about each instance. Thus, we require each concept learned to be evaluated over time. For each generalization made by UNIMEM, an evaluation process continually looks for later instances for which the generalization might be relevant. This occurs as a normal part of the memory

search process, since the generalizations to be evaluated are exactly those that might be used to store the new instances. UNIMEM checks whether a relevant generalization is confirmed or contradicted by each new instance.

A new instance found by UNIMEM is considered to contradict an applicable concept if it possesses a predictable feature indicating that the concept is relevant, but also another feature with the same property as the concept (such as the region of a state), but with a different value (Midwest instead of East, perhaps). When this condition occurs, intuitively, confidence in the concept should be reduced.

Early versions of confidence for generalizations in GBM simply involved adding or subtracting points from a numeric confidence level for each GEN-NODE, resulting in a property much like the confidence in conclusions discussed in [Collins 78], or the confidence in rule application used in some expert systems (e.g., MYCIN [Shortliffe 78]). In a domain rich in information this technique will not suffice, as there will almost *always* be extraneous information in each generalized concept, as the result of inevitable coincidences, that will cause confidence in the concept to be undermined.

What we would like to do when a generalization is disconfirmed is to throw away the "bad" (overly specific) parts and keep the "good" parts. The problem then reduces to identifying the components of a generalization that are overly specific, so that they can be deleted, leaving intact a valid generalization. Furthermore, for this to be useful, it must be done at a minimum of cost, hopefully occurring as a natural part of the memory update process, and requiring only a small amount of extra record-keeping. The task is somewhat similar to that for which pattern recognition techniques are used (see [Cohen and Feigenbaum 82] for an AI perspective to pattern recognition), but deals with concrete, if pragmatic, concept definitions, rather than statistical representations.

The solution devised for UNIMEM is straightforward. Instead of keeping a single confidence level as part of each GEN-NODE, UNIMEM tracks how often each feature of a concept is confirmed or contradicted. In effect, a confidence level is maintained for each feature of each concept, rather than a single value for an entire concept.

Specifically, a counter is maintained for each feature of each generalization and these counters are incremented or decremented as their features are confirmed or contradicted, respectively, in a situation where a concept is deemed relevant. The counter modification occurs as UNIMEM determines which GEN-NODEs best describe a new instance, as described in Section 3. If a counter passes a negative threshold (another adjustable parameter), then we can eliminate the feature from the generalization, since the feature has been wrong much more often than right. We sometimes have to eliminate entire generalizations when too many of their features have been eliminated. Details of this process, and an example of its application in the domain of football plays, can be found in [Lebowitz 82].

When this scheme was added to UNIMEM, it proved quite effective in culling extraneous features from generalizations, and only totally disconfirming those concepts that were completely the result of coincidence. In several test domains this procedure produced generalized concepts that made excellent intuitive sense. We show here a simple example from the domain involving information about states in the United States. Our use of this domain is fully explained in the detailed example in Section 5.[6]

Figure 7 illustrates a concept (GND1) generalized by UNIMEM. Roughly, this concept describes states with moderately high per capita income, rather low taxes, high school expenditure, and fairly low minority population (the last is actually a broad category that covers most states). This concept can be used to describe the seven states listed.

```
GND1:
INCOME          RANGE          INC3:4
TAXES           RANGE          TAX2:5
SCHOOL-EXP      RANGE          SCH3:3
MINORITY-PCT    RANGE          MIN1:2
Organizing: IOWA, KANSAS, MICHIGAN, MONTANA, NEBRASKA, PENNSYLVANIA, TEXAS
```

Figure 7: Final UNIMEM Generalization

[6] A different run of the program is used for the example here.

Figure 8 shows how this concept was initially generalized from Iowa and Nebraska). Notice that these states are similar in a number of additional ways, e.g., they are both farm states, so UNIMEM initially generalized an over-general, and not widely applicable, concept. These features, which are extraneous in the sense that they inhibit wider application of the concept, were ultimately removed by the evaluation process described in this section, leaving a much more useful concept.

```
GND1:
CRIME-RATE      RANGE          CRI3:5
STATE-DEBT      RANGE          DEB2:7
INCOME          RANGE          INC3:4
TAXES           RANGE          TAX2:5
MIGRATION-NET   RANGE          MIG1:9
SCHOOL-EXP      RANGE          SCH3:3
STATE           REGION         FARM
MINORITY        RANGE          MIN1:2
Organizing - IOWA NEBRASKA
```

Figure 8: Initial Generalization

5 A UNIMEM Example

As a further illustration of how GBM is maintained, including the formation of new concepts, we will present here an example taken from an actual run of UNIMEM in which we provided the program with a number of facts about each state in the United States. Figure 9 shows a small portion of GBM after information from 42 states (not including Oregon) had been added to memory. (The states were presented to UNIMEM in random order.[7])

Each GEN-NODE in Figure 9 is shown in terms of a set of features. For features derived from numeric data, the third column of each feature (the value) indicates a category derived from the numeric value by a method described in [Lebowitz 85]. For example, the fourth feature of GEN-NODE GND1, taxes, has the value TAX2:5, indicating that the tax rate for the states described by this

[7] Since UNIMEM has certain subjective aspects (in the sense of [Abelson 73; Carbonell 81], the concepts formed in GBM vary depending on the order instances are added. However, the effect does not seem to be strong, and the concept evaluation process described in the next section tends to lead to similar concepts arising over time, though not necessarily identical ones.

```
GND0
[ARIZONA MASSACHUSETTS NEWMEXICO SOUTHDAKOTA WESTVIRGINIA]

   GND1
   INDUSTRY        TYPE            MANUFACTURING (20)
   INDUSTRY        TYPE            TOURISM       (-2)
   INDUSTRY        TYPE            AGRICULTURE   (16)
   TAXES           RANGE           TAX2:5        (14)
   MINORITY        RANGE           MIN1:2        (32)
   STATE           SIZE            SIZ4:6        (deleted)
   STATE           REGION          MT            (deleted)
   INDUSTRY        TYPE            MINING        (deleted)
   INDUSTRY        TYPE            ELECTRONICS   (deleted)
   []

       GND5
       INCOME          RANGE           INC3:4      (4)
       INDUSTRY        TYPE            MINING      (1)
       SCHOOL-EXP      RANGE           SCH3:3      (0)
       STATE           SIZE            SIZ4:6      (deleted)
       URBAN-PCT       RANGE           URB6:6      (deleted)
       [UTAH]

           GND7
           CRIME-RATE      RANGE           CRI5:5      (-1)
           STATE-DEBT      RANGE           DEB3:7      (1)
           INDUSTRY        TYPE            GOVERNMENT  (-1)
           STATE           SIZE            SIZ4:6      (0)
           URBAN-PCT       RANGE           URB6:6      (0)
           [COLORADO NEVADA]

           GND13
           STATE-DEBT      RANGE           DEB5:7      (0)
           FARM-VAL        RANGE           FAR5:6      (0)
           STATE           SIZE            SIZ4:6      (0)
           URBAN-PCT       RANGE           URB6:6      (0)
           [MICHIGAN MINNESOTA]
```

Figure 9: A Section of UNIMEM GBM Without Oregon

GEN-NODE falls in the second of five categories, i.e., rather low. The numeric value following each feature indicates UNIMEM's current confidence in that feature (as described in the previous section). These values start at 0. The threshold for eliminating a feature was -3 for this run. The features followed by a "deleted" are not actually in the generalizations, but were originally included, and then deleted by the concept evaluation algorithm. Listed under each GEN-NODE are the instances (states) indexed there.

The section of GBM shown in Figure 9 includes five GEN-NODEs. The top-level node, GND0, has no features and hence describes all instances. It serves to organize the GBM hierarchy for states, and index any instances not yet described by any generalization. GND1 describes states with fairly low taxes, low minority population and industries including manufacturing, tourism and agriculture.

Additional feature present when it was created (from Idaho and Colorado, as it happens), have been deleted to make the GEN-NODE more widely applicable.

GND1 organizes several sub-GEN-NODEs, one of which, GND5, is shown in Figure 9. This node describes middle-income mining states with high school expenditures. Utah is indexed under GND5. This GEN-NODE organizes, in turn, two yet more specific GEN-NODEs, GND7 and GND13. GND7 describes mid-sized states with relatively high crime rates, moderate state debt, government as a significant industry and high proportion of urban population. Colorado and Nevada are indexed under it.[8] GND13 describes mid-sized states with high valued farm property, fairly high state debt and a high proportion of urban population. It indexes Michigan and Minnesota. Notice how for the states at the bottom of the hierarchy, such as Colorado, Nevada, Michigan and Minnesota, none of the information in GEN-NODEs GND1, GND5, and GND7 or GND13 will have to be repeated for the specific instance.

With GBM containing the information in Figure 9, we next added information about Oregon to memory. Figure 10 shows the first phase of this addition procedure. Shown are the features given to describe Oregon. Also shown are the results of the search phase, where UNIMEM determined that GND5 (as well as GEN-NODEs in other parts of GBM) best described the new instance. GND5 was selected because it contained at least one feature of Oregon (two, in fact, income and school expenditure), none of its features are contradicted by Oregon, and neither GND7 nor GND13 is appropriate (GND7 conflicts in state debt and urban percentage, and GND13 conflicts in farmland value and urban percentage).

Having decided that GND5 is the GEN-NODE that currently best describes Oregon, UNIMEM proceeds to update GBM, by attempting to index Oregon under that node. This results of process are shown in Figure 11. During the indexing process, UNIMEM notices that Utah, which is already indexed under GND5, has the identical values for state size, crime rate, and region of the country as does Oregon.

---

[8]Note that although these states probably have small numbers of total urban residents, the *proportion* of such residents is high.

```
*(run-state 'oregon)

Features: OREGON (STATE)
  STATE              REGION          WS
  POPULATION         RANGE           POP5:7
  URBAN-PCT          RANGE           URB5:6
  MINORITY           RANGE           MIN1:2
  MIGRATION-NET      RANGE           MIG8:9
  STATE              SIZE            SIZ4:6
  SCHOOL-EXP         RANGE           SCH3:3
  CRIME-RATE         RANGE           CRI4:5
  STATE-DEBT         RANGE           DEB5:7
  MILITARY-MONEY     RANGE           MIL4:9
  INCOME             RANGE           INC3:4
  FARM-VAL           RANGE           FAR4:6
  TAXES              RANGE           TAX2:5
  INDUSTRY           TYPE            MANUFACTURING
                     TYPE            FORESTRY
                     TYPE            TOURISM
                     TYPE            FOOD-PROCESSING
                     TYPE            AGRICULTURE

Best existing S-MOP(s) --
GND5 -- potential remindings: UTAH
<and others>
```

Figure 10: UNIMEM Finding a GEN-NODE that Describes Oregon

Thus, a new GEN-NODE, GND50, can be created with these features. (It also inherits all the features of GEN-NODEs GND1 and GND5).

```
Creating more specific STATE (GND50) than GND5 from events UTAH OREGON
 with features:

STATE          REGION          WS
STATE          SIZE            SIZ4:6
CRIME-RATE     RANGE           CRI4:5
------
  SCHOOL-EXP       RANGE           SCH3:3
  INCOME           RANGE           INC3:4
  INDUSTRY         TYPE            MINING
  MINORITY         RANGE           MIN1:2
  TAXES            RANGE           TAX2:5
  INDUSTRY         TYPE            MANUFACTURING
                   TYPE            TOURISM
                   TYPE            AGRICULTURE

<processing for other GEN-NODEs that describe Oregon>
```

Figure 11: UNIMEM Adding Oregon to GBM

Figure 12 shows how GBM has been changed by the addition of Oregon. GND50, the new GEN-NODE, has been added under GND5. Oregon and Utah have both been indexed there. Also note how the confidences of features supported by Oregon have be incremented, and those contradicted have been decremented, using

the algorithm described in the previous section. For example, in GND13, confidence in state debt and state size has increased, and confidence in farm value and urban percentage has gone down.

```
GND0
[ARIZONA MASSACHUSETTS NEWMEXICO SOUTHDAKOTA WESTVIRGINIA]

   GND1
   INDUSTRY        TYPE          MANUFACTURING (21)
   INDUSTRY        TYPE          TOURISM       (-1)
   INDUSTRY        TYPE          AGRICULTURE   (17)
   TAXES           RANGE         TAX2:5        (15)
   MINORITY        RANGE         MIN1:2        (33)
   STATE           SIZE          SIZ4:6        (deleted)
   STATE           REGION        MT            (deleted)
   INDUSTRY        TYPE          MINING        (deleted)
   INDUSTRY        TYPE          ELECTRONICS   (deleted)
   []

      GND5
      INCOME          RANGE          INC3:4      (5)
      INDUSTRY        TYPE           MINING      (0)
      SCHOOL-EXP      RANGE          SCH3:3      (1)
      STATE           SIZE           SIZ4:6      (deleted)
      URBAN-PCT       RANGE          URB6:6      (deleted)
      []

         GND7
         CRIME-RATE      RANGE          CRI5:5      (-2)
         STATE-DEBT      RANGE          DEB3:7      (0)
         INDUSTRY        TYPE           GOVERNMENT  (-2)
         STATE           SIZE           SIZ4:6      (1)
         URBAN-PCT       RANGE          URB6:6      (-1)
         [COLORADO NEVADA]

         GND13
         STATE-DEBT      RANGE          DEB5:7      (1)
         FARM-VAL        RANGE          FAR5:6      (-1)
         STATE           SIZE           SIZ4:6      (1)
         URBAN-PCT       RANGE          URB6:6      (-1)
         [MICHIGAN MINNESOTA]

         GND50
         STATE           REGION         WS          (0)
         STATE           SIZE           SIZ4:6      (0)
         CRIME-RATE      RANGE          CRI4:5      (0)
         [OREGON UTAH]
```

Figure 12: The Same Section of GBM With Oregon

6 RESEARCHER

As mentioned earlier in this paper, RESEARCHER [Lebowitz 83c; Lebowitz 83d], is a program that reads patent abstracts and adds information from them to a Generalization-Based Memory so that it can effectively answer questions. In this paper, we look only at the process of taking representations of two objects (or,

equivalently, a generalized concept and a concrete object) and forming a generalized concept. The representations we compare are frame-like and primitive-based, concentrating on the physical relations among the various parts of a complex object. (See [Wasserman and Lebowitz 83] for a complete description of the representation scheme.)

In the disc drive domain, typical concepts the generalization process might identify as being useful would be floppy disc drives or double sided discs. As with all our work, this must be done without specifically providing with examples of these concepts. Instead, instances stored together in Generalization-Based Memory are recognized as being similar and generalized.

The use of GBM is more complex here than in the UNIMEM. The "features" that two objects have in common can only be determined by comparing two complex object representations. The matching problem is much the same as that faced by Winston in his blocks world learning work [Winston 72]. Our problem is in certain ways both more difficult and easier than Winston's. It is more difficult because we are dealing with much more complex representations. It is simplified, however, at least in the long run, by the existence of an entire GBM, rather than a model of a single concept. We believe this will simplify the matching process. We look here both at the complexity of matching object descriptions and at how GBM can simplify the process.

The representations for two similar, slightly simplified, disc drive patents, used to test the initial version of RESEARCHER's generalization module are shown in Figure 13.

Clearly the two disc drives in Figure 13 have much in common that can be the source of a new concept derived through generalization -- an "enclosed disc drive". Figure 14 shows the concept created by RESEARCHER's generalization module. The process that created this generalization, while conceptually similar to the GBM update algorithm shown in section 4, differs in many details, largely due to the impossibility of representing complex physical objects as simple sets of features.

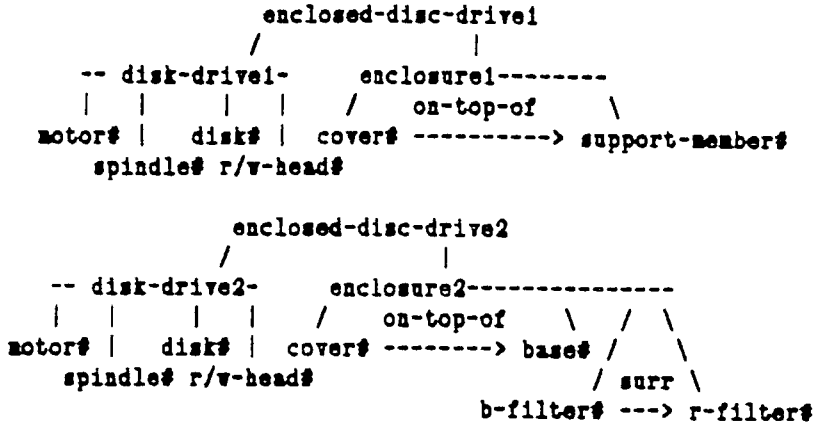

Figure 13: Similar Disc Drives

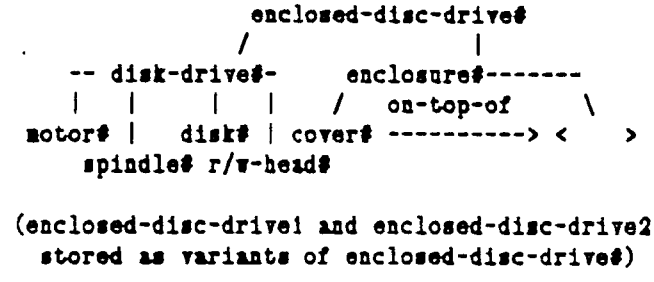

Figure 14: Generalized Enclosed Disc Drive

The idea illustrated in Figure 14 is that RESEARCHER finds the parts of two objects that are similar, and abstracts them out into a generalized concept. In this example, the two devices contained similar disc drives and enclosures. Each had a cover on top of some other object. These similarities form the basis of a generalized enclosed disc drive. Only the additional parts and relations of each instance need be recorded in memory along with the generalization. Currently, the generalization module of RESEARCHER, which is integrated in a simple fashion with the parser, is able to handle a moderate number of simple examples, including indexing the new objects as variants of existing generalizations.

Adapting GBM for use on complex structural descriptions has proven to be a difficult problem, even when only considering the assorted relations among the objects in the descriptions. Here we present one of the major problems and suggest the nature of the possible solution.

A central problem in generalizing structural descriptions is the process of matching two representations (either of two objects or an object and a generalized object), thereby determining what parts and relations correspond (as was pointed out for simpler examples in [Winston 72]). Clearly, if we have two distinct disk drive representations and wish to determine that the disk mounts in them are similar, then we must determine that they should be compared with each other. (Note that if the similarity is strong enough, we may wish to modify the representations to point to a single disk mount representation in memory.) Since one part of the description of complex objects is a set of relations, we must associate the relations in one object with those in the other.

The matching process here is a quite difficult one. Since we are dealing with structured objects, the parts of very similar objects may be aggregated differently in various descriptions. For example, a read/write head might be described as a direct part of a disc drive in one patent, but part of a "read/write assembly" in another. This makes the inherent similarity difficult to identify.

At the moment, we deal with this "level problem" with simple heuristics that allow only a limited amount of "level hopping" during the comparison process (to avoid the need to consider every possible correspondence among levels), and a bit of combinatoric force.

We feel that the ultimate solution to the level problem lies in more extensive use of Generalization-Based Memory. If a new object can be identified as an instance of a generalized concept, with only a few minor differences (done with a discrimination-net-based search of the sort described in Section 3), then the levels of aggregation will be set. By using Generalization-Based Memory, we need compare only a small number of *differences* between objects, rather than entire complex descriptions. This should allow RESEARCHER to meet all our performance

constraints (i.e., generalize pragmatically, be incremental, and handle large numbers of objects), even using the complex representations needed to describe real world objects.

In effect, what we are doing here is using the generalized descriptions that we have created to dynamically form a canonical framework for describing new objects. Such an approach, we believe, can help solve one of the major problems with canonical representations systems. Such representation schemes have many well-known advantages (see [Schank 72], for example), including simplifying the inference process. However, it is often difficult to select the canonical primitives needed for such schemes, and in domains that change over time, perhaps impossible. A dynamically created framework of the sort we are suggesting has the potential to gain the advantages of systems based on canonical primitives with the ability to adapt to the domain and without the problems of initially selecting the primitives.[9] A similar approach for cognitive modeling type tasks is taken in [Schank 82], and the issues of a dynamically changing canonical framework are a topic of our current research.

7 Conclusion

We believe that our work with Generalization-Based Memory has several important morals. The first is that the development of a dynamic set of concepts is a powerful approach to take when learning from a rich input domain. It is not realistic to hope to find the "right" set of concepts all at once, so it is crucial that we constantly update the concepts that we have and look for new ones. This allows us to take advantage of new information that is being provided and hopefully adapt to changes in the domain. Furthermore, the use of long-term memory, in the form of GBM, allows us to deal with many concepts at once, and still retain efficiency. In fact, as we have shown, considering many concepts at once often ends up being easier than learning one at a time and certainly leads to more powerful systems. We feel that our development of UNIMEM and RESEARCHER indicate that the idea of Generalization-Based Memory is a sound

[9] While we still have to develop an initial representation for the instances given to our system, it is not as crucial as in other systems, since many properties of the representation can change over time.

one, and that these programs can serve as valuable testbeds for the pursuit of important issues in concept learning.

References

[**Abelson 73**] Abelson, R. P. The structure of belief systems. In R. C. Schank and K. Colby, Ed., *Computer Models of Thought and Language*, W. H. Freeman Co., San Francisco, 1973.

[**Bobrow and Winograd 77**] Bobrow, D. G. and Winograd, T. "An overview of KRL, a knowledge representation language." *Cognitive Science 1*, 1, 1977, pp. 3 - 46.

[**Carbonell 81**] Carbonell, J. G. *Subjective Understanding: Computer Models of Belief Systems.* UMI Research Press, Ann Arbor, Michigan, 1981.

[**Charniak et al. 80**] Charniak E., Riesbeck, C. K., and McDermott, D. V. *Artificial Intelligence Programming.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1980.

[**Cohen and Feigenbaum 82**] Cohen, P. R. and Feigenbaum, E. A., eds. *The Handbook of Artificial Intelligence, Volume 3.* William Kaufmann, Inc. Los Altos, California, 1982.

[**Collins 78**] Collins, A. Fragments of a theory of human plausible reasoning. TINLAP-2, Urbana-Champagne, Illinois, 1978.

[**DeJong 83**] DeJong, G. F. An approach to learning from observation. Proceedings of the International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 171 - 176.

[**Dietterich and Michalski 83**] Dietterich, T. G. and Michalski, R. S. Discovering patterns in sequences of objects. Proceedings of the International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 41 - 57.

[**Hayes 77**] Hayes, P. J. On semantic nets, frames and associations. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, Cambridge, MA, 1977.

[**Kolodner 84**] Kolodner, J. L. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.

[**Langley 81**] Langley, P. "Data-driven discovery of natural laws." *Cognitive Science 5*, 1, 1981, pp. 31 - 54.

[**Lebowitz 80**] Lebowitz, M. Generalization and memory in an integrated understanding system. Technical Report 186, Yale University Department of Computer Science, 1980. PhD Thesis.

[**Lebowitz 82**] Lebowitz, M. "Correcting erroneous generalizations." *Cognition and Brain Theory 5*, 4, 1982, pp. 367 - 381.

[**Lebowitz 83a**] Lebowitz, M. "Generalization from natural language text." *Cognitive Science 7*, 1, 1983, pp. 1 - 40.

[**Lebowitz 83b**] Lebowitz, M. "Memory-based parsing." *Artificial Intelligence 21*, 4, 1983, pp. 363 - 404.

[**Lebowitz 83c**] Lebowitz, M. Intelligent information systems. Proceedings of the Sixth International ACM SIGIR Conference, ACM SIGIR, Washington, DC, 1983.

[**Lebowitz 83d**] Lebowitz, M. RESEARCHER: An overview. Proceedings of the Third National Conference on Artificial Intelligence, Washington, DC, 1983.

[**Lebowitz 85**] Lebowitz, M. "Classifying numeric information for generalization." *Cognitive Science* , 1985. in press

[**Michalski 80**] Michalski, R. S. "Pattern recognition as rule-guided inductive inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence 2*, 4, 1980, pp. 349 - 361.

[**Michalski 83**] Michalski, R. S. "A theory of methodology of inductive learning." *Artificial Intelligence 20*, 1983, pp. 111 - 161.

[**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, 1983, pp. 396 - 409.

[**Minsky 75**] Minsky, M. A framework for representing knowledge. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

[**Mitchell 82**] Mitchell, T. M. "Generalization as search." *Artificial Intelligence 18*, 1982, pp 203 - 226.

[**Mitchell 83**] Mitchell, T. M. Learning and problem solving. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983.

[Mostow 83] Mostow, J. Operationalizing advice: A problem-solving model. Proceedings of the International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 110 - 116.

[Quillian 78] Quillian, M. R. Semantic memory. In M. Minsky, Ed., *Semantic Information Processing*, MIT Press, Cambridge, MA, 1978.

[Quinlan 79] Quinlan, J. R. Induction over large data bases. Technical Report HPP-79-14, Stanford University Computer Science Department, 1979.

[Quinlan 83] Quinlan, J. R. Learning from noisy data. Proceedings of the International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 58 - 64.

[Riesbeck 81] Riesbeck, C. K. Failure-driven reminding for incremental learning. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981.

[Riesbeck 83] Riesbeck, C. K. Knowledge reorganization and reasoning style. Technical Report 270, Yale University Department of Computer Science, 1983.

[Sammut and Banerji 83] Sammut, C. and Banerji, R. Hierarchical memories: An aid to concept learning. Proceedings of the International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 74 - 80.

[Schank 72] Schank, R. C. "Conceptual Dependency: A theory of natural language understanding." *Cognitive Psychology 3*, 4, 1972, pp. 532 - 631.

[Schank 80] Schank, R. C. "Language and memory." *Cognitive Science 4*, 3, 1980, pp. 243 - 284.

[Schank 82] Schank, R. C. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, New York, 1982.

[Shortliffe 78] Shortliffe, E. H. *Computer-Based Medical Consultation: MYCIN.* Academic Press, New York, 1978.

[Wasserman and Lebowitz 83] Wasserman, K. and Lebowitz, M. "Representing complex physical objects." *Cognition and Brain Theory 6*, 3, 1983, pp. 333 - 352.

[Winston 72] Winston, P. H. Learning structural descriptions from examples. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1972.