



## **The Choice of Learning Strategy in an Autonomous System**

DIANA F. GORDON

*Navy Center for Applied Research  
in Artificial Intelligence  
Information Technology Division*

October 5, 1987

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE			Approved for public release; distribution unlimited.		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Report 9075			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory		6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Naval Research Laboratory		
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000			7b. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Office of Naval Research		8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217-5000			10. SOURCE OF FUNDING NUMBERS		10. SOURCE OF FUNDING NUMBERS
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.	
	61153N		RR015-09-42	DN156-045	
11. TITLE (Include Security Classification) The Choice of Learning Strategy in an Autonomous System					
12. PERSONAL AUTHOR(S) Gordon, Diana F.					
13a. TYPE OF REPORT Interim		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) October 5, 1987	15. PAGE COUNT 26
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Machine learning Robotics		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>Machine learning is a relatively new field within artificial intelligence. Most of the work to date has concentrated on individual learning strategies. Autonomous systems such as robots, however, require a variety of techniques.</p> <p>This report addresses the problem of strategy choice within a multistrategy system. Learning techniques are first categorized along five dimensions. Afterwards, a description is given of potential sources of constraints on the choice of strategy. Finally, existing learning systems that use multiple strategies are critiqued.</p>					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>		
22a. NAME OF RESPONSIBLE INDIVIDUAL Diana Gordon			22b. TELEPHONE (Include Area Code) (202) 767-2686	22c. OFFICE SYMBOL Code 5510	

## CONTENTS

1. INTRODUCTION .....	1
2. LEARNING STRATEGIES .....	1
2.1 Dimension 1: Nature of the Modification .....	1
2.1.1 New Knowledge .....	2
2.1.2 Refinement of Existing Knowledge .....	3
2.2 Dimension 2: Type of Inference Technique Involved .....	5
2.2.1 Monotonic Inference .....	5
2.2.2 Nonmonotonic Inference .....	5
2.3 Dimension 3: Explanatory Nature of the Inference .....	7
2.4 Dimension 4: Direction of Inference .....	7
2.5 Dimension 5: Autonomy of Learner .....	7
2.5.1 Discovery Systems .....	8
3. CONSTRAINTS ON STRATEGY CHOICE .....	8
3.1 External Sources of Constraints .....	8
3.1.1 Quality of Input .....	8
3.1.2 Form of Input .....	9
3.1.3 Time of Input .....	9
3.2 Internal Sources of Constraints .....	9
3.2.1 Background Knowledge .....	10
3.2.2 Focus of Attention .....	10
3.2.3 Goals .....	11
4. INTERACTION OF CONSTRAINTS .....	12
5. EXISTING MULTISTRATEGY SYSTEMS .....	13
5.1 LEX .....	13
5.2 INDUCE .....	14
5.3 BACON.5 .....	15
5.4 Systems Combining Empirical and Analytical Learning .....	15
5.5 EURISKO .....	16
5.6 SOAR .....	17
5.7 GRAPES .....	17
5.8 PI .....	18
6. CONCLUSIONS .....	19
7. ACKNOWLEDGMENT .....	20
8. REFERENCES .....	20

# THE CHOICE OF LEARNING STRATEGY IN AN AUTONOMOUS SYSTEM

## 1. INTRODUCTION

Although the blossoming of the field of machine learning is a relatively new development within artificial intelligence, it is apparent from the literature that there exists a wide variety in the type of learning strategies. Most researchers choose to concentrate in depth on a single strategy; e.g., how to inductively form concepts from data, how to modify rules based on feedback, how to extract problem-solving macros from experience, or how to create analogies then learn from them. Each method has its advantages. However, an agent that has no teacher to provide strict guidance, that has limited computational resources, and that is situated in a relatively complex environment is left with the problem of choosing appropriate learning strategies. Intelligent robots and autonomous vehicles that interact with their environment to achieve goals, for example, need to address the issue of strategy selection. Choosing a learning strategy involves heuristic techniques, i.e., there is no single, correct learning method for all situations. A system that can use constraints to guide the strategy selection process is better equipped to learn autonomously.

This report discusses the repertoire of known learning strategies and available sources of constraints to aid in the choice of strategies. Section 2 categorizes the strategies along five different dimensions. Example implementations of these strategies, taken from the literature, are given where appropriate. Section 3 lists and explains some important sources of constraints on strategy choice, with examples of some learning systems that use them. Since these constraints interact and their interaction affects the learning process, Section 4 discusses some of these interactions. Finally, existing machine learning systems that involve the use of multiple learning strategies are discussed in Section 5. Since work in the areas of multistrategy systems and systems that use numerous contextual and other constraints is in its infancy, this report does not present or propose complete solutions. Rather, the intention is to present an overview of the problem and related machine learning research.

## 2. LEARNING STRATEGIES

Learning strategies usually involve some type of inference. Based on this inference, these strategies may be categorized either by the search space that they delineate or else by their means of traversing this space. The dimensions along which we categorize learning strategies include the nature of the modification (which defines the search space), as well as the type of inference technique, the explanatory nature and direction of the inference, and the autonomy of the learner (which define the means of traversing the search space).

### 2.1 Dimension 1: Nature of the Modification

The inferential processes of a learning system perform some type of modification to the state of the system's knowledge. The nature of this modification defines the search space of possible resultant changes to the system. Either new internal knowledge structures may be created (or incorporated as is, as in the case of rote learning), or else existing ones may be refined. If new structures are created, they must either be declarative or procedural in form.

### 2.1.1 New Knowledge

#### Rote Learning

The method for incorporating knowledge *as is* (e.g. direct observations or by being told) is called *rote learning*. The rote learning strategy has received less attention than other learning strategies, perhaps because it is considered trivial. Nevertheless, it is not a simple strategy. Rote learning is learning without any inferences drawn from the input data [Michalski, Carbonell, and Mitchell, 1983], [Cohen and Feigenbaum, 1982]. Although with rote learning the data remains unaltered, some inference is often required to decide where to place the new information. Perhaps a great deal of inference may be involved in that large portions of the system need to be examined and adjusted in order to incorporate this new piece of knowledge (see description of reason maintenance systems later this report).

The earliest example of rote learning, which has since become a classic example, is in Samuel's checkers playing program [Samuel, 1963]. This program uses the minmax search technique for searching game trees [Nilsson, 1980]. This technique involves a scoring polynomial composed of heuristics for scoring board configurations. By using *lookahead*, and a method of *backing up* the values of the scoring polynomials from the lookahead level of the game, a player can select the *best* next move to make. Effectiveness of this technique is proportional to both the accuracy of the scoring polynomial and the depth of lookahead used. Therefore, Samuel's program has an added rote learning component. By having his system memorize both the board configuration and backed-up score after minmax has been done, this memorized information can later be retrieved and used, rather than having to be recomputed. This technique provides the effect of a deeper search without it actually having been performed. Since the number of board positions that can be saved and searched in a reasonable amount of space and time is limited, bookkeeping is done on the memorized items. The checkers playing program catalogues stored board positions for easy retrieval, deletes redundancies, and discards old records to save room. Samuel's method of ordering the records is based on frequency of use.

#### Creation of New Declarative Knowledge

Knowledge can be memorized but it can also be created. One of the most commonly used strategies for creating new declarative knowledge structures is concept formation. Concept formation is "the development of a new concept based on observations of instances." Michalski has done much work in this area. He approaches it both from a standpoint of learning a concept when given examples structured by a teacher, and also from the standpoint of classifying objects based on attributes when no instructor is present [Michalski, Carbonell, and Mitchell, 1983]. Michalski's STAR methodology for learning structural descriptions (concepts) from examples, used in programs INDUCE and AQ11, is a method for searching the space of possible concept descriptions. The algorithm consists of selecting a positive example and its description and creating the most general description that includes this example and does not include any of the negative examples. If there are positive examples that this description does not cover then the process is repeated with one of the uncovered examples. CLUSTER/2, Michalski's program which performs concept learning without teacher guidance, accepts a random set of objects along with their attributes. It then constructs a hierarchy of classes of the objects based on similarity of attributes.

The formation of relationship rules is another strategy for creating new, declarative knowledge structures. Relationship rules that may be formed include causal, temporal, class-property (If B is a bird, then B flies.), identity, part-whole, categorical (If B is a bird, then B is an animal.), physical (taller), and predictive. Michalski mentions the formation of class-property rules, which he calls

*descriptive generalizations*, but does not treat the subject in depth [Michalski, Carbonell, and Mitchell, 1983]. Thagard's program, PI, on the other hand, forms and subsequently uses and refines class-property rules [Holland, Holyoak, Nisbett, and Thagard, 1986]. In PI, a class-property rule, which Thagard calls an *instance-based generalization*, is formed by first searching for classes and properties that share a common instance. When these pairs are found, provided there are no counterexamples, a rule is formed. The strength of this rule is based on both the number and the variability of the instances.

Newly formed, declarative knowledge structures may also be in a very complex form. The formation of theories, explanations, and justifications might require a complex chain of reasoning involving both learning and problem-solving strategies. PI models scientific theory formation [Thagard and Holyoak, 1985]. This program is able to formulate, using a number of learning and problem-solving strategies, the wave theory of sound. Since this program is an example of multistrategy integration, it is described in depth later in this report.

### Creation of New Procedural Knowledge

In addition to declarative knowledge structures, new structures containing procedural information may be developed. The classic AI example is condition-action rules. These are prevalent in expert systems, and one of the most natural uses for machine learning is the creation of new rules to augment the knowledge bases of these expert systems.

Another type of procedural data structure that may be formed is a plan. Carbonell [Michalski, Carbonell, and Mitchell, 1983] creates new plans using analogy. His program solves problems, then uses a metaheuristic search to adapt the old solution to the new situation to formulate a plan suited to solve the new problem. Although plan formation from scratch is generally considered problem-solving, when analogy is used to form a new plan then the old and new plans may be combined to form a generalized plan, and this involves learning.

#### 2.1.2 Refinement of Existing Knowledge

New knowledge structures that enter a system must somehow be properly incorporated into the system, and both new and old structures must be adapted over time to suit new and changing circumstances. Therefore, learning generally involves some form of knowledge refinement. Two tactics for refining knowledge exist: modifying individual structures of a system, and modifying the system as a whole.

### Modification of Individual Structures

Generalization, or specific-to-general modification, and specialization (also called discrimination), or general-to-specific modification, are the most widely used strategies for modifying individual concepts and rules. Mitchell's *candidate elimination algorithm* [Michalski, Carbonell, and Mitchell, 1983], also described in the literature by a similar technique called *focusing* [Bundy, Silver, and Plummer, 1985], exemplifies the use of these two strategies, which often complement each other. This algorithm is used for learning concepts and is based on the *version space* representation. A version space of a concept is a specification of its most general and its most specific form known at any given time. For example, if the concept is "bird," then the most general form might be "animal with two legs" and the most specific form might be "canary." The general and specific boundaries are sufficient for delineation of a partially learned concept. Training examples narrow the range, thereby refining the concept description. Positive training examples raise the specific boundary whereas negative examples lower the general boundary. When the range converges to a single description (general and specific boundaries coincide), one may be assured that the concept has been

completely learned. The candidate elimination algorithm is used in a program by Mitchell et al. called LEX.

Another strategy for refining data structures is to modify classification hierarchies. A system may do this by tacking on exceptions or else by rearranging the hierarchy itself. The former is far more common. For example, suppose an agent believes that knives are a subclass of eating utensils. It then observes a knife that does not fit this category. It could either maintain the existing classification hierarchy and tack on the new knife as an exception, or it could modify the hierarchy itself to make knives no longer a subclass of eating utensils.

When an agent's beliefs and observations contradict, retraction of the original belief is a method of resolving the conflict. This strategy is rarely useful, though, because it is usually safer to modify or maintain the belief unless overwhelming evidence is found to the contrary.

Interesting and useful results may sometimes be obtained by combining pieces of existing data structures to form a single new one. This is the motivation for the *crossover* operation of genetic algorithms. Genetic algorithms are algorithms developed by Holland to simulate the biological natural selection process [Holland, Holyoak, Nisbett, and Thagard, 1986], [Goldberg, 1985]. They consist of operators that perform reproduction and recombination upon a population of individuals to facilitate adaptation to a changing environment. The elements of this population may be rules represented as bit strings or any number of data structures. In classifier systems, which use genetic algorithms, the individual rules of the population are the parts that get altered. One operation for alteration is the crossover operation. It consists of taking two individuals and choosing selected portions of both to form a new *offspring*. Thagard, in his program PI, adapts this operator to recombine concepts. He entitles his operator *conceptual combination*. Through the use of conflict-resolution rules, PI is able to combine two seemingly conflicting (based on their default properties) concepts, such as "feminist" and "bank teller," to form a new "feminist bankteller" concept possessing reasonable characteristics.

### Modification of Multiple Structures

In the process of learning, an agent may modify its knowledge base as a whole. Grefenstette and Pettey compare two approaches to genetic algorithms [Grefenstette and Pettey, 1986]. The first is the classifier system approach, where each structure of the population is a single production rule. The second is a rule sets approach, for which each structure consists of an entire set of rules that is evaluated for its fitness and modified as an entity (rules are altered rather than bits). A combination of the two approaches may prove to be advantageous.

Other approaches to modifying a system as a whole include self-organization and *truth maintenance*. To incorporate new knowledge, large portions of a system might need to be altered. For example, if rote learning is used to acquire new information, and the new information contradicts existing information, then internal consistency must be reestablished. This reestablishment of consistency is one of the purposes of a *truth maintenance system*. Doyle, the developer of the earliest truth maintenance system, states that beliefs usually have justifications [Doyle, 1981]. Each proposition in his system either has at least one currently valid reason or has no currently acceptable reasons (either has no reasons or no currently acceptable ones). The *reason* for each belief is an ordered pair consisting of the set of beliefs currently held to be valid and the set of beliefs currently not accepted. When a contradiction occurs, this truth maintenance system invokes a process called *dependency-directed backtracking* to find and remove at least one of the current assumptions to regain consistency. De Kleer has extended Doyle's truth maintenance process to an assumption-based rather than justification based one [De Kleer, 1986]. In his assumption-based truth maintenance system,

each belief is labeled with the sets of assumptions (representing the contexts) under which it holds. The assumptions are the primitive data from which all other data are derived. In De Kleer's system, there is no need for the overall database to be consistent, only a particular context.

## 2.2 Dimension 2: Type of Inference Technique Involved

Once the type of modification to be made to the knowledge base has been chosen, a system must select a means of achieving this modification. One way of categorizing such inference techniques is on the basis of whether or not they are monotonic.

### 2.2.1 Monotonic Inference

The term *monotonic* refers to a feature that holds of first-order logic, namely, if A and B are sets of first-order formulae and w is any formula, then if w is valid in the presence of information A, then w is still valid after B is discovered. Deductive inference preserves the monotonicity of a system. Some forms of generalization and specialization may be realized solely through the use of deductive inference [Michalski, Carbonell, and Mitchell, 1986, Chapter 9], [Mitchell, Keller, Kedar-Cabelli, 1986].

### 2.2.2 Nonmonotonic Inference

Although learning can be done using only deduction, most learning also involves *leaps of faith*. Induction, analogy, abduction, circumscription, and statistical inferences are some of the possible inference techniques for making *leaps of faith*. These techniques may introduce nonmonotonicity into the system in that the result is drawn from incomplete knowledge and therefore may have to later be revised as new evidence arises. Learning often consists of a process of forming and revising such conclusions.

#### Induction

Inductive inference is the most commonly used nonmonotonic machine learning technique. The dictionary defines induction as "the process of reasoning or drawing a conclusion from particular facts or individual cases" [Webster, 1983]. Nevertheless, machine learning literature uses the term to denote the heuristic process of forming and refining knowledge structures based on new information using incomplete knowledge. Although analogy, abduction, and circumscription may fit the dictionary definition of induction as well, they are usually considered to be separate processes. Induction is often the technique used for rule and concept formation, and also for generalization and specialization. Michalski [Michalski, Carbonell, and Mitchell, 1983] views inductive learning as a heuristic search through a space of symbolic descriptions, constrained by background knowledge. The goal state of this heuristic search, he states, is an inductive assertion that implies the observational statements, satisfies the problem background knowledge, and maximizes a given preference criterion. Operators for searching this space are generalization, specialization, and reformulation. Michalski distinguishes between *selective* and *constructive* rules for inductive search. If every descriptor in the generated description is among the descriptors of the initial description, then the rule is *selective*, otherwise it is *constructive*.

#### Analogy

Analogy, as applied to problem solving, is a two-step process. Given a *target* problem and situation, the steps of analogy consist of finding the most similar *source* problem solution, adapting it to fit the current target situation, and (often there is a third step of) generalizing those aspects that

both the source and target solutions share in common. The generalization step performed after an analogy has been drawn is a form of inductive inference. Winston has researched the use of analogous precedents for finding solutions to exercises [Winston, 1982], [Winston, 1984]. His program MAC-BETH takes a story and a precedent and forms causal connections between elements of the precedent and those of the new story. These causal connections, in the form of links, are then made into a new rule for future use by the system. Carbonell, as described above, views analogy as a source for plan formation. Darden applies analogical reasoning to scientific theory or hypothesis formation [Darden, 1983].

### Abduction

Another type of nonmonotonic inference used in learning is *abduction*. This process is useful in diagnosis. It consists of generating explanations from a given a set of symptoms. Abduction is a leap of faith in that the resultant explanation is plausible, yet is not necessarily true. Reggia sees the computational implementation of abductive inference as a *generalized set covering* or *parsimonious covering* process [Reggia, 1985]. His domain is medical diagnosis. According to this process, an explanation (consisting of a set of disorders) may be chosen if it can both account for all the manifestations present and be parsimonious. Thagard believes abduction plays "a major role in the justification, as well as the discovery of scientific theories" [Holland, Holyoak, Nisbett, Thagard, 1986]. When Thagard's program PI finds competing explanations for a set of evidence, it selects the best one based on both consiliency and simplicity. Consiliency "favors one hypothesis over another if what is explained by the latter is a proper subset of what is explained by the former." Simplicity favors the hypothesis that is based on the fewest assumptions [Thagard, in press].

### Circumscription

McCarthy [McCarthy, 1980], [McCarthy, 1986], [Etherington, 1984] has developed an interesting technique for drawing tentative conclusions. McCarthy's technique, called *circumscription*, is motivated by the problem of allowing an agent that has limited time for computations to draw conclusions based only on the evidence at hand, without having to consider the myriad of other possibilities. For example, if a robot's task is to put out a fire with an extinguisher with which it is equipped, the robot should not have to consider all potential obstacles to performing its task (such as a defective extinguisher or a brick that might fall on top of the extinguisher and break it) before acting. If new evidence is presented, such as a fire extinguisher that actually is defective, the circumscriptive conclusion might need revision. Although revising erroneous circumscriptive conclusions would involve learning, circumscription currently remains a domain studied by researchers of nonmonotonic logic rather than machine learning.

### Statistical Inference

In addition to the logical techniques for machine learning, there are also statistical techniques. A reason for using a statistical, or numerical, technique might be to allow formation of beliefs such as "Most birds fly." Numerical learning methods deal directly with the uncertainty inherent in the new knowledge structures formed by a learner, as well as the uncertainty of the existing knowledge. Two popular approaches to dealing with this uncertainty are probability and *fuzzy logic* [Zadeh, 1983]. One advantage of using probability for non-monotonic reasoning is that it can limit the amount of inference used. In Ginsberg (1985), an example is given. The motivation is similar to that of circumscription; namely, to accept a conclusion without consideration of all circumstances that could prohibit such a conclusion. For instance, suppose we are trying to prove proposition  $p$  and proving  $p$  involves proving that proposition  $q$  is unlikely. If proving  $q$  unlikely would not affect the eventual probability of  $p$  more than some insignificant value, then the system should not bother trying to prove

the unlikelihood of  $q$ . The advantage of fuzzy logic is that it allows derivation of propositions with quantifiers such as "most" and "usually."

### 2.3 Dimension 3: Explanatory Nature of the Inference

One aspect of learning programs that relates to the monotonic vs nonmonotonic distinction is the empirical vs analytical distinction. When the rules that are formed and used by a system are shallow (i.e., quantitative), thereby requiring large amounts of data, the system is considered *empirical*. When a system possesses and uses, and perhaps forms, explanatory structures, then it is considered analytical. If learning is empirical, then induction and other leaps of faith must be used to form a conclusion. However, in a purely analytical system, the knowledge of the system is complete enough to use deductive inference.

There has recently been an increasing interest in a powerful set of analytical techniques for deductively formulating a new generalization from a single example. The techniques are called *explanation-based generalization*. They are model-driven, knowledge-intensive methods of rule-learning that use the technique *propagation of constraints*. These techniques are motivated by Dijkstra's concept of *weakest precondition* [Dijkstra, 1976]; i.e., backward constraint propagation is performed upon a rule to derive the weakest possible sufficient rule precondition. Explanation-based generalization relies heavily on a strong domain theory to constrain the generalization process. From a single training example, these analytical methods can produce a valid generalization along with a deductive justification of the generalization. The EBG method [Mitchell, Keller, and Kedar-Cabelli, 1986] is a method that encompasses other explanation-based generalization techniques. It first constructs an explanation tree in terms of the domain theory that proves how the training example satisfies the goal concept definition (the concept to be learned) and whose leaves satisfy the operability criterion (which specifies the language of the target generalization). This method then regresses the goal concept through this explanation structure to determine a sufficient set of conditions under which the explanation structure holds, stated in terms that satisfy the operability criteria. Therefore, the EBG method is a two-step process: the first step creates explanations separating relevant from irrelevant example feature values, and the second step "analyzes this explanation to determine the particular constraints on these feature values that are sufficient for the explanation structure to apply in general" [Mitchell, Keller, and Kedar-Cabelli, 1986].

### 2.4 Dimension 4: Direction of Inference

Inference may either be data driven or model driven. In data-driven methods, learning is triggered by the data, whereas model-driven systems learn by formulating expectations and subsequently testing these assumptions. For example, suppose a system is designed to learn about the time of occurrence of high tide. A data-driven system would probably collect daily observational data correlating times and tide levels and then formulate a conclusion. A model-driven system might use the theory of gravitational attraction between the earth and the moon to predict high tide. It would then test these expectations. If expectations were to conflict with results, the model-driven system would refine its theory. A system that is both data and model driven is BACON.5, which is described in Section 5.

### 2.5 Dimension 5: Autonomy of Learner

Another aspect by which one may compare machine learning systems is that of the autonomy of the learner (also called *amount of inference*) [Michalski, Carbonell, and Mitchell, 1983] [Michalski, Carbonell, Mitchell, 1986]. The role of a teacher is to pre-digest and then present material to an agent in a manner conducive to learning. Without such an instructor, the agent must first transform

the input to a useful form before other learning processes can take over. Learning autonomy ranges from *being told* by a teacher exactly what the agent needs to know to observation and *discovery* at the other end of the scale. Since this report relates to autonomous systems, we only discuss learning by discovery.

### 2.5.1 Discovery Systems

Both Lenat and Langley have built discovery systems, and although the aim of both is to do scientific discovery, their approaches contrast sharply. Lenat's program AM discovers new domain concepts. His subsequent program, called EURISKO, not only discovers domain concepts but also new heuristics [Lenat, 1983], [Lenat, and Brown, 1984]. EURISKO has an internal agenda of topics to explore. Rules and concepts are selected for exploration or modification based on how *interesting* they are rated by Lenat's program. EURISKO's basic motivation is to analyze and mutate its own code in the pursuit of increased *worth*, a numerical measure of usefulness. Given an initial set of concepts and heuristics, EURISKO uses its heuristics to seek out and explore interesting patterns for the purpose of creating new concepts and heuristics, which are themselves open to future modification. Although AM and EURISKO's mutations are largely syntactic ones, these programs have achieved some surprisingly sophisticated results. For example, from initial set theory concepts and operators, AM has learned the concept of natural numbers, as well as advanced number theory concepts such as Goldbach's conjecture. AM's power is largely derived from the fact that the syntax mirrors the semantics.

Langley's program BACON, instead of performing creative self-mutations, discovers empirical laws by detecting regularities in numeric and nominal scientific data [Michalski, Carbonell, and Mitchell, 1983]. BACON operates within the domain of chemistry. Two laws discovered by BACON are Ohm's law and Archimedes' law of displacement.

## 3. CONSTRAINTS ON STRATEGY CHOICE

An agent capable of more than one of the above learning methods needs to select between them. Along each of the previously mentioned dimensions, a choice may be made. Factors that influence this choice of learning strategy can arise from two types of sources: external sources and internal sources. The external sources include anything originating in the agent's environment, whereas the internal sources include the learner's goals, background knowledge, and focus of attention.

### 3.1 External Sources of Constraints

Aspects of the environment of the agent that affect the choice of learning strategy include the quality, completeness and form of the input as well as whether or not the data is input incrementally.

#### 3.1.1 Quality of Input

The quality of the information received by the agent from its environment may range from very noisy to very reliable. For example, if the agent were a robot, input quality could depend upon the accuracy of the robot's sensors. Most machine learning systems to date assume noise-free input. Only a few researchers, such as Quinlan [Michalski, Mitchell, and Carbonell, 1986], have tackled the problem of noisy data. Noisiness of the data could affect the choice of analytical vs empirical techniques and also that of model- vs data-driven techniques. Since empirical methods are less costly than analytical ones, and since it could be a waste of computational effort to attempt many unsuccessful analytical inductions on noisy data, empirical methods seem preferable in data-driven learning situations. Once the noise has been filtered, analytical methods may be used [Lebowitz, 1986]. In

terms of model- vs data-driven techniques, model-driven methods seem to have a greater potential for noise immunity than do data-driven methods since they can use noise-screening heuristics. An example of a noise-screening heuristic is, "Ignore all incoming data whose value is a negative number." Analytical, model-driven methods are especially noise-resistant because their knowledge of causality reduces the likelihood of forming spurious conclusions. For instance, an agent knowledgeable about water and automobiles would be less likely to conclude, "The rain made the car start running," than an agent lacking the appropriate analytical theories. In addition to being unreliable, data might also be conflicting. Once again, model-driven, analytical techniques would be more helpful than data-driven ones for resolving any incoming inconsistencies. Also, if the information received is incomplete, nondeductive leaps of faith would be required since logical deduction would not be possible.

### *3.1.2 Form of Input*

The form of the input as well as the quality may cause a preference for some learning strategies over others. If an agent notices a set of objects, the presence of these objects would be likely to trigger a different type of learning than would be triggered if the agent notices a sequence of events. In the former case, concept formation or formation of class-predicate rules might be appropriate. In the latter case, a temporal or predictive rule would probably be suitable. One system that implements this constraint is Thagard's program PI [Holland, Holyoak, Nisbett, and Thagard, 1986]. PI looks for regularities in the data, and each regularity detector calls its appropriate, corresponding strategy.

### *3.1.3 Time of Input*

In addition to the form of the input, whether or not the data is input incrementally is also influential. Michalski's conceptual clustering techniques need all the data present at once. If the input from which to learn were presented over a period of time, then whenever new input arrived all the data would have to be reprocessed by rerunning the entire algorithm. Less brittle techniques capable of incremental learning would be more useful if the system's input were presented over a long period of time. For example, in the case of conceptual clustering, a more flexible system might be capable of reclassification.

## **Incremental Learning and Repetition**

When learning is incremental, a repetition of events may be noticed. Repetition is a useful trigger for pattern-directed learning. Formation of predictive and temporal rules such as "When it rains, anything on the ship's deck will get wet" is appropriate in this case, as is formation of class-property rules such as, "Birds have wings."

## **3.2 Internal Sources of Constraints**

Sources originating within the agent include its knowledge, goals, and attention. Background knowledge of the agent may be general or domain specific. Some of the early work in machine learning is predicated upon the notion that a machine can learn with little or no prior knowledge. Learning under such conditions, though, is a formidable task. As Winston claims [Winston, 1984], you need to know to learn. The amount of knowledge given to a learning system beforehand may range all the way from shallow to deep, general as well as domain-specific background knowledge. Memory of previous experiences, justifications, and expectations, and other aspects of the agent's model of the world all may play a role in the learning process.

### 3.2.1 Background Knowledge

#### Memories

Consider the following example of memories influencing the choice of learning strategy. Assume an agent sees 50 birds, and every one of them flies. Such an agent would likely conclude “All birds fly.” After seeing a single exception, specialization would be a reasonable strategy; and the belief could be revised to, “All birds but this one fly.” Alternatively, a more statistical conclusion such as, “Most birds fly,” might also be plausible. However, if as time progresses 20 more exceptions arise, a more reasonable strategy would be to retract the original conclusion or else modify it by postulating the existence of two classes of birds: those that fly and those that do not (reclassification strategy). In this example, the number of exceptions encountered has a marked influence upon the learning strategy choice.

#### Amount of Knowledge

The amount of domain knowledge available within the system will strongly affect the choice of deductive vs tentative inference methods. In the face of incomplete knowledge, either internal or external or both, a system can only rely upon speculative techniques such as empirical, data-driven learning methods. Analytical techniques may only be used when deeper, more explanatory knowledge is available. Lack of knowledge can also have a more obvious affect on the learning process. If an agent has no prior knowledge of a given concept, then it would be impossible for it to refine or augment knowledge about this concept without first learning about the concept itself. Since new knowledge builds upon previous knowledge, the amount and nature and organization of existing knowledge has direct relevance to the type of learning an agent will perform. Mitchell, for example, purposely restricts the vocabulary given to his system LEX because with a smaller vocabulary, LEX does not have to deal with a combinatorial explosion of potential results of induction. Utgoff deals with the ensuing problem of how one might augment the vocabulary of such a system as LEX [Michalski, Carbonell, Mitchell, 1986]. He calls the vocabulary-related constraints *biases* and considers the process of vocabulary extension a form of *shift of bias*. A system’s knowledge may also bias it in favor of a particular learning strategy; i.e., an agent that has an elaborate set of heuristics and other domain knowledge about physical relationships between objects might be more biased toward learning physical relationships between new objects it encounters than toward learning about other types of relationships. Thus what an agent knows (or does not know) constrains what it is able to learn.

### 3.2.2 Focus of Attention

Another factor influencing an agent’s learning is its focus of attention. If an agent has limited computational resources, then its attention must be restricted to certain aspects of its environment and internal knowledge. There are a number of ways in which focus of attention may constrain what is learned. First, an agent will only be aware of a limited set of external events. Therefore, in addition to being presented with only a restricted set of stimuli at any time, it may also only be capable of attending to a subset of these. Furthermore, an agent might also be restricted from accessing all of its knowledge or all consequences of its knowledge at any one time. An agent that knows all the logical consequences of its knowledge is considered *logically omniscient*. Logical omniscience is not very practical for computationally limited systems. Fagin and Halpern, in their report, “Belief, Awareness, and Limited Reasoning,” discuss the importance of adopting an approach to dealing with lack of omniscience [Fagin and Halpern, 1985]. Using human thought processes as an example of resource-bounded rumination, they claim people are not aware of everything; they do not always know the relevant rules; and they do not focus on all issues simultaneously. Drapkin, Miller, and Perlis have constructed a memory model for real-time default reasoning [Drapkin, Miller, and Perlis,

1985] to model this restriction. This model has short-term memory (STM), long-term memory (LTM), and intermediate-term memory (ITM). Their program models "an autonomous system that can act in an ever-changing world." STM represents the system's current focus of attention. It is small (currently it holds eight beliefs, but this can be changed). When STM exceeds its size capacity, the excess information flows into ITM. ITM keeps a record of all information that has passed through STM. Information may enter STM by observation, from LTM or ITM, or by the use of the logical rule of inference modus ponens. Observations such as "bird(Tweety)" or "yellow(Tweety)" are supplied to the system (to simulate visual input from the external environment). LTM is the location of the bulk of the system's knowledge. It is composed of association pairs, where the first element of the pair is the *trigger*. For example, "bird(X)" may be a trigger associated with the belief "bird(X) -> flies(X)." When an instantiation of "bird(X)," such as "bird(Tweety)" is in STM then "bird(X)" would trigger the other element of the pair, namely, "bird(X) -> flies(X)," to be brought into STM. On the next inference cycle modus ponens can then be used to infer "flies(Tweety)," which is now put into STM. If, after these additions to STM, it is too full, the excess goes into ITM.

PI [Holland, Holyoak, Nisbett, and Thagard, 1986], like the memory model, has a focus of attention mechanism. The implementation of attention in PI, though, is somewhat different than that of the memory model. In PI there is also a long-term memory, but the equivalent notion to short-term memory is a message list of *active* concepts, rules and problems. One way in which Thagard's message list differs from the short-term memory of Drapkin et al. is that rules on the message list have an associated strength (measure of past usefulness), whereas in the short-term memory of the memory model there are no numerical measures of usefulness; an item is either in or out of the short-term memory. PI uses the strengths of rules to select those rules that will be fired next. Once a focus of attention mechanism exists within a system, it can have great impact upon the type of learning that occurs. If the attention of the system were upon the passing of time, it would be likely to learn temporal relationships; if it were attending to colors, it would probably learn generalizations about colors of certain classes of objects; and if its attention were focused on building a structure from blocks, then it might learn generalizations about shapes and sizes of blocks that are stackable.

### 3.2.3 Goals

One of the ways in which attention may be focused is by the presence of goals. These goals might be problem-solving goals, general goals, or learning goals. SOAR, a project of Laird, Rosenbloom, and Newell [Laird, Rosenbloom, and Newell, 1986], uses goals as a means of focusing the learning that occurs within the system. The type of learning on which Laird et al. concentrate is a process called *chunking*. This process, to be described in greater detail below, is a form of knowledge compilation. SOAR chunks only information that relates to the subgoal-satisfaction procedure just completed. Because of this, SOAR learns only information relevant to the problem-solving task. Michalski, in a recent program CLUSTER/S, has added goals to the process of *conceptual clustering* [Michalski, Carbonell, and Mitchell, 1986]. This program uses a *goal-dependency network* to constrain the process of concept formation. The goals in the network are general goals such as "survive" and "be healthy and beautiful." These high-level goals are then decomposed into more specific ones, such as "drink water" and "eat lean meat." As an example, given a system goal of survive, which breaks down into subgoals of avoiding explosive and flammable materials, CLUSTER/S is able to form more useful classifications of trains carrying various substances than those it would have discovered without the aid of such a goal hierarchy. Such a hierarchy of goals could also impact the choice of learning strategy. If an agent devoid of relevant goals encounters 97 grey and 3 brown rocks in a bin, it would be reasonable for the agent to draw a statistical conclusion such as, "Most of the blocks in this bin are gray." If, instead, the first thing that the agent notices is that the brown rocks explode, and it has a survive goal, then it might be more reasonable in light of this goal to induce a class-property relationship about the explosive quality of brown rocks. It is advantageous for a system to have, in addition to general goals, knowledge of its own learning goals.

To give the LEX system (described above) this advantage, Keller has added *contextual meta-knowledge* to Mitchell's program, LEX [Keller, 1986]. This contextual framework makes LEX more analytical, more responsive to feedback on its behavior, gives it a sense of self-direction for deciding which concepts are worth learning, and provides it with adaptability. Keller did this by giving his system, METALEX, knowledge of its own problem-solving procedure, problem-solving goals, and performance criteria. A learner that knows its learning purpose, such as the improvement of problem-solving efficiency, also has an advantage in strategy selection. For example, a student studying for an examination under strong time limitations might be better able to use the time wisely if he or she were aware of both the time limitation and the learning goal of passing the exam. For instance, if the test expected memorization only, then an aware student would be more able to select a rote learning strategy rather than construction of theories from the text material (which would be both time consuming and also tangential to the immediate learning goal).

### **Success or Failure of Goals as Learning Triggers**

When environmental feedback fails to meet an agent's expectations, this is an indication that its world model needs refining. When the nature of the learning trigger is a conflict between beliefs and observations, the learning strategies appropriate to employ will not include chunking a successful problem-solving experience to use in the future. A more appropriate tactic to revise and refine the world view would be to analyze what went wrong and why it went wrong. On the other hand, if environmental feedback reinforces the original beliefs, and a sequence of actions was just performed based on these beliefs, chunking would be an ideal learning strategy to use.

### **Choice of Learning Task**

Another way in which goals may affect learning is in the choice of learning task. For example, a robot in a factory whose goal is to remove defective parts will probably choose to learn a definition of the class of parts that are frequently defective. On the other hand, if this factory robot has the goal of fixturing parts in a machine tool, then a more appropriate learning problem would be to define the class of fixturable parts.

## **4. INTERACTION OF CONSTRAINTS**

Apparently, an agent's environment, goals, knowledge, and attention can each serve as a meaningful guide toward contextually appropriate learning. When combined, these constraints form an even more complete picture of the current situation. However, when combined, the constraints may interact.

In a realistic situation, when an autonomous agent needs to select among learning strategies, constraints will not be involved in isolation. Instead, many will play a part in influencing the learning choice at any time. A system that is able to benefit from the full contextual situation will have an advantage in terms of the learning it will be able to perform. However, when multiple constraints are simultaneously present, there exists potential for interactions. For instance, an agent's goals may affect its focus of attention as well as its choice of learning task and learning strategy. Once a goal has focussed the agent's attention, though, the object(s) of the agent's attention might in turn alter the agent's knowledge and goal. If the agent now has a new goal, then this new goal might redirect the focus of attention which may then change the subject of the learning. Another influence on the agent's focus of attention and learning, besides its goals, is its knowledge.

The preceding set of constraint interactions may be demonstrated by an example. Assume the agent is a mobile pick-and-place robot on board a ship. This robot is capable of formulating and executing plans to achieve either of two goals: "Transport N object(s) from location X to location Y"

or "Put out fire" (the robot possesses a chemical spray that extinguishes fires). If the goals conflict, the latter will take precedence. As this robot executes plans to achieve its goals, it performs appropriate learning to help it improve its performance in future attempts to achieve the same goal. The robot is able to detect characteristics of objects such as their quantity, approximate weight, shape, size, and whether or not they are burning. Assume the robot is given (by some person or other agent) the goal, "Transport 10 objects from the front deck of the ship to the stern." This goal will cause the robot's attention to be focused on the objects on the front deck. Furthermore, since the goal includes object transportation, learning the class of objects that can be lifted and transported is appropriate. For instance, the robot might conclude, "All objects weighing less than 5 pounds can be lifted and transported." Suppose that one of the objects to be transported is burning. Observance of fire triggers the goal "Put out fire," which takes precedence over the "Transport  $N$  objects . . ." goal. While satisfying this new goal, it would be useful for the agent to form a generalization about the characteristics of flammable objects based on the current training example. The form of this generalization may depend upon the completeness of the agent's domain theory about flammability. For instance, if there is no domain theory then the robot might conclude "A flat object can burn." If, on the other hand, it has a strong domain theory, then the robot's attention might be focused upon the material of which the object is composed; and the robot might conclude that "An object made of cotton fabric can burn" (perhaps using explanation-based learning methods).

Apparently, an agent's goals, knowledge, focus of attention, and learning form a network of interactions as the agent acts and reacts within its environment. The preceding example demonstrates one possible set of interactions, but many other combinations of interactions are possible as well.

## 5. EXISTING MULTISTRATEGY SYSTEMS

We now discuss machine learning systems that use multiple learning strategies. These systems are reviewed in light of the variety of strategies they use, and also, in light of their use (or lack of use) of explicit constraints to limit the choice. Each of the systems discussed uses a unique combination of learning strategies:

- LEX combines generalization and specialization;
- INDUCE uses various types of generalization;
- BACON.5 is both model- and data-driven;
- Three systems combine empirical and analytical approaches;
- EURISKO learns multiple types of knowledge structures; and
- SOAR, GRAPES and PI possess a variety of learning strategies. SOAR and GRAPES use a single learning mechanism, but PI does not.

### 5.1 LEX

Mitchell's program LEX [Michalski, Carbonell, and Mitchell, 1983] uses the *candidate elimination algorithm* to form the preconditions of condition-action rules. It solves problems within the domain of symbolic integration, and its rules recommend the use of operators to solve integration

problems whenever a precondition of matching an integration pattern is met. An example *version space* for a LEX rule would be:

Specific:  $3x \cos(X)dx \rightarrow$  apply OP2  
with  $u = 3x$  and  $dv = \cos(x)dx$ ,

and

General:  $f1(x)f2(x)dx \rightarrow$  apply OP2  
with  $u = f1(x)$  and  $dv = f2(x)dx$ .

Any heuristic that is more general than the specific boundary and more specific than the general boundary is plausible. Thus a *version space* represents all possible versions of a heuristic consistent with the instances. Initially, the general boundary consists of any preconditions that are required to use the rule, and the specific boundary is defined by the first positive instance encountered. It follows that all rules have a beginning form, and this precludes the need for an initial search to represent the rule. In this way, LEX is able to avoid the complexity of the process of creating a new data structure. Thereafter, the version space is refined by positive and negative examples. There are only two allowable learning strategies in the system (since search through the space of hypotheses is breadth-first). The minimum generalization of the specific boundary necessary to incorporate a new positive instance, and the minimum specialization of the general boundary necessary to incorporate a new negative instance are the two strategies. Since the search through the space of hypotheses is breadth-first, the system need not select among degrees of generalization. Criteria for selecting between the two strategies is fixed. Negative examples trigger the single allowable method of specialization, and positive examples trigger the only allowable method of generalization. Advantages of a breadth-first approach are its monotonicity and its lack of need to store the instances for further reference.

## 5.2 INDUCE

Michalski's program INDUCE [Michalski, Carbonell, and Mitchell, 1983] performs induction differently than does LEX. Michalski views inductive learning as heuristic search through a space of symbolic descriptions. A teacher presents INDUCE with a set of observational statements, each describing an example, along with the information as to whether the example is a positive or negative example of the concept whose description is to be learned. These initial observational statements become the start state of the inductive search. The goal state is a description of the class. Strategies for searching the space of possible descriptions include generalization, specialization, and reformulation. Applied to the domain of conceptual data analysis, INDUCE is able to determine common and distinguishing properties of cancerous vs normal cells, given initial descriptors of the characteristics of example cells. Mitchell constrains the induction process in LEX both syntactically, by restricting the language used, and also by his using the *version space* method. Michalski's INDUCE uses an induction process that is less (implicitly) constrained by the design of the system. Michalski adds two types of explicit constraints to his system. The first is in the form of background information attached to *descriptors* of observational data. For example, if the descriptor were "color" or "length," then background information might include the domain or type of the descriptor allowable. The second type of constraint is in the form of preference criteria such as *simplicity* and *putational cost*. The combination of the use of a modified first-order predicate calculus for the descriptions, and the use of heuristic search to perform the induction allows a wider space of potential choices of inductive conclusion. INDUCE is therefore given both semantic background knowledge and syntactic preference criteria to trim this space. The manner in which this background knowledge will be accessed and used, and the form it will take, however, is fixed, and is not open to alteration by the program itself. Neither LEX nor INDUCE is intended to be used for an autonomous agent. The intention in both

cases is to present well-designed, nonproblem-specific methods of induction that can be applied to problem-solving tasks. Nevertheless, to be used for autonomous, contextually situated agents, it would be desirable for LEX to have a richer language and INDUCE to have its preference criteria generated by the system.

### 5.3 BACON.5

An interesting and useful combination of approaches may be found in BACON.5, by Langley et al. [Langley, Zytkow, Bradshaw, and Simon, 1983], [Langley, Bradshaw, and Simon, 1982]. Whereas earlier versions of BACON are strictly data driven, this version has both data driven heuristics for discovering relationships and expectation-driven heuristics for taking advantage of previous discoveries to make new ones. All versions of BACON discover regularities in scientific data within the domain of chemistry. The BACON.5 system contains "four data-driven heuristics for relating numeric terms, recursing to higher levels of description, postulating intrinsic properties such as mass and specific heat, and finding common divisors. BACON.5 also includes expectation-driven strategies for directing search based on discoveries that the program has already made. These include heuristics for expecting similar forms of laws, reducing the amount of data that must be gathered and taking advantage of the symmetrical form of some laws." [Langley, Bradshaw, and Simon, 1982]. Reasoning by analogy in a simple form is also included. Although the criteria for switching between data- and model-driven techniques is not explicitly stated, it is assumed that expectation-driven methods use existing knowledge whenever such knowledge is available. Otherwise, data-driven strategies must be adopted. BACON.5 is an empirical system. One way in which Langley et al. would like to improve their system is by adding both analytical techniques and the discovering new analytical rules to use in these techniques. This would move the BACON series into the realm of qualitative scientific discovery and would increase the noise immunity of the system.

### 5.4 Systems Combining Empirical and Analytical Learning

Explanation-based generalization is an effective deductive learning strategy that requires very complete background knowledge. Empirical (also called similarity-based) techniques, such as the candidate elimination algorithm, rely on little or shallow knowledge. Since the amount of an agent's background knowledge will probably vary under different circumstances, a robust learner should have a combination of explanation-based and empirical techniques. Contributing researchers in this direction include Mitchell, Lebowitz, and Kedar-Cabelli.

Mitchell discusses a method for combining empirical and analytical learning techniques for learning rules from examples [Mitchell, 1982]. His method uses the *version spaces* representation for rules described above. With the common language of *version spaces*, both empirical and analytical techniques continually refine the general and specific boundaries of a version space of a rule until it is fully learned. Explanation-based generalization is intended to precede the similarity-based learning, but criteria for deciding when to switch between the techniques is not discussed.

Lebowitz, on the other hand, describes a method for applying explanation-based learning to the results of similarity-based learning to create new causal rules [Lebowitz, 1986]. His system uses similarity-based learning (SBL) when applicable explanation-based learning (EBL) rules are missing or the EBL payoff is not likely to be too high. After SBL has created a number of generalizations of which the system is fairly confident, then EBL is applied to these SBL generalizations. Lebowitz uses the idea of *predictability* to focus an otherwise unmanageable (computationally explosive) explanation process. Predictability decides what elements are used in the explanation process. To form causal rules, it is necessary for his system to distinguish which elements of the SBL generalizations are causes and which are results. Predictability recommends calling those features of the generalization that are unique to that generalization *predictive*; they are most likely to be the causes in a causal

explanation. It is practical to assume that they cause the remaining nonpredictive features. Backward chaining methods find the causal chains that connect the predictive features to the nonpredictive ones. The second step, that of generalizing the resultant causal explanation chain, is only alluded to in the paper describing this work.

Kedar-Cabelli has a radically different approach to combining empirical and analytical techniques [Kedar-Cabelli, 1985]. Her system uses the two techniques in different phases of the process of analogy. It is also notable that for Kedar-Cabelli's system, context is important. The context, in the form of a purpose for the analogy, is explicitly supplied to the system beforehand. Most systems that use analogy, according to Kedar-Cabelli, analogize between two concepts. Her system instead analogizes between examples of concepts. Explanation-based generalization is used to extract those features of the base example that are relevant to the given purpose, or goal. The resultant explanation is then modified to fit the new example. Afterwards, the two explanations constructed for both the base and target examples are combined using similarity-based techniques to form a single generalization of the goal concept.

## 5.5 EURISKO

In each of the systems mentioned so far in this section, the form of knowledge structure created or refined is of one type. One system that is more varied in terms of the type of modification is EURISKO by Lenat (1982). EURISKO creates both declarative (concepts) and procedural (heuristics) knowledge structures, and both creates new structures and refine existing ones. Various aspects of concepts and heuristics are also open to modification. For example, the *worth* measure of both concepts and heuristics may be learned. Furthermore, multiple strategies such as generalization, specialization, and analogy are used. EURISKO is an offshoot of an earlier program by Lenat called AM. AM only learns concepts. The discovery processes of EURISKO are less limited than those of AM because EURISKO's control structures (heuristics), as well as its concepts, are open to formation and modification. Both systems have been applied to the domain of mathematics. Other domains to which EURISKO has been applied, such as games and circuit design, all share a certain trait in common, namely, that they are completely formalizable within the machine. For this reason, some of Lenat's conclusions do not apply for the scenario of an autonomous agent situated in a complex environment. The primary way in which EURISKO chooses between strategies is by metaheuristics. In complex situations, where information originates externally as well as internally, such heuristics would be difficult to imagine. For example, if the right-hand side of the heuristic were *Use Retraction*, what would the left-hand side look like? One of the most useful ideas implemented in EURISKO, though, is to use the same form (in this case frames) for both the declarative and procedural knowledge and to make all knowledge structures of the system open to modification by the system itself. EURISKO also possesses an internal focus of attention mechanism. For EURISKO, this is implemented as a *focus of attention* heuristic that "recommends pursuing a course of action simply because it's been worked on recently" [Lenat, 1982]. EURISKO's agenda, or blackboard, also serves to focus the attention of the program. Learning triggers consist of *interestingness* heuristics, which comb the knowledge for structures that meet a given set of criteria for interestingness. EURISKO uses learning triggers as sources of constraints. For example, feedback is used to determine which type of heuristics to form: "Heuristics that serve as plausible move generators originate by generalizing from past successes; heuristics that prune away implausible moves originate by generalizing from past failures" [Lenat, 1982].

EURISKO is a hybrid system not only from the point of view of the variety in knowledge structures formed and refined, but also from the point of view of the varying level of autonomy involved in the one system. In addition to discovery, EURISKO also does rote learning when it fills the *examples* category associated with a concept or heuristic.

## 5.6 SOAR

SOAR, a project of Laird, Rosenbloom, and Newell, is a program possessing both a general problem solver and a general learning technique. The goal of the authors of SOAR is to create a "general learning mechanism," which they define to be one "capable of learning all that needs to be learned." The mechanism that they have developed is called *chunking* [Laird, Rosenbloom, and Newell, 1986]. The *chunking* technique is an offshoot of an earlier learning technique which compiles knowledge into macros [Fikes, Hart, and Nilsson, 1972]. In Fikes et al.'s system, named STRIPS, a problem-solver is augmented with the capability of forming generalized plans. These generalized plans, whose problem-specific constants are replaced by problem-independent parameters, are stored in a data structure called a *triangle table* and used in subsequent problem-solving. In SOAR, the experiences of the problem-solver are codified in the chunks, where a chunk is defined as "a symbol that designates a pattern of other symbols" and is basically a problem-solving macro. The theory behind learning in SOAR is that as one practices a skill, chunks are formed that improve performance. In terms of the implementation, a chunk is a modified subgoal-satisfying procedure. Preconditions for the chunk are formed from those subgoal preconditions that existed before the subgoal was instantiated but that were necessary for subgoal satisfaction. Actions for the chunk are extracted from subgoal actions that resulted from subgoal satisfaction. Identifiers of the chunk are then variabilized. The major reason that these chunks are applicable in a general way is that only aspects of the situation that were referenced during problem-solving are stored in the chunks. Chunks differ from STRIPS triangle tables in that triangle tables are a single data structure containing the macro. In SOAR, for each problem-solving operator there is a chunk that causes it to be selected. Because of this, learning is incremental. In addition to learning operator-selection chunks, SOAR can also learn chunks that implement the operators. Since transfer between macro-operators exists (some operators are just symmetric equivalents of others, for instance), the number of macro-operators is greatly reduced. SOAR does generalization by use of this chunking technique. In addition to generalization, the authors of SOAR would like the program to be able to do discrimination so it can deal with overgeneralized chunks [Laird, Rosenbloom, and Newell, 1984]. Although Laird et al. claim that chunking is general enough to do any kind of learning, a number of other types of learning have yet to be proven implementable within this technique. Chunking acquires condition-action rules from goal-based experience. It does not form concepts or various types of relationship rules; neither does it consider modification of the system as a whole, as does truth maintenance, fair game.

## 5.7 GRAPES

Anderson has developed two multistrategy programs, ACT and GRAPES, both described in the paper, "Knowledge Compilation: The General Learning Mechanism" [Michalski, Carbonell, and Mitchell, 1986]. GRAPES uses a mechanism that is quite similar to chunking. ACT, the predecessor of GRAPES, incorporates multiple strategies such as analogy, generalization, specialization, and knowledge compilation, in a single program. These learning strategies are applied to problem-solving in the domain of creating geometry proofs from both declarative knowledge and practice examples. Anderson's newer program, GRAPES, uses his subsequent discovery that the knowledge compilation technique (which is analogous to chunking) is actually able to do generalization and specialization as well as the formation of proceduralized macros that it was originally designed to perform. Anderson's *knowledge compilation* consists of two subprocesses: composition, which merges multiple productions into a single production, and proceduralization, which builds new productions by collapsing the formerly separate processes of *information retrieval* and *production matching*. GRAPES constructs new LISP functions from the combination of a template and existing LISP function examples. The problem is hierarchically decomposed into a goal tree. When a new instance is encountered, the system compares it with the template and uses analogous examples to fill in as needed. Knowledge compilation is then used to create a new production rule based on a generalization of the experience. The form of the compilation depends upon the structure of the goal tree. Compiling the process of

making the analogy results in the most specific generalization of the two instances. In addition to generalization, Anderson uses knowledge compilation for discrimination. The discrimination algorithm consists of comparing a new example with previous instances and finding a discriminating characteristic. The concept is discriminated by this characteristic; and this whole process is compiled. Although compilation is able to implement both of these strategies, the compilation procedure succeeds the point of strategy choice, which is the subject of this report. Anderson considers strategy choosing the domain of the problem-solver rather than the learner. According to Anderson, "the induction process occurs as a conscious problem-solving effort to find a basis for dealing with a new case. In compiling the results of this problem solution, productions are formed that will extend to the new situation." [Michalski, Carbonell, and Mitchell, 1986]. Other machine learning researchers, such as Michalski and Carbonell, on the other hand, consider induction and analogy a part of the learning process. Here, learning is intended to include the broader, more inclusive definition, and chunking, or knowledge compilation, is considered to be a very general and powerful technique that may be used to implement a number of different learning strategies.

In both SOAR of Laird et al. and GRAPES by Anderson, goals, background knowledge, and feedback such as success in problem-solving, play a role in constraining that which is chunked. Since they will only chunk, or compile, what is directly related to the problem-solving experience, their searches are able to be both sufficiently trim and contextually appropriate.

## 5.8 PI

The final system that is examined herein is PI, written by Thagard. PI forms scientific theories by triggering various types of inference in the process of theory construction. The types of inference that are triggered include deduction, induction (generalization and specialization), abduction, analogy, and conceptual combination (which is a variant of Holland's crossover operator, both described above). PI has rediscovered the wave theory of sound [Thagard and Holyoak, 1985]. A problem-solver with a focus-of-attention constraint is at the heart of the system. Attention focus is implemented by the combination of a long-term memory and an *active list* containing rules, messages (observations and other propositional facts), concepts, and problems [Holland, Holyoak, Nisbett, and Thagard, 1986]. Problem-solving begins when the initial conditions and goals are put on the active list. The process then proceeds by *spreading activation*, which uses both forward and backward chaining. The chaining is accomplished by firing the best of those rules on the active list whose concepts are active and then putting the results of the actions back on the list. Rules compete on the basis of *strengths*, which may be altered (or learned) by the system. Rules that have contributed to a successful solution are rewarded by having their strength increased, and those that contributed to an unsuccessful projection have their strengths decreased as punishment.

At each iteration of PI's problem-solving loop, those conditions that trigger non-monotonic inferences, such as generalization or analogy, are tested. If active concepts have common instances, and if the resultant generalization would be agreeable to the agent's general goals, then PI will form a class-property rule. Generalization of a rule precondition is triggered if two active rules share a common action and a common precondition conjunct (syntactic trigger). Specialization is triggered if an active rule and active instance conflict, and formation of a new concept is triggered by syntactically similar preconditions of two active rules. Therefore, focus of attention (in the form of *active status*), goals (by being put on the message list), input (in the form of observations asynchronously put on the message list), background knowledge (as triggered rules), and feedback (success, failure) are all involved in constraining the choice of learning strategy. Thus PI holds much promise as a system not only useful for scientific discovery but also as a framework for modeling cognitive processing capabilities of an autonomous agent within a complex world. It is, however, only partially developed. Most potential enhancements to the program lie in two directions. More breadth, depth, and a deeper understanding of the strategies themselves would be helpful, as well as improved understanding of the constraints in terms of how they interact in the strategy selection process.

The first area for improvement, that of developing and refining strategies, has been the major thrust of research in the field of machine learning. It will no doubt continue to flourish. PI could be enhanced by the formation of other types of relationship rules such as temporal or predictive rules based on repetition of events. It might also improve by using a richer repertoire of tactics for dealing with contradictions. Much human learning occurs as a result of mistakes providing negative feedback. PI is able to respond to two types of contradictions: when a projection is unsuccessful, the rules involved are punished by having their strength reduced; and when an exception is encountered to a class-property rule such as, "All birds are blue," then specialization is called, which always modifies the rule to make it into a more specialized version, using an *unusual* property of the exception as the discriminating conjunct to be added to the rule. Alternative approaches to the latter situation might consider the degree of unusualness of this property (near miss vs far miss), or the number of exceptions encountered so far. Additionally, the system should have effective ways of screening noise.

The other direction in which PI may be enhanced is through the study of the constraints themselves. For example, both the general and problem-solving goals may form a complex, interacting network. Currently, these goals are not explicitly interrelated [Thagard, personal communication].

To date, PI has been tested in domains where it is given unrealistic slices of information from which to draw conclusions so that the program will not waste time searching unproductive paths. As Thagard states, "This simulation of course falls well short of capturing the complexity of how humans might solve the problem, because they operate with a wealth of possibly useful and possibly misleading information that the program does not have" [Holland, Holyoak, Nisbett, and Thagard, 1986]. Adding flexibility, in the sense of improved strategies and a deeper understanding of the role played by constraints on the strategy choice, might be two ways of increasing PI's ability to handle messier, more realistic domains.

Although none of the programs described in this section is a complete answer to the design of an autonomous agent situated in an environment, each has tackled a portion of the problem of combining multiple strategies within a single system; and each has had to implement at least some of the preceding constraints to limit the search space involved in the learning process.

## 6. CONCLUSIONS

Learning involves change in the system's state of knowledge. The nature of this change and the nature of the inferential processes involved in instituting such a change may differ. Distinctly different learning strategies may be appropriate as circumstances vary. Clues that help to define contextual appropriateness may arise from a number of different sources for an autonomous agent. Potential sources include the nature and timing of environmental input, the agent's goals and previous knowledge, the object(s) of its attention, and the degree of match between the agent's expectations and its experiences. Such clues may act as guidelines for choosing among the various learning strategies.

Most machine learning systems to date have implemented a limited number of learning strategies. Since the domains in which they have been tested lack the variety and complexity of the real world, the need for multiple strategies and multiple constraint sources has been limited. However, in order that autonomous agents, such as robots, may cope in less restricted environments, it will be desirable for them to possess the ability to capitalize upon different situations each in an appropriate way.

## 7. ACKNOWLEDGMENT

The author thanks John Grefenstette for patiently reading many earlier versions of this manuscript and offering valuable comments and suggestions.

## 8. REFERENCES

- Bundy, A., B. Silver and D. Plummer, "An Analytical Comparison of Some Rule-Learning Programs," *Artificial Intelligence* 27, (2), 137-181 (1985).
- Cohen, P. and E. Feigenbaum, (eds.), *Handbook of AI 3* (William Kaufmann, Inc., 1982), Vol. 3.
- Darden, L., "Reasoning by Analogy in Scientific Theory Construction," Proceedings of the International Machine Learning Workshop, Monticello, Illinois, 1983.
- De Kleer, J., "An Assumption-Based TMS," *Artificial Intelligence* 28 (2), 127-162 (1986).
- Dijkstra, E., *A Discipline of Programming* (Prentice-Hall, 1976).
- Doyle, J., "A Truth Maintenance System," in *Readings in Artificial Intelligence*, (B. Webber and N. Nilsson, eds. (Tioga Publishing Company, Palo Alto, Calif., 1981), pp. 496-516.
- Drapkin, J. M. Miller and D. Perlis, "A Memory Model for Real-Time Default Reasoning," University of Maryland Technical Report, 1986.
- Etherington, D., R. Mercer and R. Reiter, "On the Adequacy of Predicate Circumscription for Closed-World Reasoning," Proceedings of the Non-Monotonic Reasoning Workshop, AAAI, Oct. 1984, pp. 70-81.
- Fagin, R. and J. Halpern, "Belief, Awareness, and Limited Reasoning: Preliminary Report," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI85, pp. 491-501.
- Fikes, R., Hart, P., and N. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence* 3, 251-288 (1972).
- Ginsberg, M., "Does Probability have a Place in Non-Monotonic Reasoning?," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI85, pp. 107-110.
- Goldberg, D., "Dynamic System Control Using Rule Learning and Genetic Algorithms," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI85, 588-592.
- Grefenstette, J. and C. Pettey, "Approaches to Machine Learning with Genetic Algorithms," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1986, 170-174.
- Holland, J., K. Holyoak, R. Nisbett, and P. Thagard, *Induction: Processes of Inference, Learning and Discovery* (The MIT Press, Cambridge, MA, 1986).
- Kedar-Cabelli, S., "Purpose-Directed Analogy," Proceedings of the Seventh Annual Conference of the Cognitive Science Society, Irvine, CA, 1985.

- Keller, R., Development of a Framework for Contextual Concept Learning, in *Machine Learning, A Guide to Current Research* (Kluwer Academic Publishers, Boston, 1986), pp. 127-132.
- Laird, J., P. Rosenbloom, and A. Newell, "Towards Chunking as a General Learning Mechanism," Proceedings of the Conference on Artificial Intelligence, AAAI84, pp. 188-192.
- Laird, J., P. Rosenbloom, and A. Newell, "Chunking in SOAR: The Anatomy of a General Learning Mechanism," in *Machine Learning Journal 1*, (Kluwer Academic Publishers, Boston, 1986), pp. 11-46.
- Langley, P., G. Bradshaw, and H.A. Simon, "Data-Driven and Expectation-Driven Discovery of Empirical Laws," Proceedings of the Canadian Society for Computational Studies of Intelligence, (Saskatoon, Saskatchewan, 1982), pp. 137-43.
- Langley, P., J. Zytkow, G. Bradshaw, and H. Simon, "Three Facets of Scientific Discovery," Proceedings of the International Conference on Artificial Intelligence, IJCAI83, pp. 465-468.
- Lebowitz, M., "Integrated Learning: Controlling Explanation," *Cognitive Science 10*, 219-240 (1986).
- Lenat, D.B., "The Nature of Heuristics," *Artificial Intelligence 19*, (1), 189-249 (1982).
- Lenat, D.B., "Theory Formation by Heuristic Search: The Nature of Heuristics II: Background and Examples," *Artificial Intelligence 21* (1), 31-59 (1983).
- Lenat, D.B., "EURISKO: A Program that Learns New Heuristics and Domain Concepts—The Nature of Heuristics III: Program Design and Results," *Artificial Intelligence 21* (1), 61-98 (1983).
- Lenat, D.B. and J.S. Brown, "Why AM and EURISKO Appear to Work," *Artificial Intelligence 23* (3), 269-294 (1984).
- McCarthy, J., "Applications of Circumscription to Formalizing Common Sense Knowledge," *Artificial Intelligence 28*, (1), 89-116 (1986).
- McCarthy, J., "Circumscription—A Form of Non-Monotonic Reasoning," *Artificial Intelligence 13*, 81, 27-40 (1980).
- Michlaski, R., J. Carbonell, and T. Mitchell, (eds.) *Machine Learning* (Tioga Publishing Co., 1983).
- Michalski, R., J. Carbonell, and T. Mitchell, (eds.) *Machine Learning II* (Tioga Publishing Co., 1986).
- Mitchell, T., R. Keller, and S. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning Journal 1* (Kluwer Academic Publishers, Boston, 1986), pp. 47-80.
- Mitchell, T., "Toward Combining Empirical and Analytical Methods for Inferring Heuristics," LCSR-TR-27 Technical Report, Laboratory for Computer Science Research, Rutgers University, New Brunswick, NJ, 1982.
- N. Nilsson, *Principles of Artificial Intelligence* (Tioga Publishing Company, 1980).

- J. Reggia, "Abductive Inference," Proceedings of the Symposium on Expert Systems in Government, Oct. 1985.
- A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," in *Computer and Thought*, E.A. Feigenbaum, and J. Feldman, (eds.), (McGraw-Hill, New York, 1963), pp. 71-105.
- P. Thagard, *Computational Philosophy of Science* (Bradford Books, MIT Press, in press).
- P. Thagard, and K. Holyoak, "Discovering the Wave Theory of Sound," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI85, pp. 610-612.
- Thagard, P., private communication.
- Webster's New Twentieth Century Dictionary Unabridged* (Simon and Schuster, New York, 1983).
- P. Winston, *Artificial Intelligence* (Addison-Wesley Publishing Company, Inc., 1984), 11, Ch. 12.
- P. Winston, "Learning New Principles from Precedents and Exercises," *Artificial Intelligence* 19 (3), (1982), pp. 321-350.
- Zadeh, L. "Common Sense Knowledge Representation Based on Fuzzy Logic," in *Computer*, Oct. 1983.