# A Summary of Machine Learning Papers From IJCAI-85

Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, OR 97331

Nicholas S. Flann
Department of Computer Science
Oregon State University
Corvallis, OR 97331

David C. Wilkins
Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

This report reviews the 31 papers on machine learning that were presented at the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85) held in Los Angeles during August, 1985. The papers are grouped according to a taxonomy of the various subareas of machine learning research. The areas receiving the most attention at IJCAI-85 included learning apprentice systems and methods of explanation-based learning, although virtually all areas of machine learning research were represented. The paper describes some opportunities for further research, especially in the area of discovering new terms. The wide variety and high quality of the papers demonstrates that machine learning is a very healthy field of research.

# 1 Introduction

The growing enthusiasm surrounding work in machine learning evidenced itself at the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85) held in Los Angeles during August, 1985. In addition to the first tutorial on machine learning, conference attendees heard 31 learning papers—more than in any previous IJCAI. A total of 75 learning papers were submitted (making it the fourth most popular category, after expert systems, natural language, and knowledge representation). All areas of work in machine learning—from interactive knowledge acquisition to automated theory formation—were well represented.

Because of the diversity of topics, it is difficult to pinpoint the most interesting or significant papers presented at the conference. Sound, high quality research was presented in all areas. That in itself is a significant trend, because it demonstrates that the area of machine learning is attracting competent newcomers at a good pace. The area of explanation-based learning continued to receive lots of attention. Perhaps the most significant new development has more to do with methodology than science—namely, the focus on "learning apprentice systems". The paper by Mitchell, Mahadevan, and Steinberg discusses the notion of a learning apprentice as a learning system that watches an expert use a knowledge-based problem solving system. The learning system can acquire new knowledge by observing the ways that the expert solves problems. It can even interrupt the expert to ask some questions if it is having difficulty. Smith, et al. describe another learning apprentice system. In addition to the work on learning apprentices, many other areas were also covered at the conference. Rather than attempting to select a few papers for in-depth review, we have decided to present a broad description of all of the different areas of machine learning that were presented at the conference.

Table 1 shows the way that we have broken down the various parts of machine learning. The top division is between automatic learning methods and interactive methods. The interactive systems tend also to focus on knowledge acquisition in support of expert systems. Under the interactive systems, we examine (a) systems that interview the user and (b) learning apprentice systems.

Under the heading of automatic learning systems, we break things into empirical (inductive) and analytic (deductive) methods. The empirical methods, of course, have received the most attention in the past, and many different aspects of empirical learning were discussed at the conference including (a) theories of inductive learning, (b) methods for performing specific tasks (clustering, concept learning, etc.), (c) experimentation, and (d) formation of new terms. Work in analytical learning is more focused, with two papers discussing the learning of macro operators and several papers discussing methods and problems in the area of explanation-based learning.

# 2 Interactive knowledge acquisition systems

Interactive knowledge acquisition systems focus on interacting with an expert in order to create, debug, modify, or extend a knowledge-based expert system. Two basic strategies have been pursued. One approach (exemplified by Bennett's ROGET system; Bennett, 1985) concentrates on the interactive design of new knowledge systems. The program interviews the expert in order to acquire the basic structure of the task. The result produced by the program is a prototype expert system to which more knowledge can be added. The other approach to interactive knowledge acquisition takes over at this point. It aims to assist the expert as he or she adds knowledge (or modifies existing knowledge) to improve the performance of the expert system. This approach was first pursued by Davis in Tieresias (Davis, 1982). Recently, the term "learning apprentice system" has gained prominence as a way of describing this second kind of learning system.

```
Learning and Knowledge Acquisition

1. Interactive methods
   a. Interviewing systems
   b. Learning apprentice systems
2. Automatic methods
   a. Empirical methods
      i. theory
      ii. specific tasks
          - conceptual clustering
          - concept learning
          - learning search heuristics
          - learning expressions
          - scientific theory formation
          - control of systems
      iii. learning by experimentation
      iv. formation of new terms
   b. Analytic methods
      i. macro operators
      ii. explanation-based learning
```

Table 1: Taxonomy of machine learning topics at IJCAI-85.

### 2.1 Interactive acquisition and design of knowledge-based systems

Three papers presented new results in this area. Kahn, Nowlan, and McDermott describe MORE, a system for acquiring the design of diagnostic systems. Like ROGET, MORE has a fairly strong set of expectations for how diagnostic systems are constructed. It begins with categories such as hypotheses (i.e., basic causes of faults or diseases—the goal of the diagnosis), symptoms, tests, test conditions, symptom attributes, and symptom-conditions. By interviewing the expert, it constructs a kind of object/relation model of the domain. It has eight strategies for eliciting the information from the expert. The body of the paper is a discussion of these strategies.

Marcus, McDermott and Wang describe a more primitive system, SALT, which acquires the initial knowledge base for a configuration system. Configuration systems, such as R1 and VT, operate by constructing (configuring) a solution from a starting set of components subject to various domain-specific constraints. SALT is basically a menu-based editor that knows about the structure of three key types of expertise for configuration systems: methods for configuring a component, constraints on configurations, and fixes—that is, expertise about how to adjust a configuration when some constraint is violated. SALT generates OPS5 rules which are then incorporated into a "configuration system" shell.

Lanka describes a method for automatically constructing a database schema from examples of how the database will be used. This is a new kind of task that has not previously appeared in the machine learning literature. The task is similar to that addressed by ROGET and MORE in that it involves identifying the key objects and attributes in the domain and inferring their relationships. However, it is significantly different in that the key constraints being applied by the learning system are linguistic. The learning system analyzes example English questions that will be asked of the

database, and infers the structure of the database from these questions. For example, from the question "Which faculty teach a course in the CIS department?", the system infers that "faculty" and "course" are both entities and "course" is a function of "department". The paper describes various problems that arise when different examples yield conflicting analyses.

### 2.2 Learning Apprentice Systems

Apprenticeship learning systems were identified as a fruitful area of research in Tom Mitchell's Computers & Thought lecture at IJCAI-83. These systems are defined as "interactive knowledge-based consultants that directly assimilate new knowledge by observing and analyzing the problem solving steps contributed by their users through their normal use of the system." Two papers discussed learning apprentices at IJCAI-85.

Mitchell, Mahadevan, and Steinberg describe a learning apprentice system for VLSI circuit design called LEAP. This system is likely to be the precursor to a whole generation of apprentice learning systems. This direction should be a healthy one for the machine learning field. It studies learning in the context of actual problem solving, it provides a nice framework for tapping into the rich knowledge that human experts possess, and it emphasizes the justification of acquired heuristic rules via a deep domain theory. The VLSI design system is an "abstract-refinement" design system (in the terminology of Mostow, 1985). Its design operators show how to refine a goal into subgoals. The body of the paper for the most part duplicates Mahadevan's paper (see below) in which he shows how to apply the methods of explanation-based learning to acquire new goal-subgoal rules. Readers should study the Mahadevan paper first, and then read the Mitchell, et al. paper (skipping the duplicate sections).

Smith, Winston, Mitchell, and Buchanan describe a learning apprentice for the Dipmeter Advisor expert system at Schlumberger. The Dipmeter Advisor is similar to many diagnostic expert systems. As in Teiresias, when the Advisor produces an answer that is incorrect, the learning apprentice is engaged to repair the failure. Unlike Teiresias, which relied on the expert to localize the missing or incorrect rule, the system described by Smith, et al., employs knowledge of the justifications for the rules in order to accomplish this task. Rules that are justified as being "definitional" cannot be in error. Rules that are justified as "theoretical" are supported by a half-order theory. They are only questioned if no other possible errors can be found. Other rules, such as "default" rules and "abductive" rules have more complicated justifications that can be checked. If the justifications hold, then the rules are assumed to be ok. Once the error is found, the system decides whether to ignore it (as being a statistical outlier) or to modify the knowledge base so that it won't repeat the error. In the latter case, it currently enters an interactive rule editor and allows the expert to modify the indicated rule.

Both of these papers describe ambitious projects that are still under very active development. The papers deserve careful study, and we can expect to hear more about these projects in the future.

## 3 Automatic Learning Systems

These systems are intended to acquire knowledge without any interaction with a human user. We first describe work on inductive (empirical) learning systems. Then, we review the work on deductive (analytical) learning systems.

## 3.1 Inductive learning systems

### 3.1.1 Theory

Two papers presented theoretical results on inductive learning. In his paper, Valiant extends his previous work on the learnability of concepts in polynomial time. One of the most interesting aspects of this work is Valiant's probabilistic criterion for determining when a learning system has successfully learned a concept. Let D+ and D– be the sets of all possible positive (and negative) examples of the concept (respectively). Valiant views the learning system as an agent existing in some world where it is more or less likely to encounter these various positive and negative instances. Hence, he assumes probability distributions over the D+ and D– sets, distributions that indicate the probability of observing any particular training instance. His criterion doesn't insist that the program discover the "right" concept definition, but rather that it find a concept description that handles all of the most common cases. There are two kinds of errors that the learning program could make: (a) incorrectly labeling a positive example as negative (type 2 error) and (b) incorrectly labeling a negative example as positive (type 1 error). Valiant says that a program has learned a concept when the total probability of making a type 1 error is less than 1/h and the total probability of making a type 2 error is also less than 1/h, for some positive constant h. He is interesting in algorithms for finding such concepts in time polynomial in h and in t (the number of descriptors in the concept language). In his paper at IJCAI, he describes extensions to his algorithms that (a) handle existentially quantified DNF expressions and (b) handle modest amounts of noise (but not both). He also sketches how to implement an approximate version of his basic algorithm as a connectionist model. Finally, Valiant presents some results concerning the learnability of "rules of thumb"—that is, concept descriptions that handle only half of the examples. He shows this to be very difficult in general.

The second paper of a theoretical nature is Segen's discussion of learning in the presence of noise or exceptions. Segen is proposing a "bias" for resolving the tradeoff between correctness and generality in noisy learning situations. This bias, based on Kolmogorov complexity, views all concept descriptions as Turing machine programs. Each such program is completely consistent with the data. Some programs may achieve this consistency by using a lookup table containing all of the training instances that have been observed. Other programs might consist of general rules that cover all of the data. Still other programs will use a combination of a general rule and a table of exceptions. This is the approach that Segen advocates. His bias prefers the shortest program consistent with the training instances. He sketches a method for constructing the "general rule" part of such a program as a conjunction of features. The features are selected incrementally to satisfy the property of ultimately producing the shortest program.

### 3.1.2 Specific learning tasks

Thus far in our review, we have classified the IJCAI papers according to their learning methods. However, for inductive learning, the number and diversity of methods makes this very difficult. In this section, we instead resort to classifying the papers according to the learning tasks they are addressing.

**Concept learning** This is one of the oldest areas of machine learning, and only one paper was presented that described a practical algorithm for concept learning. Arbab and Michie describe a modification to Bratko's AODCL (itself an extension of Quinlan's ID3 system) to improve the efficiency of the descriptions that it produces. ID3 is one of the most successful induction algorithms, and it tends to produce concept descriptions (in the form of decision trees) that can be evaluated

very efficiently. Unfortunately, these decision trees are very difficult for people to understand. Bratko observed that "linear" trees—that is, trees in which every node has at most one child that is not a leaf node—are much easier to understand. He defined an index of linearity and modified ID3 to construct decision trees of maximum linearity (subject to the constraint of consistency with the data, of course). Arbab and Michie show that the sometimes there are several trees of maximal linearity, and they augment the algorithm to prefer the most efficient of these maximal trees.

**Conceptual Clustering** Two papers discussed methods of conceptual clustering. Fisher and Langley present a nice review of approaches to this learning task. They identify three (interdependent) tasks that must be addressed in conceptual clustering methods: aggregation, characterization, and formation of hierarchies. Aggregation is the task of partitioning the training instances into sets of disjoint clusters of similar items. This step is unique to clustering. Characterization is the problem of finding compact, general descriptions of the items in a cluster. This is very similar to the problem of learning a single concept from examples. And many algorithms address a third task of constructing a hierarchy of clusters according to their mutual similarity. Fisher and Langley point out that most clustering methods do not solve these three problems independently. Rather, most methods employ strategies—such as top-down discrimination and characterization—that accomplish all three tasks simultaneously. The paper describes these various strategies and explores other important aspects of clustering algorithms.

The second paper, by Phelps and Musgrove, describes a domain-specific approach to clustering two-dimensional binary images of people and animals. The method is interesting because it gives additional weight to image features that occupy more space in the image. Thus, it prefers to compare the "torsos" of people and animals first, because this occupies the most area in the image. Then, it will compare the head and limbs, and so on. The result is a classification hierarchy of objects that parallels the size hierarchy of body parts.

**Search Heuristics** Ever since the first presentation of Mitchell, Utgoff, and Banerji's LEX system (at IJCAI-81), there has been a significant amount of attention devoted to the learning of search heuristics in forward-searching problem solvers. IJCAI-83 saw the presentation of at least five papers describing such systems. Now, the flurry of interest seems to have died down. There is only one paper at IJCAI-85 on this topic—that of Neves. Neves' ALEX system learns operators and search heuristics for simple algebraic problem solving by analyzing worked-out problems in textbooks. In this sense, it is like VanLehn's Sierra system, which learned to subtract as well as to solve simple algebraic equations. ALEX is much simpler, however. It begins with a set of operators for the physical manipulation of equations (e.g., adding and deleting single terms). By analying worked-out problems, it learns (a) legal operators (e.g., add something to both sides of an equation), (b) recognizers for the legal operators (rules that compare the before and after equations and decide which operator was applied), and (c) search heuristics for operators. The learning process is unusual in that it is a non-incremental one-shot process. A fixed sequence of generalizing transformations is applied to generalize a specific example of an operator-application to obtain a search heuristic. Once constructed, these search heuristics are never modified. The generalization transformations appear to be very domain-specific.

**Expressions (function induction)** One view of the general problem of induction is that the learner is given a set of training examples, each of which specifies the values for a set of variables. The task of the learner is to notice systematic patterns relating the variables. Ideally, the relations are highly constraining, so that, given the values for some of the variables, values can be predicted

for the other variables. In concept learning, one of the variables indicates the "class" of the training instances, and the task is to predict its value given the values of the remaining variables. In the work of Langley and his colleagues on the BACON system, the task was to find functional relationships among numerically-valued variables. Functional relationships can be represented as arithmetic expressions, so it makes sense, in such domains, to search a space of arithmetic expressions. Brian Falkenhainer at this IJCAI described work which extends the work of Langley, et al., in two directions. First, he describes two heuristics (proportionality graph and units analysis) that are useful for constraining the search of expressions. Second, he presents a method for learning piecewise functional relations (i.e., the overall function is made up of an exclusive disjunction of separate functions over different regions of the instance space).

**Control systems** Another area of long-time interest in AI is learning control systems. These are real-time control systems that have some kind of adaptive capabilities. Three papers related to this topic were presented at the conference. Selfridge and Sutton describe a system that learns to control a simple robot cart running between two stops (in one dimension). The task is to balance an upright pole on the cart and avoid touching the stops. The cart is controlled through the application of a constant force in either of the two directions of movement. The learning task involves learning when to apply these forces as a function of the position of the cart, the angle of the pole to the vertical, the velocity of the cart, and the rate of change of the angle. The "concept" is represented as a vector of weights that is applied to the vector of the observable input values. The learning algorithm adjusts these weights in response to feedback from the environment whenever the pole falls or the cart strikes one of the stops. The overall learning task is quite difficult, because of the sparseness of the feedback. However, the authors show how learning can be accellerated by first teaching the system a simple task (e.g., balancing on a longer track, with a heavier pole, etc.) and then presenting it with increasingly more difficult tasks. There is significant transfer from one task to the next, and overall learning time of the final task is shortened.

The other two papers describe learning systems based on the genetic algorithm, a general search method created by John Holland. The papers summarize Ph.D. dissertations; they provide a good introduction to this flavor of machine learning research. One paper, by Goldberg, describes an application to the control of a natural gas pipeline. The other paper, by Schaffer and Grefenstette does not deal exclusively with control, but since it also employs the genetic algorithm, we have included it here.

When the genertic algorithm was first adapted to work with rule-based systems by John Holland in 1978, it searched a space of rules. Beginning with the work of Smith in 1980, several researchers have investigated searching the space of production system programs (sets of rules). For this latter case, Schaffer and Grefenstette have discovered that on multi-objective problems, such as pattern classification with more than two pattern classes, the search does not converge to an adequate program. Correcting this problem is shown to require the use of multidimensional measures of program fitness by the learning critic. Modifications to the genetic algorithm to allow this are described.

Goldberg applies a learning classifier to gas pipeline control and inertial object control. His system uses the genetic and bucket brigade algorithms to apportion credit among rules. The bucket brigade is an analog to an internal service economy with bidding and action. It was also developed by Holland and is as important an idea as the genetic algorithm itself. This paper does not extend the theory of genetic or bucket brigade algorithms, but it does provide a convincing example of their effectiveness.

**Scientific theory formation** Thagard and Holyoak describe an application of machine learning methods to issues in the history and philosophy of science. The specific case in question is the discovery of the wave theory of sound. Thagard and Holyoak have developed and implemented a reconstruction of the process by which this theory might have been discovered. Four processes are involved: (a) instance-based generalization (a simple form of concept learning in which rules of the form "All A's are B's" are learned from examples of A's that are B's), (b) condition-based generalization (simple generalization of the left-hand sides of implication rules), (c) abduction (backward chaining from observed effects to infer causes), and (d) conceptual combination (defining a new concept by combining aspects of existing concepts and resolving conflicts). Of these, conceptual combination is the most interesting. It is used to construct the concept of "sound wave" by combining properties of "sound" (e.g., spreads spherically, reflects) with the concept of "wave" (e.g., reflects, spreads in 1 or 2 dimensions). Conceptual combination differs from simple conjunction of two concepts because conflicts (like spherical vs. 1 or 2 dimensions) must be resolved.

**Episodic memory** An interesting induction system is Salzberg's Handicapper, which predicts the results of horse races. Handicapper has an episodic memory of previous horse-races, and it applies a set of four basic heuristics to generalize these memories in response to prediction failures. The heuristics are (a) blame attributes about which you know very little (unusualness, lack of knowledge), (b) blame inconsistent features, (c) prefer gradual modifications (avoid blaming attributes that would entail major changes to the memory), and (d) prefer explanations involving spatio-temporal proximity. The system starts with a weak causal model of horse racing. It learns rules that relate some conjunction of features of a horse to the probability that it will win when raced against horses with different features. The system out-performs the published predictions of experts in the Daily Racing Form.

### 3.1.3 Forming New Terms

The new term problem is, of course, a central problem of induction. At IJCAI, four papers presented new work on this problem. Judea Pearl's paper presents a method for proposing new random variables to simplify the description of a set of binary random variables. The basic idea is as follows. You are given several training instances, each of which specifies the values for a vector of binary random variables. In order to simplify the description of this data, you propose that whenever two variables appear to be correlated, there are "hidden variables" that are "causing" the observed correlation. Pearl shows how to work backward from the observed variations to define these hidden variables in such a way that the observed variables are conditionally independent (i.e., once the values of the hidden variables are known, no unexplained observable correlation remains). Pearl shows that any set of random variables can be "explained" in this manner by a tree of hidden variables, and he presents an algorithm for uniquely reconstructing this tree.

Heeffer describes the results of a method for learning good descriptive terms in chess. The method first analyzes each available primitive descriptor for its "satisfaction factor"—the degree to which it is satisfied by the training data. New terms are built by combining primitive terms under guidance from the user. The new terms are evaluated by an unusual method in which the program (using the terms) attempts to reconstruct a complete chess position by asking binary questions. Terms (i.e. predicates) that are commonly satisfied are converted to binary questions and asked before less commonly satisfied predicates.

Rendell also addresses the problem of finding good descriptive terms in chess. He applies a method that starts by comparing the discriminating power of primitive features with respect to the overall learning task (e.g., discriminating won from lost positions). Features with similar

discriminating power are assumed to be related in some way. Rendell's method assumes that the feature values are at least ordinal and that they are comparable (i.e., they range over the same value set). Features that appear to be similar are "superimposed" (i.e., merged and checked for combined discriminating power) to assess their similarity. The result is a grouping of features into sets of features with similar discriminating power. Conjunctive pairs of features are formed by taking the cross products of two feature sets with different discriminating power. The superposition method is again applied to evaluate the discriminating power of these conjunctive pairs. This yields a set of pairs called a "structural pattern class". This class is then extended by considering translations and rotations (on the chess board) that map one element of the class into another. In this way patterns such as "pair of diagonal supporting pawns" are constructed. This somewhat complicated method has been tested on the 15 puzzle.

Fu and Buchanan present a method for generating intermediate concepts (and associated rules) in a hierarchical knowledge base of production rules. For example, if the knowledge base contains the rules $L1 \Rightarrow M$, $L2 \Rightarrow M$, $L1 \Rightarrow H1$, and $L2 \Rightarrow H1$ (where $L1$ and $L2$ are "low level" observables, $H1$ is a "high level" conclusion, and $M$ is an intermediate concept), then their system will propose a new rule $M \Rightarrow H1$ if it does not lead to a contradiction. New intermediate concepts are proposed by naming conjunctions. If the rule base contains $L1 \Rightarrow H1$, $L2 \Rightarrow H1$, $L1 \Rightarrow H2$, and $L2 \Rightarrow H2$, then this can be re-expressed as $L1 \Rightarrow H1 \wedge H2$, $L2 \Rightarrow H1 \wedge H2$. A new term (say $G0001$) is created for $H1 \wedge H2$, and the rules are reformulated as $L1 \Rightarrow G0001$, $L2 \Rightarrow G0001$, $G0001 \Rightarrow H1$, and $G0001 \Rightarrow H2$. All of these rules have confidence factors attached, so the new rules are not necessarily deductive consequences of the old rules. Indeed, the new rules may outperform the old. The method has been tested in an expert system for diagnosing Jaundice. The new intermediate conclusions have been associated directly with existing medical concepts and they lead to better understandability and improved explanation generation.

One point that strikes the reader of these papers on new term creation is that they are very difficult to understand. At present, we lack good words for describing the specific term-formation tasks and methods. Any attempt to clarify the current muddle would be a valuable contribution to the field.

### 3.1.4 Learning by experimentation

A major problem with existing inductive learning methods is that they necessarily rely on a large set of training examples. These must be gathered and classified by a teacher. Research into experimentation explores the benefits and problems of having learning systems gather their own training examples. Rajamoney and DeJong describe some beginning work in this area. Their system discovers "osmosis" by placing two solutions in adjacent reservoirs separated by what turns out to be a permeable membrane. The system notices that the concentrations of the solutions change over time, and it attempts to figure out why. It knows about five fundamental physical processes and their preconditions, and it uses these to generate possible explanations by questioning the correctness of these preconditions. Discrimination experiments are constructed to determine which of the possible preconditions is being violated in the current situation. These experiments are quantitative rate experiments—that is, the physical apparatus is modified so that the relative rates of alternative processes (corresponding to alternative hypotheses) are modified to enhance one and reduce the others. Then the rate of the osmosis is measured. This isolates the correct hypothesis. The work demonstrates one particular method of hypothesis formation and experiment design. There are, of course, many other methods that need to be investigated before we obtain a good understanding of the role of experimentation in machine learning.

## 3.2 Deductive learning systems

### 3.2.1 Learning macro operators

Two very similar papers on learning macro operators were presented at this IJCAI. Both papers address the problem of WHEN to save an operator sequence as a macro move. When macros are saved and considered in the problem solving process just like any other operator, problem solving can be slowed because of the cost of checking all of the available macros. Clever indexing schemes can lessen this problem to some extent, but the need to index the macro both by the starting state and by the desired goal state (or region) makes this very difficult. Minton's system saves macros in two different situations. First, so-called "S-macros" are constructed whenever a given sequence of operators has been used twice. Operator sequences are always saved, but they aren't converted into macros until the same sequence is encountered a second time. Once they are converted into macros, they are maintained in a pool of macros of fixed size, with macros being replaced according to an LRU discipline. The second situation in which macros are saved is when the macro encodes a "trick"—that is, a sequence of operators that overcomes a local maximum in the value of the static evaluation function (i.e., by taking a few "bad" moves to get to an even better state). These "T-macros" are always created, and they are never forgotten. Minton demonstrates the value of his macros on a synthetic robot planning problem.

Iba independently discusses what he calls the "peak-to-peak" heuristic, which amounts to creating "T-macros" that span local "valleys" in the values of the static evaluation function. Iba shows the power of these macros for solving the "HI-Q" puzzle.

### 3.2.2 Explanation-based Learning

There has lately been a significant wave of interest in explanation-based learning (EBL). In this paradigm, the learning system observes some unusual or surprising event. It proceeds to construct an explanation for the event, and then it generalizes this explanation so that it covers an entire class of similar events. In most of the current work, the "explanation" is actually a proof, based on prior knowledge, that the unusual event should have occurred. Hence, the system already knew, in principle, that the event would occur, and the goal of explanation-based learning is to make that knowledge explicit and efficiently-applicable. At IJCAI, six different papers were presented that focused on explanation-based learning.

Two papers focused on learning by reading and explaining stories. Dolan and Dyer describe an explanation-based learning system that learns "morals" by reading fables. Their representation for "morals" views them as heuristics about planning errors to be avoided. For example, in the famous fable about The Fox and the Crow, the Fox tricks the Crow into dropping a piece of cheese that it is holding in its mouth. The Fox does this by appealing to the Crow's vanity—he praises the Crow's voice and asks it to sing. The author's learning system acquires the concept of being SUCKERED (i.e., the planning failure of allowing someone else to take advantage of your dormant goals by providing one of the missing enablement conditions on that goal) by combining the existing concepts of CONF-ENABLE (failing to maintain a goal of possessing something by attempting to achieve a second goal) and ULTERIOR (failing to detect an ulterior motive in an ordinary request). The new concept is generalized by analyzing the constraints inherited from ULTERIOR and CONF-ENABLE. Constants are variablized, subject to the inherited constraints.

DeJong is one of the originators of the explanation-based learning paradigm. He and Mooney give a comprehensive description of their system for acquiring new planning schemata by reading stories. Their favorite example—learning the concept of kidnapping—involves combining existing concepts of theft and bargaining. They give a clear and detailed description of the methods by

which their system attempts to understand a new story about kidnapping. The understanding process is virtually a theory-formation process in which the system searches for motives for the actors. Preexisting plan schemata are employed to find these motives, and the resulting structure is then analyzed and generalized. The generalization process is similar to the "lifting" process employed in STRIPS. First, all constants are turned to variables. Then, the various constraints attached to the preexisting schemata are re-imposed. Only those constraints needed to maintain the causal connections between the schemata and the well-formedness of the final concept are included.

Two other papers explored EBL in the context of learning about electronic circuits. Mahadevan describes the methods employed to learn goal/subgoal refinement rules for the learning apprentice of the Rutgers VLSI expert system VEXED. In this learning situation, the learning apprentice has observed the expert perform a refinement that it has never seen before. The learning system starts by attempting to prove that this refinement will actually work (i.e., verify the refinement step). If this proof succeeds, then the proof is generalized by a process of goal regression. A generalized version of the goal of the circuit is regressed through the proof to obtain a generalized version of the corresponding subgoals. After some additional steps, this yields a generalized goal/subgoal refinement operator. Examples are given for refining (AND (OR a b) (OR c d)) in terms of (NOT (OR (NOT (OR a b)) (NOT (OR c d)))) and for reducing an integral of a sum to the sum of two integrals. Mahadevan identifies some problems with the goal regression approach. The weakest precondition (which is normally computed by goal regression) is too weak to be useful in this domain (i.e., it usually is a disjunction). He uses the particular example to select one of the disjunctions. Also, the quality of the resulting goal/subgoal rule is determined by the generality of the proof that was developed to prove the correctness of the expert's refinement.

In a closely related paper, Ellman describes an EBL method that learns design methods in the domain of sequential logic circuits. The method is illustrated generalizing a circular shift register, supplied by the expert, into a general circuit schema capable of computing any permutation of the input bits. Ellman's circuits are described in a high level language of multiplexers and registers, in contrast to the work of Mahadevan, which works at the gate level. As above, the first step involves applying a theory of how all of the circuit components work to verify that the circuit satisfies the given functional specification. Then, the functional specification is generalized by replacing constants, such as "time1" with variables, to form an over-general specification. The specification is constrained by regressing it (back-propagating it) through the proof tree. The shortcomings of the proof-based paradigm are apparent in Ellman's example, because, although it does generalize the circuit substantially, it is not able to generalize the four-stage shift register to an "n"-stage shift register. This is because the proof was not sufficiently general.

Shavlik describes yet another EBL system. This one models the process of learning about conservation of momentum in physics class. The system initially has a powerful mathematical problem solving capacity and knowledge of Newton's laws of motion. Problems are presented to the system, which it attempts to solve. Upon failure (which is always due to search constraints, not lack of basic knowledge), the system prompts the teacher for a worked solution, which is analyzed to prove that it is correct, and then generalized and remembered for future use. The system departs somewhat from the basic EBL framework described in the previous four papers in the way that it constructs the proof of correctness of the teacher's solution. The system has a kind of qualitative theory of equation solving that includes such notions as "cancellation". It observes that the teacher needs to cancel the quantity that we call the momentum. The teacher employs an instance of the equation for conservation of momentum (i.e., that the total momentum of the system is unchanged over time) in order to obtain the needed quantity for cancellation. Hence, the system focuses its efforts on verifying the equation of conservation of momentum. It is able to do this by reference to Newton's laws. By analyzing the proof tree, it is then able to determine that the equation is valid

for any number of bodies and in the presence of external forces.

The final paper on explanation-based learning reviews one of the basic mechanisms involved: goal regression. Most of the five previous papers can be viewed as employing goal regression to generalize the specific proof that has been developed. The task of goal regression can be described as follows. Given a sequence of operators ($S$) and a goal ($G$), determine the weakest precondition ($P$) of the goal with respect to the operators. In other words, determine the minimum conditions ($P$) that must be satisfied prior to the application of the operator sequence ($S$) in order that the goal ($G$) will be achieved.

Porter and Kibler review this task and describe two methods for performing it. The first method is analytic constraint back-propagation. For each operator $OP$ in the operator sequence $S$, a regression operator $OP^r$ is defined. Each regression operator maps intensional descriptions of goal states into intensional descriptions of weakest preconditions for operator $OP$. For example, if $OP$ is the STRIPS operator "unstack(x,y)", which unstacks block $x$ from on top of block $y$, and $G$ is the goal $clear(C)$, then the regression operator $OP^r$ maps $clear(C)$ into the weakest precondition $(y = C) \vee clear(C)$. Given regression operators for each operator in the sequence S, it is easy to compute the weakest precondition $P$ with respect to the goal $G$. We simply apply the regression operators in reverse order, progressively back-propagating $G$ until we arrive at the start of the operator sequence, where we obtain $P$.

Porter and Kibler point out several important conditions that must be met in order for analytic constraint back-propagation to be feasible. First, it is necessary to obtain regression operators for each operator in the sequence $S$. This can either be accomplished by providing such regression operators to the learning system or by having the learning system automatically derive them by analyzing the definitions of the forward operators. In LEX-II, for example, Utgoff provided the system with the necessary regression operators. Porter and Kibler discuss in detail the problem of automatically deriving the regression operators. For operators represented as STRIPS rules (and similar representations, such as the authors' relational models), this can be accomplished fairly easily. However, if the operators are represented as arbitrary procedures, it is a very difficult (if not impossible) task.

The second requirement for performing analytical constraint back-propagation is that the goals be represented in some explicit, relational form. The authors mention this in passing, but it is certainly true that if the goal $G$ is only represented as a recognition procedure, then it will be almost impossible to analytically back-propagate it through the operator sequence.

The third requirement discussed by Porter and Kibler is that the representation language for describing goals must be closed over the set of regression operators. In other words, it must be possible to regress an arbitrarily chosen goal through any sequence of regression operators and still be able to capture the resulting weakest precondition within the representation language. They demonstrate that the representations employed by Utgoff in LEX-II and by Minton in his Gomoku system both lacked this property, which places limits on the generality of those systems.

The second method for accomplishing the task of goal regression is empirical goal regression. Porter and Kibler describe their method, which works by applying the operator sequence $S$ in the forward direction (to selected starting states) to gather training examples of its behavior. The learning system then classifies these training examples depending on whether or not they lead to the desired goal $G$. The positive training examples are then generalized (via ordinary inductive inference) to obtain an approximate description of the weakest precondition $P$ for achieving the goal $G$. This method avoids all three of the requirements listed above for the analytical constraint back-propagation method. Regression operators are not needed, the goal $G$ can be represented by an opaque procedure, and the representation language need only be capable of representing a "useful partial characterization" of the weakest precondition, rather than a completely accurate

one.

Their method does have one important requirement of its own, however. It must be possible for the learning system to construct representative starting states to which it can apply the operator sequence $S$. Porter and Kibler describe their method of "perturbations" which solves this problem by taking a teacher-supplied example and perturbing it in various (pre-specified) ways to obtain additional training examples.

It is worth noting that this empirical goal regression method is closely related to the method employed in LEX-I by Mitchell, Utgoff, and Banerji. The problem generator in LEX-I employs two fairly sophisticated methods for generating starting states. These starting states are then processed to yield positive (and negative) examples of the successful application of a single operator. The examples are generalized, via the version space algorithm, to construct heuristics that recommended when the single operator in question should be applied. The chief difference between LEX-I and the Porter and Kibler approach is that the heuristics learned by LEX-I are not "weakest preconditions" but rather stronger statements that say not only when application of the operator will lead to a solution, but also when the operator will lead to a solution more quickly than any other operator.

In summary, the paper by Porter and Kibler raises important points about the method of constraint back-propagation and describes an interesting alternative method. They call for further study of methods that combine analytical and empirical goal regression.

One important terminological point arises in reading the paper by Porter and Kibler. When they speak of regression operators, they employ the term "inverse operator." Previous papers in this area (e.g., Utgoff, 1983) have used this term as well, but it is very misleading. If an operator $OP$ maps elements of a domain set $D$ into a range set $R$, then the inverse operator, $\mathrm{OP}^{-1}$) maps elements of $R$ into $D$. This is very different from a goal regression operator, which maps intensional descriptions of subsets of $R$ into intensional descriptions of subsets of $D$. In fact, it is possible to construct regression operators for operators that lack true inverses. In his dissertation, Utgoff began using the term "backwards operator" to refer to regression operators. Unfortunately, this term has already been used by Nilsson (1980) to describe backward-chaining production rules. In this review, we have employed the term "regression operator," which is consistent with Waldinger's (1977) use of the term "regression rule" to describe his mechanism for computing weakest preconditions. The study of regression operators and their relationship to other logical and algebraic rules of inference is an important area for future research.

## 4 Concluding remarks

This concludes our review of the sessions on learning and knowledge acquisition at IJCAI-85. As the preceding pages show, research in machine learning has been vigorous and fruitful, and we can look forward to a continuation of good work in this area.

## 5 References


Bennett, J. S., ROGET: A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system, *Journal of Automated Reasoning*, Vol 1, 49-74, 1985.

Davis, R. and Lenat, D., Knowledge-based systems in artificial intelligence, McGraw-Hill, 1982.

Mostow, D. J., Toward better models of the design process, *AI Magazine*, Vol. 6, No. 1, 44-57.

Nilsson, N. J., *Principles of Artificial Intelligence*, Palo Alto, CA: Tioga Press, 1980.

Utgoff, P. E., Adjusting bias in concept learning, *Proceedings of the International Workshop on Machine Learning*, University of Illinois, 1983.

Waldinger, R. Achieving several goals simultaneously. In *Machine Intelligence 8*, Elcock, E. W., and Michie, D., (eds.), New York: Halstead and Wiley, 1977.