MULTISTRATEGY LEARNING

by

R. S. Michalski

Encyclopedia of Microcomputers, A. Kent and J. G. Williams (Eds.), Vol. 12, New York, Marcel Dekker, 1993.

ENCYCLOPEDIA OF MICROCOMPUTERS

VOLUME 12

EXECUTIVE EDITORS
ALLEN KENT
JAMES G. WILLIAMS

ENCYCLOPEDIA OF MICROCOMPUTERS

EXECUTIVE EDITORS

Allen Kent *James G. Williams*

UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

ADMINISTRATIVE EDITORS

Carolyn M. Hall *Rosalind Kent*

PITTSBURGH, PENNSYLVANIA

VOLUME 12

Multistrategy Learning to Operations Research, Microcomputer Applications

MARCEL DEKKER, INC. NEW YORK • BASEL • HONG KONG

CONTENTS OF VOLUME 12

Contributors to Volume 12 *vii*

MULTISTRATEGY LEARNING
Ryszard S. Michalski and Gheorghe Tecuci 1

MULTIUSER PROGRAMMING
Linda S. DeBrunner 29

MUMPS
William J. Harvey and Frederick G. Kohun 45

MUSIC DESCRIPTION LANGUAGES
Mira Balaban 71

MUSIC PRINTING
Keith A. Hamel 81

NATURAL LANGUAGE PROCESSING
Klaus K. Obermeier 95

NATURAL LANGUAGE UNDERSTANDING
Inger Lytje 133

NATURAL RESOURCE MANAGEMENT AND AGRICULTURE, APPLICATIONS OF ARTIFICIAL INTELLIGENCE
Michael S. Saunders, Robert N. Coulson, and L. Joseph Folse 149

NETWORK-BASED SIMULATION MODELS, ANIMATION OF-see ANIMATION OF NETWORK-BASED SIMULATION MODELS
Volume 1, pages 139-149

NETWORK OPTIMIZATION ON MICROCOMPUTERS
Richard S. Barr 163

NETWORK TESTING SYSTEM FOR DIGITAL DATA NETWORKS-see DIGITAL DATA NETWORKS; NETWORK TESTING SYSTEM
Volume 5, pages 99-122

NETWORKS–DISTRIBUTED COMPUTING
W. Anthony Mason 187

A NEURAL ALGORITHM FOR DOCUMENT CLUSTERING
Kevin J. MacLeod 199

NEURAL NETWORKS
Young B. Park 215

NEURAL NETWORKS–CONNECTIONISM
Peter Dayan 229

NOISE PHENOMENA IN VLSI CIRCUITS
Arthur Van Rheenen and Kostas Amberiadis 253

NORMALIZED OBJECT ORIENTED METHOD
Henri Habrias 271

THE OBJECT-ORIENTED PARADIGM
Chenho Kung 287

OBJECT RECOGNITION AND VISUAL ROBOT TRACKING
Luc J. Van Gool, Dao-Bin Zhang, and André Oosterlinck 307

OFFICE INFORMATION SYSTEMS DESIGN-see AUTOMATED OFFICE SYSTEMS DESIGN *Volume 2, pages 67-124*

ONLINE INVESTING USING THE MICROCOMPUTER
Deanna K. Daniels and Michael D. Chase 345

OPEN SYSTEMS INTERCONNECTION
Ray Denenberg 353

OPERATING SYSTEMS-see DOS *Volume 5, pages 151-240*

OPERATIONS RESEARCH, MICROCOMPUTER APPLICATIONS
Peter C. Bell 375

MULTISTRATEGY LEARNING

INTRODUCTION

Building systems with learning abilities is one of the central goals of artificial intelligence. The research in this direction started already in the 1950s and over the years grew into the very active and broad field of *machine learning*. While most research in this field has been concerned with *monostrategy* learning systems (employing one primary type of inference and a single computational paradigm), in recent years, there has been a growing interest in *multistrategy learning* systems. Such systems integrate two or more types of inference and/or computational paradigms, and can potentially apply to a much wider range of learning tasks than monostrategy learners. They also resemble more closely human learning, which is intrinsically multistrategy.

This article reviews major ideas and methods in the area of multistrategy learning. To introduce the reader to the subject, it first generally characterizes learning processes, and gives a brief review of several monostrategy learning methods that summarizes their interrelationships and limitations. Subsequently, it surveys some representative multistrategy learning systems. It then outlines an *Inferential Theory of Learning*, which provides a theoretical foundation for building multistrategy learning systems (1). Finally, it presents an inference-based framework for building multistrategy learning systems that adapt their learning behavior to a given learning task (2). For further reading, the Bibliography provides a selection of representative papers on multistrategy learning.

WHAT IS LEARNING?

Learning is a multifaceted process that comprises a wide range of capabilities. A general characterization is to say that it is a goal-oriented modification of one's knowledge by exploring one's experience. Such a modification can involve any type of knowledge transformation and inference, such as deduction, induction, or analogy. The type and the way the inference is performed depends on what external input is provided to the learning process, what the goals of learning are, and what the learner already knows. Consequently, the information flow in a learning process can be characterized by the schema shown in Figure 1.

In each learning cycle, the learner analyzes the external input information in terms of the learner's *background knowledge* (BK). Such knowledge is the part of the total learner's prior knowledge that is relevant to the current learning process. This analysis involves performing inferences on the input and BK, and aims at generating new knowledge that satisfies the learning *goal*. In *symbolic* learning systems (e.g., in rule learning systems), the inferences are usually performed in a more or less explicit way, while

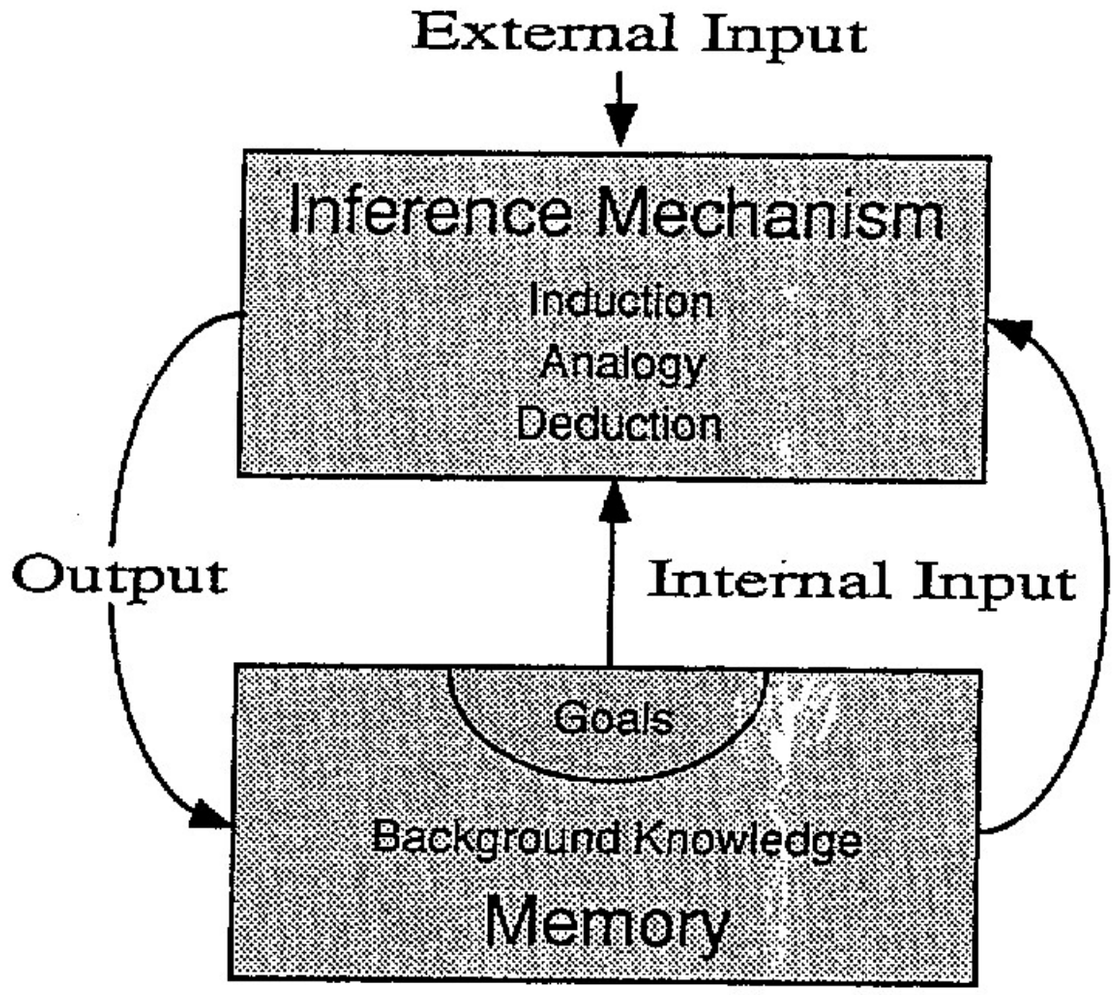

FIGURE 1 An illustration of a general learning process.

in *subsymbolic* systems (e.g., neural nets or genetic algorithms), they are performed implicitly. This means that individual transformations may not correspond to some identifiable inference rules, but the final results can be viewed as equivalent to performing inferences, e.g., they may represent a generalization of the input.

A learning goal may be one of three general types: to change the current knowledge to a more effective or operational form, to acquire new knowledge, or to validate some piece of the current (uncertain) knowledge. If the results of a given learning step (Output) satisfy the learning goal, they are assimilated within the learner's BK, and become available for use in subsequent learning processes.

In sum, to learn, an agent has to be able to perform inference, and has to possess the ability to memorize knowledge. The ability to memorize knowledge serves two purposes: to supply the background knowledge needed for performing the inference, and to record "useful" results of inference. Without either of the two components—the ability to reason and the ability to store and retrieve knowledge from memory—no learning can be accomplished. Thus, one can write an "equation":

Learning = Inferencing + Memorizing

It should be noted that the term "inferencing" means here any possible type of reasoning, including any knowledge manipulation, random searching for a specified knowledge entity, etc.

The double role of memory, as a supplier of background knowledge, and as a depository of results, is often reflected in the organization and the structure of a learning system. For example, in the simplest learning systems, such as *decision tree* learning systems, the BK includes available attributes, their legal values, and a method for attribute selection. The learned knowledge is in the form of a decision tree whose nodes correspond to individual attributes, and leaves to decision classes. In *neural nets*, background knowledge is determined by the structure of the network, i.e., by the number of

processing units (which perform some transformation of the input signals to a unit, e.g., a sigmoid function of the weighted sum of the inputs), the way they are interconnected, and by the initial weights of the connections between units. The learned knowledge resides in the new values of the weights. In a "self-contained" rule learning system, all background knowledge and learned knowledge would be in the form of rules. A learning process involves modifying prior rules and/or creating new ones.

As indicated above, a learning process depends on the input information, background knowledge, and the learning goal. These three components constitute a *learning task*. An input can be sensory observations or knowledge from some source, e.g., a teacher, or knowledge from the previous learning step. The background knowledge can be in many forms—stated facts, concept instances, previously formed generalizations, conceptual hierarchies, certainty measures, or any combinations of such types.

A learning goal is a necessary component of any learning process. Given an input, and nontrivial background knowledge, a learner could potentially generate an unbounded number of inferences. To limit the proliferation of choices, a learning process has to be constrained and/or guided by the learning goal. A learning goal determines what parts of prior knowledge are relevant, what knowledge is to be acquired, in which form, and how the learned knowledge is to be evaluated. In addition to the general types of goals mentioned above, there can be many more specific learning goals, such as to acquire a piece of knowledge that answers a given question, to derive a specific knowledge from the general knowledge, to generalize given observations, to acquire control rules to perform some activity, to reformulate given knowledge into a more effective form, to confirm a given piece of knowledge, etc.

MONOSTRATEGY VERSUS MULTISTRATEGY LEARNING

Most of research in machine learning has been concerned with monostrategy methods that employ a single inference type and computational paradigm, and are oriented toward some specific class of learning problems. A large number of such methods, or their variants, has been developed to date, such as learning decision trees or decision rules from examples, explanation-based generalization, empirical determination of equations fitting a given set of data, neural network learning from examples, genetic algorithm-based learning, case-based learning, abductive learning, and others. Recent research progress on such methods have been reported by several investigators (3–14).

Each of these methods addresses a specific class of learning problems, corresponding to a certain learning task (as defined above). For example, methods for *empirical concept learning* assume that the input consists of a number of (positive and/or negative) examples of a concept to be learned, BK contains information about the description space (attributes or predicates used for describing examples), and the goal is to learn a general concept description. Such a description is an *inductive* generalization of the positive examples that does not cover any negative examples. Different methods of empirical learning use different computational paradigms, such as decision tree, neural net, decision rules, etc., but they all address that same class of problems. In *explanation-based generalization*, the input may consist of only one example of a concept C, but BK has to contain a complete *abstract* concept definition and domain rules relating it to *operational* terms (that can be directly measured), and the goal is to learn an operational concept description, which is a *deductive* derivation from BK and the input example. In

case-based learning, the input are examples (cases), BK contains inference rules for determining the *similarity* between difference cases, and the goal of learning is to store and index past cases for future retrieval and comparison with new ones. In *abductive learning*, the input may be some fact, BK contains causal knowledge related to the input, and the goal is to hypothesize knowledge that would explain the input. In *conceptual clustering*, input is a set of entities or observations, BK contains information about the properties of these entities, and the goal is to create a partitioning of the entities into certain meaningful classes, and to provide a general description of the classes.

The above characterization of monostrategy methods shows that they are quite versatile, but each of them is useful only if applied to a designated class of problems. It also illustrates a complementary nature of their requirements. The latter suggests a possibility that by properly integrating various monostrategy methods, one could obtain a synergistic effect, that is, different methods would mutually support each other in such a way that together could be applied to a wider range of learning problems than the "sum" of monostrategy methods.

The possibility of such a synergistic integration is a major motivation for the development of multistrategy learning systems. Another important reason is that human learning is intrinsically multistrategy. Thus, by understanding the role of different learning strategies and the methods for their synergistic integration, one might be able to more adequately model human learning. The development of the theory and methodology for building multistrategy learning systems is therefore a fundamental long-term objective for machine learning research. The next section surveys a selection of existing multistrategy learning systems.

MULTISTRATEGY LEARNING SYSTEMS

Multistrategy learning systems built so far represent different methods for integrating some monostrategy methods, and are oriented toward relatively limited tasks. Most of these systems integrate some method for symbolic empirical induction (which usually requires many input examples and a small amount of BK) with a method for explanation-based generalization (which requires only one input example, and a large amount of BK). The goal was to develop systems that could learn from fewer input examples (as compared with the empirical induction) and a smaller amount of BK (as compared with explanation-based generalization). Some current multistrategy systems also include some form of abductive learning, or a simple method of analogical learning. Below is a brief characterization of several well-known systems.

UNIMEM (15) combines empirical induction, which is used to form a generalization of examples, with explanation-based learning, whose role is to explain and verify the generalization for consistency with existing domain knowledge, and to remove the unexplained features.

IOE (16) learns a concept from a set of positive examples by first constructing explanations of the individual examples, and then by empirically generalizing these explanations.

ML-SMART (17) employs a domain theory to improve and operationalize give abstract concept descriptions. The original domain theory may be incomplete and/

or incorrect, unlike in "pure" explanation-based generalization. The concept description is improved using a set of positive and negative examples.

GEMINI (18,19) and KBL (20,21) combine explanation-based learning and symbolic empirical induction to complete a domain theory by performing a theory-guided induction over a limited set of examples. GEMINI was designed for symbolic classification problems, while KBL for engineering domains involving real numbers and mathematical expressions.

ODYSSEUS (22,23) extends an incomplete domain theory by using empirical induction and abduction.

PRODIGY (24,25) is a general problem solver and learner that integrates learning by analogy, explanation-based learning, learning by abstraction, and learning by experimentation.

OCCAM (26) is an interactive multistrategy system that learns to predict and explain the outcome of events. It integrates explanation-based learning, abductive learning (based on a causal domain theory), and symbolic empirical induction.

MOBAL (27) represents an application of multistrategy learning to domain modeling, which is based on a balanced interaction between the system and the user in which different learning strategies contribute to the development of system's background knowledge, to enhancing the domain knowledge, and to the knowledge validation.

CLINT (28,29) is an interactive concept learner and theory revisor, which integrates deduction, induction and abduction using the Horn clause-based knowledge representation and reasoning (an inductive logic programming paradigm). The system enhances individual inferential strategies by employing techniques for changing the representation of the concepts to be learned, and by handling integrity constraints.

WHY (30) is a multistrategy system for learning and updating diagnostic knowledge. It employs learning strategies for learning from different types of knowledge. The phenomenological knowledge is used deductively, a causal model is used abductively, and the examples are used inductively.

Below, we describe in greater detail three other systems, which illustrate well the diversity of approaches and research issues in multistrategy learning. These include: DISCIPLE (31,32), EITHER (33), and EBL-ANN (34). The DISCIPLE system supports an interactive acquisition of knowledge from an expert, and modifies its learning behavior accordingly to its prior knowledge about the inputs received. EITHER automatically improves its initial knowledge base so that a given set of positive and negative examples is correctly classified. While both DISCIPLE and EITHER integrate different symbolic learning methods, EBL-ANN aims at integrating subsymbolic and symbolic methods.

The System DISCIPLE

DISCIPLE was designed as an advanced tool for building expert systems. It combines an expert system shell with a multistrategy learning system, both using the same knowledge base. To build an expert system with DISCIPLE, one has first to introduce into the knowledge base some basic knowledge about the given application domain (the initial background knowledge). Once this is done, DISCIPLE is ready to solve problems and improve its knowledge base by interacting with a human expert. During such cooperative

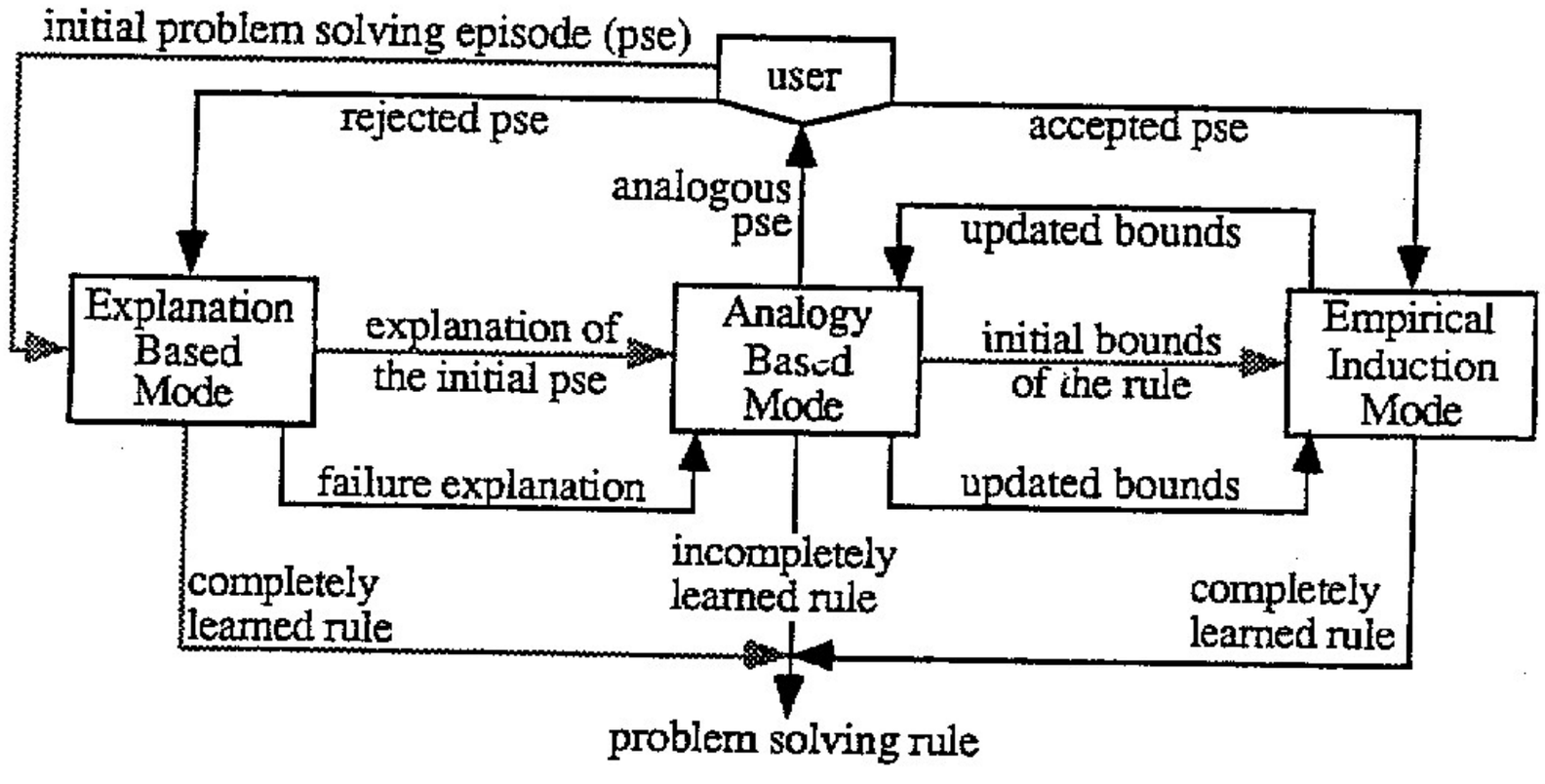

FIGURE 2 DISCIPLE's main learning steps.

problem solving processes, the system learns general problem solving rules from the problem solving steps (episodes) received from the expert, and will progressively improve its problem solving capabilities.

The learning behavior of DISCIPLE depends on how much prior knowledge it has about the problem solving episode received from the expert. This knowledge can be *complete* (if DISCIPLE can build a deductive proof of the validity of the problem solving episode), *incomplete* (if it can build only a plausible proof), or *poor* (if it cannot build any proof). In the case of complete knowledge, the learning method is purely deductive, and is a form of explanation-based learning. In the case of incomplete knowledge, the learning method involves different forms of inference, specifically, deduction, abduction, empirical induction and a simple form of analogy, and also requires a small degree of interaction with the expert. Finally, in the case of poor knowledge, the learning method integrates empirical induction and analogy, and requires a high degree of interaction with the expert. The main learning steps of DISCIPLE are presented in Figure 2.

The input to the learning process is an example of a problem-solving episode, represented by a problem and the solution indicated by the expert. The output is a general problem-solving rule. Such a rule might be complete or incomplete if the system's knowledge is insufficient to create a complete rule.

First, DISCIPLE enters the Explanation-Based Learning Mode, and tries to find an explanation of the problem-solving episode. If it is able to determine an explanation in the form of a deductive proof tree, then it deductively generalizes the problem-solving episode to a rule, using an explanation-based learning method. This ends the learning process.

If DISCIPLE is not able to find a deductive proof tree, then it looks for a plausible proof tree, using deduction, abduction, and/or analogy. If it finds a plausible proof tree (which means that it has incomplete knowledge about the input), then DISCIPLE enters the Analogy-Based Learning Mode. It generalizes the proof tree (by using a knowledge-intensive generalization procedure), and generates a reduced plausible version space for the rule to be learned. The plausible version space is the set of all the rules that could potentially be learned from the input problem solving episode. This is a partially ordered space and is represented by its two bounds, the plausible lower bound (a rule that is

supposed to be less general then the rule to be learned), and the plausible upper bound (a rule that is supposed to be more general then the rule to be learned). Each rule in this space covers only instances which are analogous to the initial problem solving episode. Then, DISCIPLE generates such an analogous instance and asks the user if it is a correct problem-solving episode or not. If the problem-solving episode is accepted by the user, then it is interpreted as a positive example of the rule to be learned. Consequently, DISCIPLE enters the Empirical Induction Mode and generalizes the lower bound of the plausible version space, so that to cover this instance. Then re-enters the Analogy-Based Learning Mode, and generates a new problem-solving episode. If the generated problem-solving episode is rejected by the user, DISCIPLE then enters the Explanation-Based Learning Mode and tries to find an explanation of failure. If such an explanation is found, DISCIPLE enters the Analogy-Based Mode. It uses the failure explanation to specialize both bounds of the version space and then generates a new problem-solving episode. If, however, no failure explanation of the rejected problem-solving episode is found, then DISCIPLE enters the Empirical Induction Mode. It interprets the rejected episode as a negative example of the rule to be learned and specializes the plausible upper bound of the version space, so as no longer to cover this episode. Then it enters again the Analogy-Based Learning Mode to generate a new problem-solving episode. The learning process ends when the two bounds of the version space become identical, and therefore the version space reduces to a single general rule. The learning process will also end when, due to the incompleteness of system's knowledge, no new useful problem-solving episodes could be generated in the Analogy-Based Mode. In such a case, the result of learning will be the current plausible version space which represents an incompletely learned rule.

If DISCIPLE cannot find even a plausible proof tree of the user's solution (because it has poor knowledge about the input), it will look for a simplified explanation in the form of a conjunction of justification statements. The justification statements may be proposed by the system (through the use of some heuristics) and accepted by the expert, or may be directly given by the expert. Then DISCIPLE will empirically generalize the explanation and will formulate a reduced version space for the rule to be learned. As in the case of incomplete knowledge, DISCIPLE will generate instances from this version space to be characterized as positive examples or as negative examples by the user. These instances are used to learn the rule.

While the main goal of DISCIPLE is to learn a general problem-solving rule from a specific problem-solving episode, it also improves the knowledge base by learning new facts and improving the existing domain rules.

By employing three learning methods (two of them integrating different learning strategies), DISCIPLE provides a solution to the so called "falling off the knowledge cliff" problem of the current systems which perform well within the scope of the knowledge provided to them, but any slight move outside their narrow competence causes the performance to deteriorate rapidly. To the contrary, in DISCIPLE, the move from one part of the application domain, characterized by a complete BK, to another part, characterized by an incomplete or inconsistent BK, causes minimal deterioration of the performance, this effect being obtained by a corresponding replacement of the learning method used.

DISCIPLE has been successfully applied to several problems, such as acquiring knowledge for loudspeaker manufacturing, acquiring qualitative chemical reactions, and acquiring rules for robot action planning.

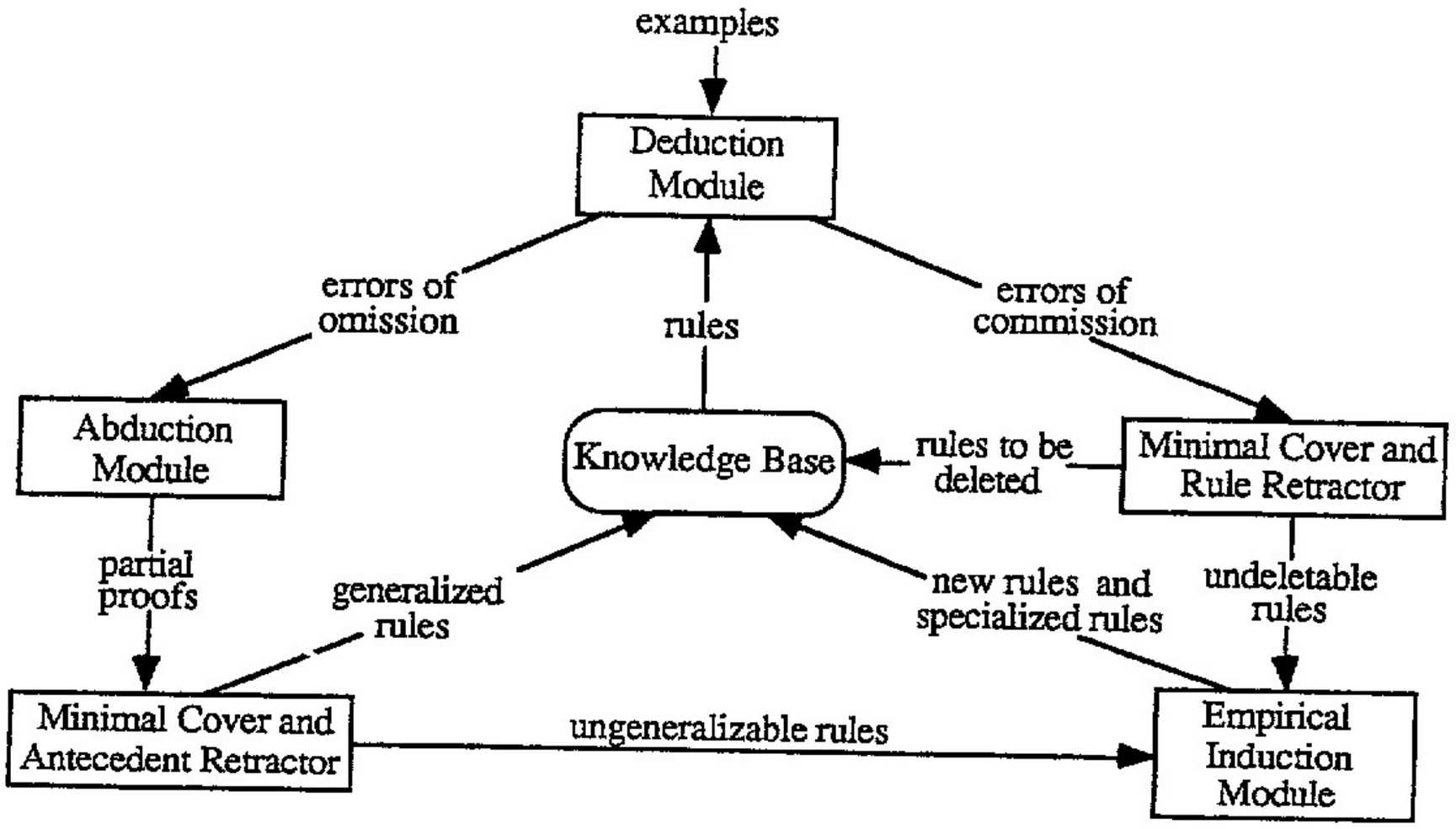

FIGURE 3 The architecture of EITHER.

The System EITHER

The EITHER system (Explanation-based and Inductive THeory Extension and Revision) employs independent modules for deduction, abduction and empirical induction to improve a knowledge base for solving classification problems (i.e., for assigning given observations to one of a predefined set of categories).

The system assumes that the initial knowledge base is partially incorrect. Its goal is to improve it by analyzing examples that are correctly classified by an expert. The knowledge base is assumed to be correct, if for every given example of some category, the system is able to build a logical proof that the example belongs to the category. If an example does not belong to the category, then the system should not be able to construct such a proof. The knowledge base is in the form of rules A & B & $\Rightarrow$. . Z, where A, B, . . . , Z are simple statements that are either true or false, and $\Rightarrow$ denotes logical implication. The knowledge base may be revised by removing rules, adding new rules, removing conditions from antecedents of the rules, or adding conditions to the antecedents.

The general architecture of EITHER is presented in Figure 3.

The input to the system are examples representing observations and their correct classification to some category. The deduction module tries to classify the examples to categories using rules from the knowledge base. Because the knowledge base is partially incorrect, the system may not be able to build a logical proof showing that a positive example of a category belongs to its category (which indicates an *error of omission*). It may also be able to build a logical proof showing that a negative example of a category belongs to its category (which indicates an *error of commission*).

The system detects errors in classification, and processes them correspondingly to their type. Errors of omission (examples of a category that the system does not recognize as belonging to this category) are sent to the abduction module. This module performs an

exhaustive search to find all partial proofs for each error of omission. These partial proofs are used to identify the conditions in the antecedents of the rules which, if deleted, would correct the problem (removing a condition from the antecedent of a rule generalizes the rule, and this may make possible to prove an example that before was unprovable). The minimal cover and antecedent retractor module uses the output of abduction to find a near-minimal set of antecedent retractions (condition removals) that would correct all of the errors of omission. Errors of commission (examples that do not belong to a category, but are classified by the system to that category), together with the corresponding proof trees, are sent to the minimal cover and rule retractor module. This module finds a near-minimal set of rule retractions (i.e., specializations of the knowledge base) that would correct all of the failing negatives.

Every modification suggested by the deduction and the abduction components (rule retractions and antecedent retractions) is tested, and accepted only if it does not have negative side effects (i.e., does not produce additional errors).

The rules that cannot be generalized or deleted without causing additional errors are sent to the empirical induction module. The empirical induction module learns new rules (in order to correct the remaining errors of omissions), or adds new antecedents to existing rules (in order to correct the remaining errors of commission).

If the retraction of an antecedent "A" from a rule "A & B $\Rightarrow$ C" would cause new errors of commission (because the knowledge base would be overgeneralized), then induction is used to learn a new set of rules for the consequent C. The positive examples of C are the positive examples that have a partial proof completed by the given antecedent retraction (i.e., the errors of omission that would have been removed by the retraction of the antecedent "A"). The negative examples of C are negative examples that become provable when C is assumed to be true (i.e., the errors of commission that would have been introduced by the retraction of the antecedent "A").

If the retraction of a rule "D $\Rightarrow$ C" would cause new errors of omission (because the knowledge base would be overspecialized), then induction is used to learn additional antecedents to add to the rule instead of retracting it. The positive examples of C are those examples that become unprovable when the rule is retracted (i.e., the errors of omission that would have been introduced by the retraction of the rule). The negative examples of C are the provable negative examples that become unprovable when the rule is retracted (i.e., the errors of commission that would have been removed by the retraction of the rule).

By combining deduction, abduction, and empirical induction, EITHER is able to handle a wide range of imperfect domain theories. The system always guarantees that the revised theory will be consistent with all the given examples. EITHER has successfully revised two expert-provided rule bases, one in molecular biology (for recognizing biological concepts in DNA sequences) and one in plant pathology (for diagnosing soybean diseases). The performed experiments with the system confirm the thesis that learning from both the domain theory and the data, has a clear advantage over learning from either the domain theory or the data alone.

The System EBL-ANN

EBL-ANN combines explanation-based learning (EBL) with artificial neural network learning (ANN). The EBL module uses an approximately correct symbolic knowledge

to explain one of the positive examples of the concept to be learned and to generalize the explanation. The result of the EBL module is an explanation structure (a general proof tree).

Based on this explanation structure, the system builds an artificial neural network. The core layout of the ANN is isomorphic to the explanation structure. Features used as inputs to the explanation structure correspond to units in the input layer of the network. Intermediate conclusions in the explanation structure correspond to hidden units, and the final conclusion becomes the output unit for the network. The units in the network are connected in the same way the inputs, intermediate conclusions, and final conclusion are connected in the explanation structure. Connections between the units that correspond to antecedents and consequents in the explanation structure are given high positive weights. Connections between the units that correspond to prohibitory antecedents and consequents are given high negative weights. After such a basic ANN is built, additional input units and low-weight connections are inserted into it, corresponding to features that were thought to be irrelevant by the approximately correct symbolic knowledge base. In particular, additional input units are added and connected to the network for every possible feature not mentioned in the explanation. Also, new links, with a randomly chosen weight within ϵ of zero, are made between units in successive layers. This produces the network which is the input to the ANN learning module.

The ANN learning module uses the training examples to teach the ANN using the backpropagation algorithm. This produces the final neural network which represents the concept to be learned. The EBL-ANN system attempts to overcome the shortcomings of both explanation-based learning and neural learning used alone. The EBL module is used to generate a justified initial configuration of the neural network. Consequently, fewer training examples are needed than if the network's initial configuration was chosen randomly. The ANN module learns from given training examples, and produces a more refined and accurate concept description, which would have not been possible to be learned by the EBL module alone. Empirical results indicate that such a combined system learns concepts more accurately, and produces better generalizations than the explanation-based system or the artificial neural net alone. It also learns much faster than a standard neural learning system.

The EBL-ANN system represents an example of using an artificial neural network as a component of a multistrategy learning system. A more general integration of symbolic and neural net learning requires developing automated methods for communicating symbolic information to a neural net, and for translating the knowledge contained in a network to a symbolic form. A schema for such a more general system is presented in Figure 4 (35).

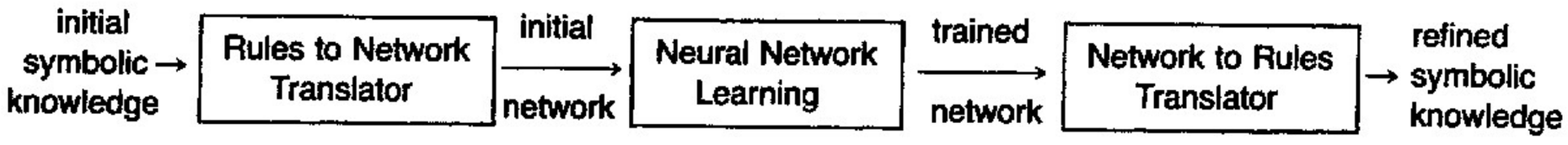

FIGURE 4 A schema for combining symbolic and neural net learning.

The figure shows a schema for using rule-based domain knowledge in combination with neural network learning. The first step creates a neural network that encodes the given domain knowledge. This knowledge need be neither complete nor correct. The sec-

ond step trains the network using a set of classified training examples. The final step extracts rules from the trained network. If the network consists of units, each of which does not have too many inputs, a simple method for creating rules that represent knowledge contained in the neural net is to use diagrammatic visualization (36).

INFERENTIAL THEORY OF LEARNING

General Presentation

The development of multistrategy learning systems should be based on a clear understanding of the roles and the applicability conditions of different learning strategies. To serve this end, the Inferential Theory of Learning (ITL) aims at explaining logical capabilities of learning strategies (i.e., their *competence*) (1,37,38). Viewing learning as a goal-oriented process of modifying the learner's knowledge by exploring the learner's experience, the theory postulates that any such process can be analyzed in terms of high-level knowledge transformation operators, called *transmutations*. Any learning process is described as a search through a *knowledge space*, conducted through an application of knowledge transmutations, i.e.:

Given	• Input knowledge	(I)
	• Goal	(G)
	• Background knowledge	(BK)
	• Transmutations	(T)
Determine	• Output knowledge, O, that satisfies goal G, by applying transmutations from the set T to input the I and the background knowledge BK	

The input knowledge, I, is the information (observations, facts, general descriptions, hypotheses) that the learner receives from the environment in the process of learning. The goal, G, specifies criteria to be satisfied by the Output knowledge, O, in order that a learning process is accomplished. Background knowledge, BK, is a part of the learner's total prior knowledge that is relevant to a given learning process. While a formal definition of "relevant" knowledge goes beyond the scope of this article, as a working definition the reader may assume that it is prior knowledge that is found useful at any stage of a learning process.

Transmutations are operators that make knowledge transformations in the knowledge space. The knowledge space is a space of all representations of all possible inputs, the learner's background knowledge, and all the knowledge that the learner can potentially generate using available transmutations. (In empirical inductive learning, the knowledge space is usually called a *description space*.) Transmutations are independent of the mechanism that performs them. They represent generic conceptual units of knowledge transformation for which one can make a mental model. They can employ any type of inference.

Let us consider a few examples. An *abstraction transmutation* takes a description of some set of entities, and transforms it to a description that conveys less information about the set (typically, an abstraction only removes information that is irrelevant for a given goal). An *inductive generalization* takes examples of a given concept, and hypothesizes a general description of the concept by conducting inductive inference. A *deductive*

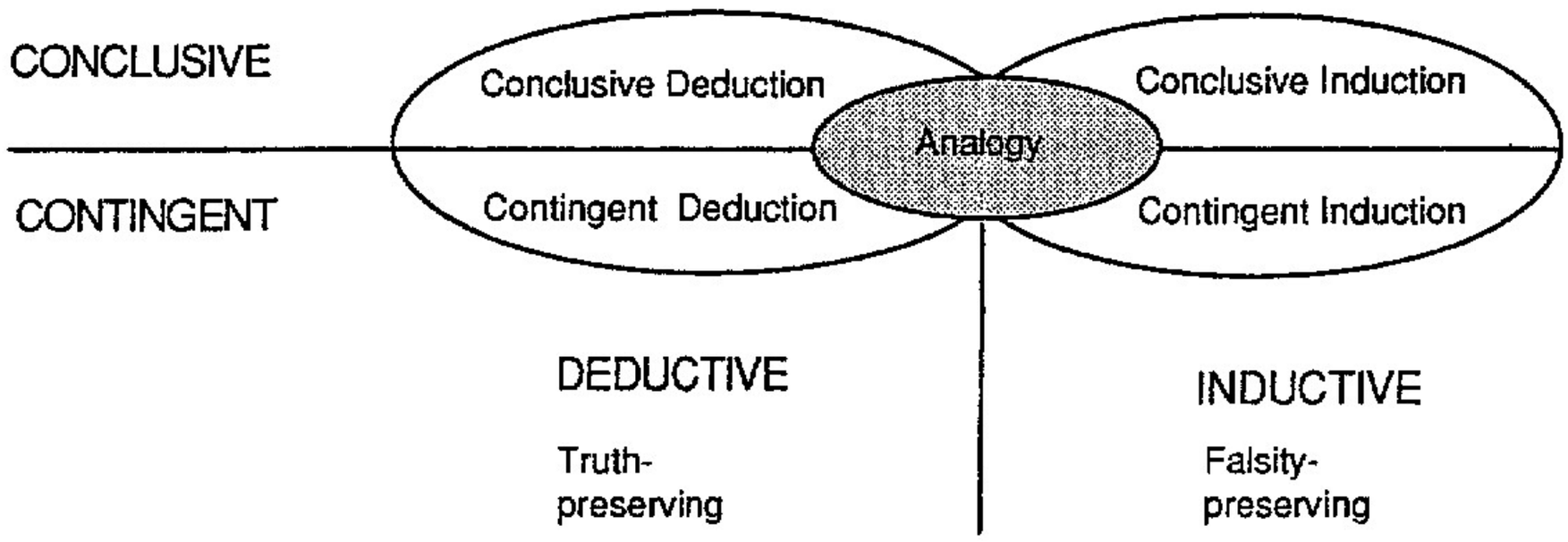

FIGURE 5 Classification of major types of inference.

generalization takes a description of some entities, and derives a description of a lar ger set of entities by exploring general knowledge it possesses about the entities. Explanation-based generalization (e.g., 39) can be viewed as an example of deductive generalization (it takes a concept example that resides in an "operational" description space, and deductively derives a generalized concept description, by exploring a known abstract concept description, and domain knowledge that links the abstract and operational description space.

Transmutations can be implemented in many different ways, depending on the underlying knowledge representation and the employed computational mechanism. In symbolic learning systems, transmutations are usually (but not always) implemented in a more or less explicit way, and executed in steps that are conceptually comprehensible. For example, the INDUCE learning system performs inductive generalization according to certain generalization rules—selective or constructive, which represent conceptually understandable transformations (40).

In subsymbolic systems (e.g., neural networks) transmutations are performed implicitly, in steps dictated by the underlying computational mechanism. These steps may not correspond to any conceptually simple operations. For example, a neural network may generalize an input example by performing a sequence of small modifications of weights of internode connections. These weight modifications are difficult to explain in terms of explicit inference rules. Nevertheless, they can produce a global effect equivalent to generalizing a set of examples.

In sum, Inferential Theory of Learning states that learning is a goal-guided process of deriving desired knowledge by using input information and background knowledge. Such a process can be viewed as a search through a knowledge space, using transmutations as search operators. When a learning process produces knowledge specified by a learning goal, it is memorized, and made available for subsequent learning processes.

Types of Inference

Any type of inference may generate knowledge that can be useful for some purpose, and thus worth learning. Therefore, a complete theory of learning must include a complete theory of inference. Therefore, a part of the effort to develop ITL is to analyze various types of inference. Figure 5 presents an attempt to schematically illustrate all major types of inference.

The first classification is to divide inferences into two fundamental types: deductive and inductive. In defining these types, conventional approaches (e.g., formal logic) do not distinguish between the input information and the reasoner's background knowledge. Such a distinction is important, however, for characterizing learning processes. Clearly, from the viewpoint of a learner, there is a difference between the information received from the senses, and the information that already resides in the learner's memory. Thus, making such distinction better reflects cognitive aspects of reasoning and learning, and leads to a more adequate description of learning processes.

To define basic types of inference in a general, language-independent way, let us consider an entailment:

$$P \cup BK \models C \tag{1}$$

where P stands for a set of statements, called *premise*, BK stands for the reasoner's *background knowledge*, $\models$ denotes logical entailment, and C stands for a set of statements, called *consequent*. It is assumed that P is logically consistent with BK.

Statement (1) can be interpreted: P and BK logically entails C, or alternatively, that C is a logical consequence of P and BK. Equation (1) succinctly explains the relationship between two fundamental forms of inference, deductive inference and inductive inference, and therefore, is called the *fundamental equation* for inference. Deductive inference is deriving consequent C, given P and BK. Inductive inference is hypothesizing premise P, given C and BK.

Deduction can thus be viewed as "tracing forward" the relationship (1), and induction as "tracing backward" this relationship. Deduction is finding logical consequences of given knowledge, and its basic form is truth-preserving (C must be true, if P and BK are true). Induction is finding premises that logically imply the input. Its basic form is falsity-preserving (if C is not true, then P cannot be true). The latter property applies to every type of induction, such as inductive generalization, abductive derivation, inductive specialization, and concretion (see below).

Based on the amount of background knowledge involved, inductive inference can be divided into *knowledge-limited* (or *empirical*) and *knowledge-intensive* (or *constructive*). A simple inductive generalization is a form of empirical induction, because it requires little background knowledge. Abduction can be viewed as a knowledge-intensive induction, because it requires background knowledge in the form of causal, or generally, implicative relationships.

In a general view of deduction and induction that also captures their approximate or common sense forms, the standard logical entailment $\models$ may be replaced by a "contingent" or "weak" entailment $\models\!\approx$ (in this context, the standard entailment is called "conclusive" or "strong"). A contingent entailment means that C is only a *plausible*, *probabilistic*, or *partial* consequence of P and BK. The difference between these two types of entailments leads to another major classification of types of inference. Specifically, inferences can be *conclusive* ("strong") or *contingent* ("weak"). Conclusive inferences assume strong entailment in (1), and contingent inferences assume weak entailment in (1). Conclusive deductive inferences (also called formal or demonstrative) produce true consequences from true premises. Conclusive inductive inferences produce hypotheses that conclusively entail consequences. Contingent deductive inferences produce consequents that may be true in some situations and not true in other situations; they are weakly truth-preserving. Contingent inductive inferences produce hypothesis that weakly entail consequences; they are weakly falsity-preserving.

The intersection of deduction and induction, that is a truth- and falsity-preserving inference, represents a equivalence-based inference (or a *reformulation* transmutation). Such an inference transforms a given statement (or set of statements) into a logically equivalent one. For example, if A is logically equivalent to A′, then the rule A ⇒ B can be transformed to rule A′ ⇒ B. Analogy can be viewed as an extension of an equivalence-based inference, namely as a "similarity based" inference (or "similization"). For example, if A is similar to **A**, then from A ⇒ B one can plausibly derive **A** ⇒ B [under the assumption that the similarity between **A** and A is *relevant* to B (e.g., they share the same properties or high-level relations that are "relevant" to B)]. Analogy occupies the central area in the diagram because deriving new knowledge by analogy can be viewed as a combination of induction and deduction. The inductive step consists of hypothesizing that a similarity between two entities in terms of certain descriptors extends to their similarity in terms of some other descriptors. Based on this similarity, and the knowledge of the values of the additional descriptors for one entity, a *deductive* step derives their values for the second entity. In order that an analogy can work, there is a tacit assumption that the additional descriptors are "relevant" in some way to the descriptors used to establish the similarity.

A Summary of Transmutations

Transmutations can be classified into two categories: *knowledge-generation* and *knowledge-manipulation* transmutations. Knowledge-generation transmutations (also called *knowledge-level* transmutations) change the informational content of knowledge, create new knowledge, determine relationships among knowledge components, organize knowledge into certain structures, etc.

In contrast, knowledge-manipulation transmutations view the input knowledge as data or objects to be manipulated, rather than statements from which new statements are to be derived. Knowledge-manipulation transmutations include inserting/deleting generated knowledge components to/from knowledge structures, physically transmitting knowledge to/from other knowledge bases, or organizing knowledge components according to some syntactic criteria. Such operations on knowledge are not classified as inferences. However, since they are performed according to well-defined algorithms that do not include generating any hypothetical knowledge, they can be formally interpreted as using deduction as the underlying inference.

Most of the transmutations are bidirectional operators, that is, they can be grouped into pairs of opposite operators. Below is a summary of knowledge transmutations that have been identified in the theory. The first eight groups represent knowledge-generation transmutations, and the remaining ones represent knowledge-manipulation transmutations.

Generalization/Specialization

A *generalization transmutation* extends the reference set of the input, that is, generates a description that characterizes a larger reference set than the input. Typically, the underlying inference is inductive, that is, the extended set is inductively hypothesized. Generalization can also be deductive, when the more general description is a logical consequence of the more specific one in the context of the given background knowledge (or can be deductively derived from the background knowledge). The opposite transmutation is *specialization*, which narrows the reference set. Specialization usually employs deductive inference, but there can also be an inductive specialization.

Abstraction/Concretion

Abstraction reduces the amount of detail in a description of an entity (an object, or a class of objects). To do so, it may change the description language to one that uses more abstract concepts which express the properties relevant to the reasoner's goal, and omit other information. The underlying inference is deduction. An opposite transmutation is *concretion*, which generates additional details about a given entity. Concretion typically utilizes an inductive inference.

Association/Disassociation

Association transmutation determines a dependency between entities or descriptions based on the observed facts and/or background knowledge. The dependency may be logical, causal, statistical, temporal, etc. Associating a concept instance with a concept name is an example of association transmutation. The opposite transmutation is *disassociation* that asserts a lack of dependency. For example, asserting that a given instance is not an example of some concept is a disassociation transmutation.

Similization/Dissimilization

Similization derives new knowledge on the basis of the similarity between various reference sets. The similization transfers knowledge from one reference set to another reference set which is similar to the original one in some viewpoint. The opposite operation is *dissimilization* that derives new knowledge on the basis of the lack of similarity between the compared reference sets. The similization and dissimilization transmutations are based on the patterns of inference presented in the theory of plausible reasoning by Collins and Michalski (41). For example, knowing that England grows roses, and that England and Holland have similar climates, a similization transmutation is to hypothesize that Holland may also grow roses. An underlying background knowledge here is that there exists a dependency between the climate of a place and the type of plants growing in that location. A dissimilization transmutation is to infer that bougainvilleas probably do not grow in Holland because Holland has very different climate than Caribbean Islands where they are very popular. These transmutations can be characterized as a combination of inductive and deductive inference.

Selection/Generation

A *selection* transmutation selects a subset from a set of knowledge components that satisfies some criteria. For example, choosing a subset of relevant attributes from a set of candidates, or determining the most plausible hypothesis among a set of candidates, is a selection transmutation. The opposite transmutation is *generation*, that generates additional components of some knowledge structure. For example, generating an additional attribute based on some algorithm, or creating an alternative hypothesis, is a generation transmutation.

Agglomeration/Decomposition

The *agglomeration* transmutation groups together certain entities into larger units according to some goal criterion. If it also hypothesizes that the larger units represent some general patterns in data then it is called *clustering*. The grouping can be done according to a variety of principles, e.g., to maximize some mathematical notion of similarity, as in conventional clustering, or to maximize "conceptual cohesiveness," as in conceptual

clustering (e.g., 42). The opposite transmutation is a *decomposition* that splits a structure of entities into subgroups according to some goal criterion.

Characterization/Discrimination

A *characterization transmutation* determines a *characteristic* description of a given set of entities that differentiates these entities from any other entities. The simplest form of such a description is a list of all properties shared by all the entities. The opposite transmutation is *discrimination* that determines a description that discriminates a given set of entities from another set of entities.

Derivations: Reformulation/Intermediate Transmutations/Randomization

Derivations represent a range of transmutations that derive one piece of knowledge from another based on a dependency relationship between them. The dependency can range from the logical equivalence to random relationship, and is a part of the reasoner's background knowledge. Thus, the extreme points of this range are the reformulation and *randomization*, respectively. The reformulation transforms a segment of knowledge into a logically equivalent segment of knowledge. For example, mapping a geometrical object represented in a right-angled coordinate system into a radial coordinate system is a reformulation. A translation of a segment of knowledge from one formal language to another is also a reformulation. In contrast, randomization transforms one knowledge segment to another one by making random changes. For example, the *mutation* operation in a genetic algorithm represents a randomization. Two related intermediate transmutations are an abduction *derivation*, which derive a cause for a given effect, and predictive *derivation*, which derives an effect from a given cause. These two derivations are complimentary in the sense that they are based on the same causal dependency, but "trace it" in the opposite directions. Another intermediate derivation is *crossover* in a genetic algorithm, which derives new knowledge by exchanging two segments of related knowledge components.

Insertion/Deletion

Insertion transmutation adds a new knowledge component generated by some transmutation to a given knowledge structure. The opposite transmutation is *deletion*, which removes some knowledge component from a given structure. An example of deletion is forgetting.

Replication/Destruction

Replication reproduces a knowledge structure residing in some knowledge base in another knowledge base. Replication is used for example in *rote learning*. There is no change of the contents of the knowledge. The opposite transmutation is *destruction* that removes a knowledge structure from a given knowledge base. The difference between destruction and deletion is that destruction removes a copy of a knowledge structure that resides in some knowledge base, while deletion removes a component of a knowledge structure residing in the given knowledge base.

Sorting/Unsorting

Sorting transmutation changes the organization of knowledge according to some well-defined criteria. For example, ordering decision rules in a rule base from the shortest

(having the smallest number of conditions) to the longest is a sorting transmutations. An opposite operation is *unsorting*, which is returning back to the previous organization.

Figure 6 provides a summary of the above described transmutations, and the underlying types of inference. Depending on the type, the amount and the way the background knowledge is employed, all knowledge-level transmutations can be done by

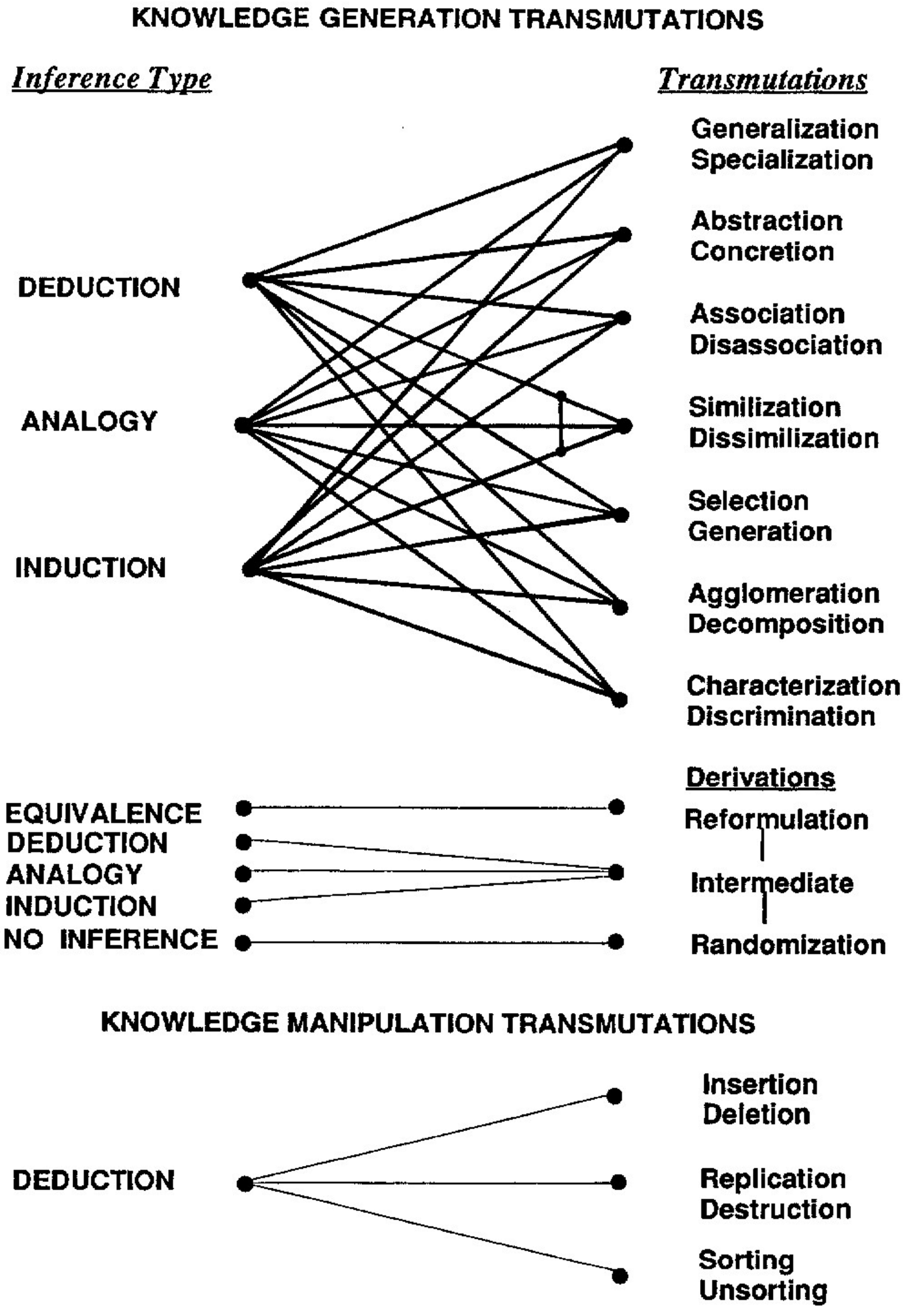

FIGURE 6 A summary of transmutations and the underlying inference types.

deduction, induction or by analogy (1). Some transmutations predominantly use one type of inference, e.g., generalization typically uses induction, and abstraction typically uses deduction. An example of generalization based on analogy is given in the next section. Similization and dissimilization transmutations are based on analogy, which can be viewed as deduction and induction combined. As mentioned above, data level transmutations can be viewed having deduction as the underlying inference.

By analyzing diverse learning strategies and methods in terms of abstract, implementation-independent transmutations, Inferential Theory of Learning offers a unifying view of learning processes. Such a view provides a clear understanding of the roles and the applicability conditions of different learning strategies, and facilitates the development of theoretically well-founded multistrategy learning systems.

MULTISTRATEGY TASK-ADAPTIVE LEARNING

Multistrategy task-adaptive learning (MTL) comprises a class of methods in which the learner determines by itself which strategy or combination of strategies is most appropriate for a given learning task. A learning task is defined by the learner's goal, the learner's BK, and the input to the learning process. In a task-adaptive learning, individual inferential strategies can be deeply integrated, so that they represent different aspects of the activity of one general system, or loosely integrated, in which case the strategies are implemented as separate modules.

This section briefly presents an inference-based framework for multistrategy task-adaptive learning (2,43). In this framework, individual strategies are integrated at the level of individual inferences (i.e., deduction, analogy, abduction, generalization, specialization, abstraction, concretion, etc.). The input to a learning process can take different forms. It may be a ground fact. It may consist of one or several positive, and/or negative examples of a concept. It may also consist of one or several positive (and negative) examples of problem solving episodes, each episode specifying a problem and its solution. The initial knowledge base of the system (which contains the BK) is assumed to be incomplete and/or partially incorrect. The goal of the system is to improve this initial knowledge base so that it can explain all the external inputs in a consistent way. This means that the inputs would be entailed by the updated knowledge. The framework is outlined in Table 1.

The system analyzes the input in terms of its current knowledge in order to build a "plausible justification tree." Such a tree demonstrates that the input is a plausible consequence of the current system's knowledge. A plausible justification tree is like a proof tree, except that individual inference steps may result from different types of reasoning (not only deductive, but also analogical, abductive, probabilistic, fuzzy, etc.).

Let us suppose that the input to a learning process is an example of some relationship, $P_n(x,y)$, where P_n is the name of a relationship holding between objects x and y:

$$P_1(a,f) \;\&\; P_2(g,a) \;\&\; P_3(b) \;\&\; P_4(h) \;\&\; \ldots \;\&\; P_j(b,c) \;::>\; P_n(a,b) \qquad (2)$$

$P_1, \ldots, P_j$ are predicates, 'a', 'f' . . . , 'c' are object names or object properties, and $::>$ denotes a *target assignment operator*, which assigns "truth" to the predicate on the right hand side (RHS), if the expression on the left hand side (LHS) is true. An example of the relationship "grows(x,y)" is:

rainfall(Thailand, heavy) & climate(Thailand, tropical) & (3)
soil(Thailand, red-soil) & terrain(Thailand, flat) &

location(Thailand, SE-Asia) ::> grows(Thailand, rice)

TABLE 1 The Learning Methodology.

- For the first positive example I_1:
  - -build a plausible justification tree T_1 of I_1
  - -build the plausible generalization T_u of T_1
  - -generalize the knowledge base so that to entail T_u
- For each new positive example I_i:
  - -generalize T_u so that to cover a plausible justification tree T_i of I_i
  - -generalize the knowledge base so that to entail the new T_u
- For each new negative example I_i:
  - -specialize T_u so that no longer to cover any plausible justification tree T_i of I_i
  - -specialize the knowledge base so that to entail the new T_u without entailing the previous T_u
- After all the examples have been processed:
  - -extract from T_u abstractions of the input examples and concept definitions

The LHS of (3) is a description of Thailand, and the RHS states that Thailand grows rice. To "understand" the example stated generally in Eq. (2), the system attempts to demonstrate that $P_n(a,b)$ is a plausible consequence of $P_1(a,f)$ & $P_2(g,a)$ & $P_3(b)$ & $P_4(h)$ & . . . & $P_j(b,c)$, in the context of the learner's BK. To make such a demonstration, the system builds the plausible justification tree in Figure 7. The root of the tree in Figure 7 is the RHS of Eq. (2). The leaves are the predicates from the LHS of eq. (2), and the intermediate nodes are intermediate predicates generated during the "understanding" process. The branches connected to any given node link this node with facts, the conjunction of which *certainly* or *plausibly implies* the fact at the node, according to the learner's knowledge. The notion "plausibly implies" means that the target (parent node) can be inferred from the premises (children nodes) by some form of plausible reasoning, using the learner's knowledge. The branches together with the nodes they link represent individual inference steps which could be the result of different types of reasoning. To indicate such inference steps in a general way, we use the symbol "⋯>". We also use "⇒" to denote logical (conclusive) implication, and "≈>" to denote plausible implication.

For example, the inference step

$$P_1(a,f) \;\&\; P_2(g,a) \cdots> P_u(a,c) \qquad (4)$$

may be a result of deduction based on the following deductive rule from the BK (Figure 7):

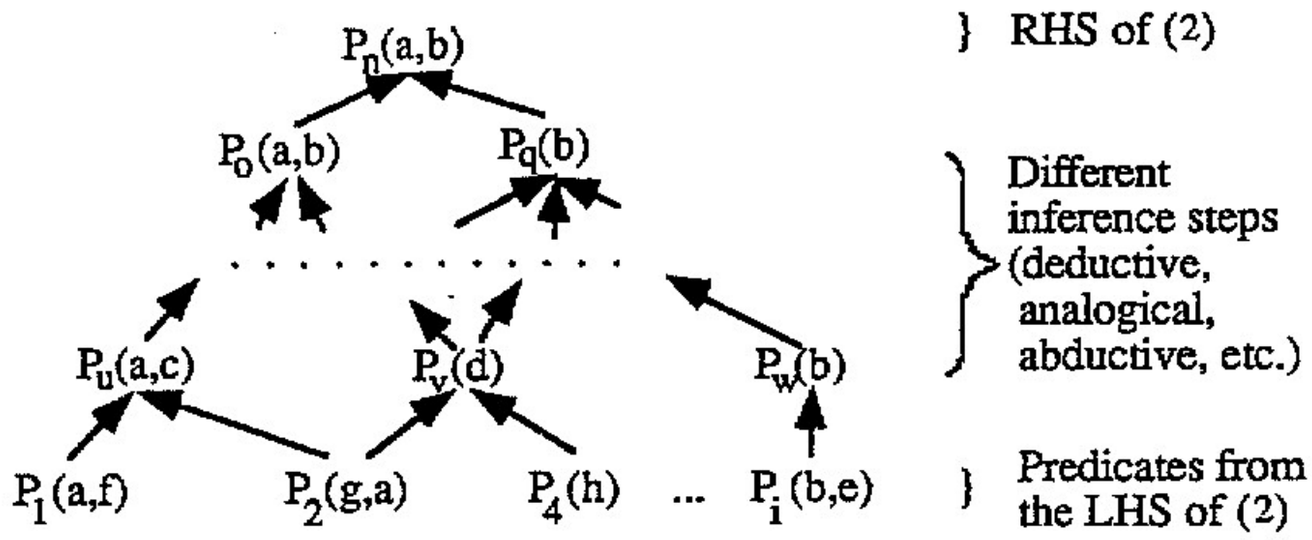

FIGURE 7 A plausible justification tree for the input $P_n(a,b)$.

$$\forall s\ \forall t\ \forall z\ \forall p\ (P_1(s,t)\ \&\ P_2(z,s) \Rightarrow P_u(s,p)) \tag{5}$$

Also, the inference step

$$P_2(g,a)\ \&\ P_4(h) \cdots> P_v(d) \tag{6}$$

may be a result of analogy with the following implication from the BK:

$$P_2(g',a')\ \&\ P_4(h') \Rightarrow P_v(d') \tag{7}$$

Indeed, suppose that the system determined that g, a, h, and d are similar to g′, a′, h′ and d′, respectively. By analogy, the system concludes that from $P_2(g,a)$ & $P_4(h)$ one can plausible infer $P_v(d)$, and hence (6).

The inference step

$$P_i(b,e) \cdots> P_w(b) \tag{8}$$

could be the result of abduction based on the following causal relationship from the BK:

$$\forall r\ (P_i(r,e) \Rightarrow P_w(r)) \tag{9}$$

Indeed, let us suppose that the predicate $P_w(b)$ needs to be true in order to build the plausible justification tree in Figure 7. Because $P_w(b)$ matches the RHS of (9), one may trace backward this rule, and hypothesize that $P_i(b,e)$ is true.

An inference step could also result from a combination of empirical generalization and deduction, which we call inductive prediction. To illustrate this, let us suppose that the knowledge base contains the following abstract examples of the relationship $P_n(x,y)$:

$$P_o(c,f)\ \&\ P_q(c)\ \&\ P_j(e) ::> P_n(c,f) \tag{10}$$
$$P_o(d,g)\ \&\ P_q(d)\ \&\ P_k(b) ::> P_n(d,g)$$

These examples can be empirically generalized to the rule

$$\forall x \forall y\ (P_o(x,y)\ \&\ P_q(y) \approx\!\!> P_n(x,y)) \tag{11}$$

Rule (11) can then be used to produce the plausible inference step:

$$P_o(a,b)\ \&\ P_q(b) \cdots> P_n(a,b) \tag{12}$$

In a more complex case, available examples may not be so easily generalizable to (11), and the system may have to use constructive induction.

Thus, the plausible justification tree in Figure 7 shows that $P_n(a,b)$ is a plausible consequence of facts from the LHS of (2). As an illustration, the plausible justification tree of example (3) is presented in Figure 8 (2):

An important result of understanding the input (i.e., building the plausible justification tree) is the generation of new pieces of knowledge which extend the knowledge base so as to entail the input.

In the case of the justification tree in Figure 7, these new pieces of knowledge are:

$P_2(g,a)\ \&\ P_4(h) \approx\!\!> P_v(d)$	(generated through analogy)
$P_i(b,e)$	(generated through abduction)
$P_o(a,b)\ \&\ P_q(b) \approx\!\!> P_n(a,b)$	(generated through inductive prediction)

By asserting these pieces of knowledge into the knowledge base, the new knowledge base entails the input. The support for these new pieces of knowledge is that they allow

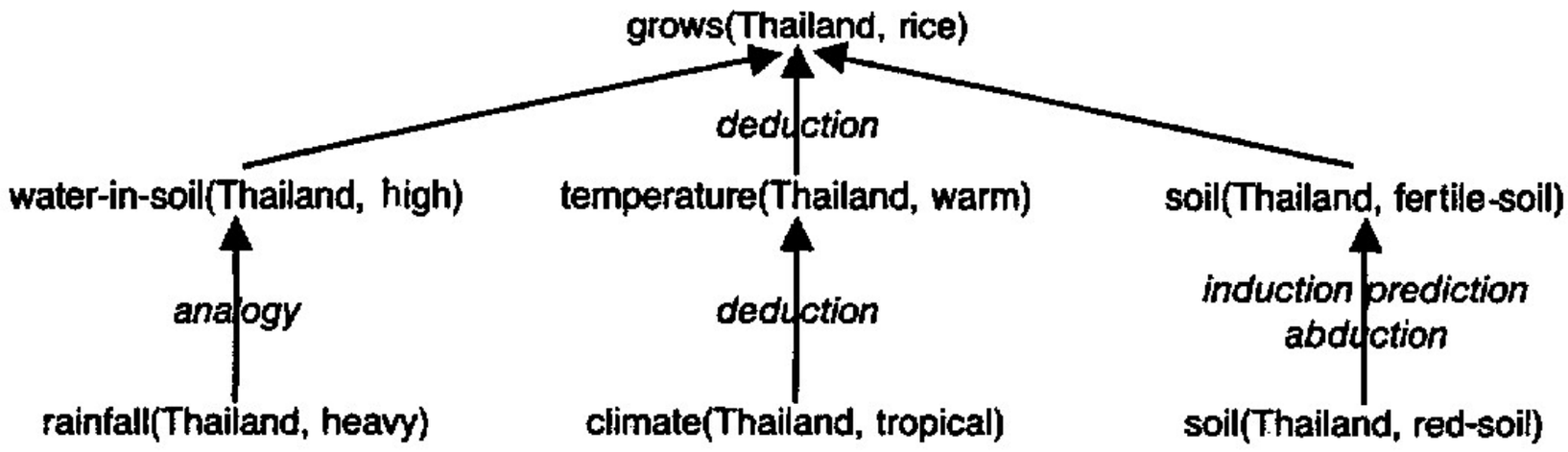

FIGURE 8 A plausible justification of example (3).

building a logical connection (a justification tree) between the knowledge base that represents a part of the real world, and a piece of knowledge (the input) that is known to be true in the real world.

The understanding process proceeds in the same way when the input is a new fact, a new relationship, or a solution of some problem. Let us suppose that the input is the relationship $P_n(a,b)$, assumed to represent a correct knowledge. In such a case, the system builds the plausible justification tree from Figure 7 that shows that the relationship $P_n(a,b)$ is a plausible consequence of the following facts and relationships contained in the knowledge base: $P_1(a,f)$ & $P_2(g,a)$ & $P_4(h)$ & . . . & $P_i(b,e)$. If the input is a specific solution S to problem P, then the plausible justification tree represents a demonstration that S solves P.

In general, a learning system should try to learn as much as possible from any input it receives. In the case of the considered learning scenario, it may do so by generalizing the plausible justification tree as much as allowed by the knowledge used to build it in the first place. By this, it will generalize the hypothesized knowledge, so that the resulting knowledge base will entail not only the received input, but also similar ones.

One way to generalize the plausible justification tree is to replace each implication with a plausible generalization, and then to unify the connections between the generalized implications. To illustrate this process, take, for instance, the inference step (4) [i.e., $P_1(a,f)$ & $P_2(g,a)$ $\cdots>$ $P_u(a,c)$]. This inference step can be locally generalized into the rule (5) that allowed it, i.e., $\forall s\ \forall t\ \forall z\ \forall p(P_1(s,t)$ & $P_2(z,s) \Rightarrow P_u(s,p))$. The branches of the tree in Figure 7, corresponding to the original inference step (4), are then replaced by the appropriate components of this rule. This is a *deductive generalization*.

Let us now consider the inference step (6) [i.e., $P_2(g,a)$ & $P_4(h)$ $\cdots>$ $P_v(d)$]. This step was made by analogy with implication (7) [i.e., $P_2(g',a')$ & $P_4(h') \Rightarrow P_v(d')$]. The generalization of this analogical inference is based on the idea that a similarity of an entity to a given entity generates an equivalence class of all entities similar to the given entity. Following this idea, one may generate a conjunctive generalization that covers all the inferences that could be derived by analogy with (7):

$$P_2(g'',a'') \ \&\ P_4(h'') \cdots> P_v(d'') \tag{13}$$

where g″, a″, h″, and d″ represent classes that contain g and g′, a and a′, h and h′, d and d′, respectively. This is a *generalization based on analogy*.

The abductive step (8) $P_i(b,e)$ $\cdots>$ $P_w(b)$ is replaced by $P_i(r,e)$ $\cdots>$ $P_w(r)$, according to the general rule (9), i.e. $\forall r\ (P_i(r,e) \Rightarrow P_w(r))$. This generalization is justified because a system abducing $P_i(b,e)$ could also abduce $P_i(r,e)$, for any r, by tracing backward the rule (9). This is a *generalization based on abduction*.

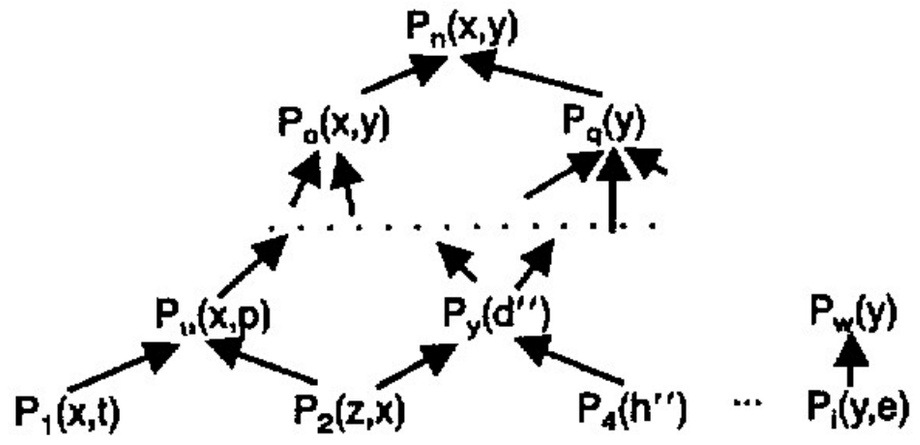

FIGURE 9 A generalization of the plausible justification tree from Figure 7.

Finally, the inference step (12) $P_o(a,b)$ & $P_q(b)$ ···> $P_n(a,b)$ was done by applying the rule (11) [i.e. $\forall x \forall y(P_o(x,y)$ & $P_q(y) \Rightarrow P_n(x,y))$], which was obtained by empirical generalization. Then the corresponding branch is replaced by rule (11). This is an *empirical inductive generalization*.

As one could notice, the generalization of an implication depends on the type of inference that generated it and of the system's knowledge. As mentioned above, after all the inference steps are locally generalized, the system unifies their connections. In particular, it makes the unifications $x=s=a''$, $z=g''$, $y=r$ and builds the general plausible justification tree from Figure 9.

To give a specific example, the generalization of the plausible justification tree in Figure 8 is indicated in Figure 10 (2).

During the generalization of the plausible justification tree, some of the previously learned knowledge may also be generalized as, for instance, the analogical implication (6) $P_2(g,a)$ & $P_4(h) \Rightarrow P_v(d)$, which was generalized to the rule (13) $P_2(g'', a'')$ & $P_4(h'') \Rightarrow P_v(d'')$. Other generalized knowledge pieces have been generated during the building of the plausible justification tree in Figure 7. An example of such a knowledge piece is the rule (11).

The generalized plausible justification tree shows how a generalization of the input is entailed by the knowledge base. However, this tree was obtained by making both plausible inferences and plausible generalizations. Consequently, both the tree and the corresponding knowledge pieces learned are less certain. One may improve the tree, as well as the knowledge pieces hypothesized during its building, by learning from additional concept examples.

For each new positive example E_i, the system will generalize the plausible justification tree in Figure 9 so as to cover a plausible justification of E_i. Also, some of the knowledge pieces from the knowledge base may be generalized so as to cover inferences from the plausible justification of E_i.

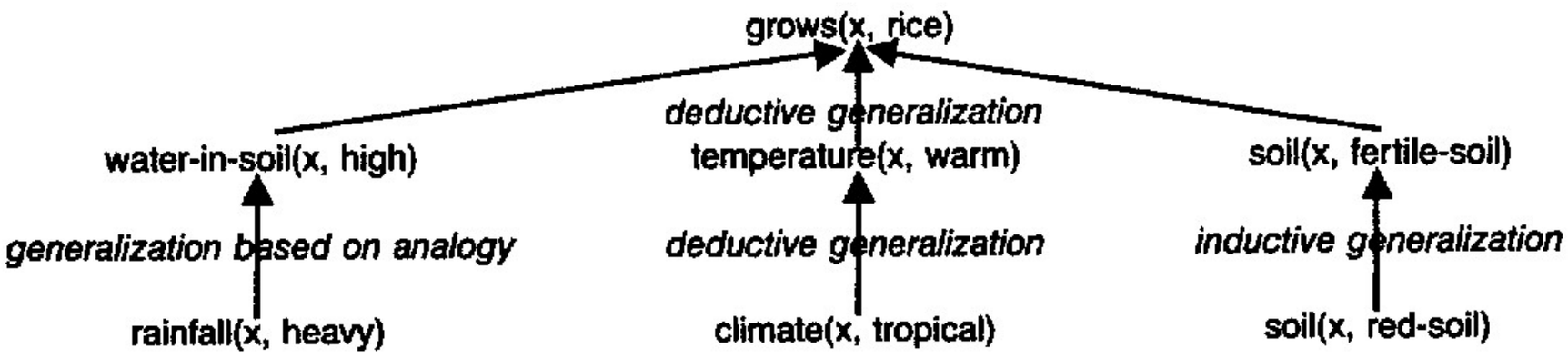

FIGURE 10 A generalization of the plausible justification tree from Figure 8.

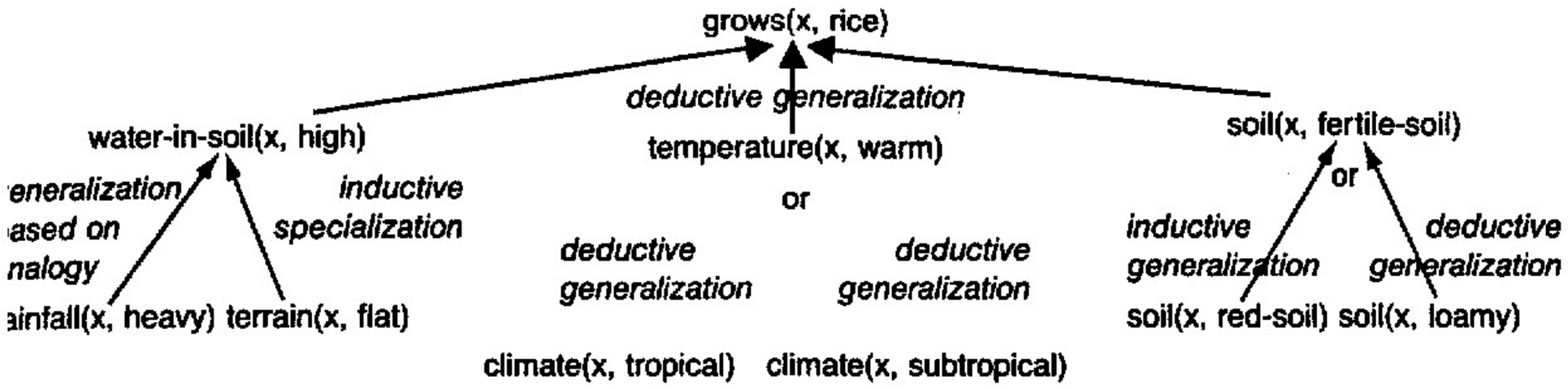

FIGURE 11 An improved general justification tree.

For each new negative example N_j, the system will specialize the general plausible justification tree so as to no longer cover a plausible justification tree which would show that N_j is a positive example. This may also require the specialization of some knowledge pieces from the knowledge base as, for instance, the rule (11) or the rule (13).

For instance, as a result of learning from the positive example (14) and from the negative example (15), the tree from Figure 10 is improved as indicated in Figure 11.

Positive Example 2: (14)

rainfall(Pakistan, heavy) & climate(Pakistan, subtropical) & soil(Pakistan, loamy) & terrain(Pakistan, flat) & location(Pakistan, SW-Asia)::>grows(Pakistan, rice)

Negative Example 3: (15)

rainfall(Jamaica, heavy) & climate(Jamaica, tropical) & soil(Jamaica, loamy) & terrain(Jamaica, abrupt) & location(Jamaica, Central-America)::> ¬ grows(Jamaica, rice)

After all the examples have been processed, the system may extract several (operational or abstract) concept definitions from the final general justification tree. For example, if the final general justification tree is the one from Figure 9, then its leaves represent an operational definition of $P_n(x,y)$:

$$P_1(x,t) \;\&\; P_2(z,x) \;\&\; P_4(s) \;\&\ldots\&\; P_i(y,e)::>P_n(x,y) \qquad (16)$$

Also, the top part of the justification tree represents the most abstract characterization of $P_n(x,y)$:

$$P_o(x,y) \;\&\; P_q(y)::>P_n(x,y) \qquad (17)$$

Other learnable knowledge pieces are various abstractions of the input examples. For instance, one abstraction is obtained by instantiating the variables in the above abstract characterization [eq. (17)], to specific arguments in a certain example:

$$P_o(a,b) \;\&\; P_q(b)::>P_n(a,b) \qquad (18)$$

Other abstractions would correspond to lower levels of the generalized justification tree. Table 1 synthesizes the main steps of this learning methodology.

As demonstrated above, the types and order of inferences performed in building a plausible justification tree depend on the relationship between the input and the knowledge base. The presented learning method thus integrates dynamically different types of elementary reasoning mechanisms. Moreover, if a particular learning task corresponds to

a monostrategy method, then the behavior of the system corresponds to the application of such a method (2).

CONCLUSION

This article described a conceptual framework and several methods for multistrategy learning—a central research direction in the field of machine learning. A major premise of the framework is that learning is a goal-oriented process in which the learner's initial knowledge is modified by exploring the learner's experience. By being able to employ in a learning process two or more inferential strategies and/or computational paradigms, multistrategy systems offer significant advantages over monostrategy learning systems. Although a number of experimental multistrategy systems have been developed, the research in this area is still at an early stage. The outlined Inferential Theory of Learning gives theoretical foundations for understanding diverse learning strategies and approaches, and for developing multistrategy learning systems. More research is needed, however, on the formalization of various concepts and further development of various aspects of theory. The inference-based framework for building multistrategy task-adaptive learning systems shows how, by dynamically integrating several types of knowledge transmutations, a system can adapt its learning behavior to the learning task. A comprehensive presentation of the current research in multistrategy learning is given elsewhere (44). Other representative papers on multistrategy learning are indicated in the Bibliography.

ACKNOWLEDGMENTS

The research on this article was done in the GMU Center for Artificial Intelligence. The Center's research is supported in part by the National Science Foundation Grant No. IRI-9020266, in part by the Office of Naval Research Grant No. N00014-91-J-1351, and in part by the Defense Advanced Research Projects Agency Grant No. N00014-91-J-1854, administered by the Office of Naval Research & Grant.

REFERENCES

1. R. S. Michalski, Inferential Learning Theory: Developing Theoretical Foundations for Multistrategy Learning, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
2. G. Tecuci, An Inference-Based Framework for Multistrategy Learning, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
3. J. E. Laird, (ed.) Proceedings of the Fifth International Conference on Machine Learning, University of Michigan, Ann Arbor, June 12–14, 1988.
4. D. Touretzky, G. Hinton, and T. Sejnowski, (eds.), Proceedings of the 1988 Connectionist Models, Summer School, Carnegie Mellon University, June 17–26, 1988.
5. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
6. D. Schafer (ed.), Proceedings of the 3rd Intern. Conference on Genetic Algorithms, George Mason University, June 4–7, 1989.

7. A. M. Segre (ed.), Proceedings of the Sixth International Workshop on Machine Learning, Cornell University, Ithaca, New York, June 26–27, 1989.
8. R. Rivest, D. Haussler, and M. Warmuth, Proceedings of the Second Annual Workshop on Computational Learning Theory, University of Santa Cruz, July 31–August 2, 1989.
9. M. Fulk and J. Case, Proceedings of the 3rd Annual Workshop on Computational Learning Theory, University of Rochester, Rochester, NY, August 6–8, 1990.
10. B. W. Porter and R. J. Mooney, (eds.), Proceedings of the 7th International Machine Learning Conference, Austin, TX, 1990.
11. Y. Kodratoff and R. S. Michalski, (eds.) *Machine Learning: An Artificial Intelligence Approach Vol. 3*, Morgan Kaufmann, San Mateo, CA, 1990.
12. L. Birnbaum and G. Collins, Proceedings of the 8th International Conference on Machine Learning, Chicago, June 1991.
13. M. Warmuth and L. Valiant (eds.), Proceedings of the 4th Annual Workshop on Computational Learning Theory, University of California at Santa Cruz, CA, August 5–7, 1991.
14. D. Sleeman and P. Edwards, Proceedings of the Ninth International Workshop on Machine Learning (ML92), Aberdeen, July 1–3, 1992, Morgan Kaufman, San Mateo, CA, 1992.
15. M. Lebowitz, Integrated Learning: Controlling Explanation, *Cognitive Sci, 10*, 219–240 (1986).
16. N. Flann and T. Dietterich, A Study of Explanation-Based Methods for Inductive Learning, *Machine Learning, 4*, 187–266 (1989).
17. F. Bergadano and A. Giordana, Guiding Induction with Domain Theories, in *Machine Learning: An Artificial Intelligence Approach, Vol. 3*, Y. Kodratoff and R. S. Michalski (eds.), Morgan Kaufmann, San Mateo, CA, 1990.
18. A. P. Danyluk, The Use of Explanations for Similarity-Based Learning, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Milan, Italy, Morgan Kaufmann, 1987, pp. 274–276.
19. A. P. Danyluk, Gemini: An Integration of Analytical and Empirical Learning, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
20. B. L. Whitehall, Knowledge-based Learning: Integration of Deductive and Inductive Leaning for Knowledge Base Completion, *Ph.D. thesis*, University of Illinois at Champaign-Urbana, 1990.
21. B. L. Whitehall and S. C-Y. Lu, Theory Completion using Knowledge-Based Learning, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
22. D. C. Wilkins, W. J. Clancey, and B. G. Buchanan, *An Overview of the Odysseus Learning Apprentice*, Kluwer Academic Press, New York, 1986.
23. D. C. Wilkins, Knowledge Base Refinement as Improving an Incorrect and Incomplete Domain Theory, in *Machine Learning: An Artificial Intelligence Approach, Vol. 3*, Y. Kodratoff and R. S. Michalski (eds.), Morgan Kaufmann, San Mateo, CA, 1990.
24. S. Minton and J. G. Carbonell, Strategies for Learning Search Control Rules: An Explanation-Based Approach, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Milan, Italy, Morgan Kaufmann, San Mateo, CA, 1987, pp. 228–235.
25. M. Veloso and J. Carbonell, Automating Case Generation, Storage, and Retrieval in PRODIGY, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
26. M. J. Pazzani, Integrating Explanation-Based and Empirical Learning Methods in OCCAM, Proceedings of the Third European Working Session on Learning, Glasgow, Scotland, 1988, pp. 147–166.
27. K. Morik, Balanced Cooperative Modeling, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

28. L. De Raedt, Interactive Concept Learning, *Ph.D. thesis*, Catholic University of Leuven, Leuven, Belgium, 1991.
29. L. De Raedt and M. Bruynooghe, Interactive Theory Revision, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
30. C. Baroglio, M. Botta, and L. Saitta, "WHY: A System that Learns Using Causal Models and Examples, in *Machine Learning: A Multistrategy Approach, Vol. 4*, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
31. G. Tecuci, DISCIPLE: A Theory, Methodology, and System for Learning Expert Knowledge, *Ph.D. thesis*, University of Paris-South, 1988.
32. G. Tecuci and Y. Kodratoff, Apprenticeship Learning in Imperfect Theory Domains, in *Machine Learning: An Artificial Intelligence Approach*, Vol. 3, Y. Kodratoff and R. S. Michalski (eds.), Morgan Kaufmann, San Mateo, CA, 1990.
33. R. J. Mooney and D. Ourston, A Multistrategy Approach to Theory Refinement, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
34. J. W. Shavlik and G. G. Towell, An Approach to Combining Explanation-Based and Neural Learning Algorithms, in *Readings in Machine Learning*, J. W. Shavlik and T. Dietterich (eds.), Morgan Kaufmann, San Mateo, CA, 1990.
35. G. G. Towell and J. W. Shavlik, Refining Symbolic Knowledge Using Neural Networks, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
36. J. Wnek and R. S. Michalski, Comparing Symbolic and Subsymbolic Learning: A Case Study, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.
37. R. S. Michalski, Toward a Unified Theory of Learning: An Outline of Basic Ideas, Proceedings of the First World Conference on the Fundamentals of Artificial Intelligence, M. De Glas and D. Gabbay (eds.), Paris, France, July 1–5, 1991.
38. R. S. Michalski, Inferential Theory of Learning as a Conceptual Framework for Multistrategy Learning, *Machine Learning J.*, *Multistrategy Learning* (special issue) 1993.
39. T. M. Mitchell, T. Keller, and S. Kedar-Cabelli, Explanation-Based Generalization: A Unifying View, *Machine Learning*, *1*, 47–80 (1986).
40. R. S. Michalski, Theory and Methodology of Inductive Learning, *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Tioga Publishing Co., San Mateo, CA, 1983.
41. A. Collins and R. S. Michalski, The Logic of Plausible Reasoning: A Core Theory, *Cognitive Sci. 13*, 1–49 (1989).
42. R. S. Stepp and R. S. Michalski, How to Structure Structured Objects, in *Proceedings of the International Machine Learning Workshop*, University of Illinois Allerton House, Urbana, June 22–24, 1983, pp. 156–160.
43. G. Tecuci, Plausible Justification Trees: A Framework for the Deep and Dynamic Integration of Learning Strategies, *Machine Learning J.* , *Multistrategy Learning* (special issue) (1993).
44. R. S. Michalski and G. Tecuci (eds.), *Machine Learning: A Multistrategy Approach*, Vol. 4, San Mateo, CA, Morgan Kaufmann, 1993.

BIBLIOGRAPHY

Bala, J., K. DeJong, and P. Pachowicz, Multistrategy Learning from Engineering Data by Integrating Inductive Generalization and Genetic Algorithms, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Cohen, W., The Generality of Overgenerality, in *Machine Learning: Proceedings of the Eighth International Workshop*, L. Birnbaum and G. Collins (eds.), Chicago, IL, Morgan Kaufmann, San Mateo, CA, pp. 490–494, 1991.

de Garis H., Genetic Programming: Evolutionary Approaches to Multistrategy Learning, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

DeJong, G and R. Mooney, Explanation-Based Learning: An Alternative View, *Machine Learning*, *1*, 145–176 (1986).

Dietterich, T. G., Learning at the Knowledge Level, *Machine Learning*, *1* (3), 287–316 (1986). Reprinted in J. W. Shavlik and T. G. Dietterich (eds.) *Readings in Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1990.

Gordon, D. F., An Enhancer for Reactive Plans, in *Machine Learning: Proceedings of the Eighth International Workshop*, L. Birnbaum, and G. Collins (eds.), Chicago, IL, Morgan Kaufmann, San Mateo, 1991, pp. 505–508.

Gould J. and Levinson R., Experience-Based Adaptive Search, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Hirsh, H., Incremental Version-Space Merging: A General Framework for Concept Learning, Doctoral dissertation, Stanford University, 1989.

Hunter L., Classifying for Prediction: A Multistrategy Approach to Predicting Protein Structure, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Josephson J., Abduction: Conceptual Analysis of a Fundamental Pattern of Inference, Technical Research Report 91-JJ, Laboratory for Artificial Intelligence Research, The Ohio State University, 1991.

Kodratoff Y., Induction and the Organization of Knowledge, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Kodratoff, Y and G. Tecuci, DISCIPLE-1: Interactive Apprentice System in Weak Theory Fields, *Proc. IJCAI-87*, Milan, Italy, 1987, pp. 271–273.

Matwin S. and B. Plante, Theory Revision by Analyzing Explanations and Prototypes, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Michalski, R. S., A Methodological Framework for Multistrategy Task-Adaptive Learning, in *Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems*, Knoxville, October 1990, Elsevier, New York, 1990.

Michalski R.S. and G. Tecuci (eds.), Proceedings of the First International Workshop on Multistrategy Approach, Harpers Ferry, WV, November 7–9, 1991, Center for Artificial Intelligence, George Mason University, 1991.

Mooney, R. and S. Bennet, A Domain Independent Explanation Based Generalizer, in *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, Morgan Kaufmann, San Mateo, CA, 1986, pp. 551–555.

Pazzani M., Learning Causal Patterns: Making a Transition from Data-Driven to Theory-Driven Learning, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Ram A. and M. Cox, Introspective Reasoning Using Meta-Explanations for Multistrategy Learning, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Reich, Y., Macro and Micro Perspectives of Multistrategy Learning, in *Machine Learning: A Multistrategy Approach*, Vol 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Russell, S., *The Use of Knowledge in Analogy and Induction*, Morgan Kaufmann, San Mateo, CA, 1989.

Saitta, L. and M. Botta, Multistrategy Learning and Theory Revision, *Machine Learning J.*, *Multistrategy Learning* (special issue) (1993).

Segen J., GEST: A Learning Computer Vision System that Recognizes Hand Gestures, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Tecuci G., Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base, *IEEE Trans. Sys., Man Cybernetics*, 22 (5) (1992).

Tecuci, G. and R. S. Michalski, A Method for Multistrategy Task-adaptive Learning Based on Plausible Justifications, in *Machine Learning: Proceedings of the Eighth International Workshop*, L. Birnbaum, and G. Collins (eds.), Chicago, IL, Morgan Kaufmann, San Mateo, CA, 1991, pp. 549–553.

Tecuci, G. and R. S. Michalski, Input Understanding as a Basis for Multistrategy Task-Adaptive Learning, in *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, Springer-Verlag, New York, 1991, pp. 419–428.

Vafaie, H. and K. DeJong, Improving a Rule Induction System Using Genetic Algorithms, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Widmer, G., Learning with a Qualitative Domain Theory by Means of Plausible Explanations, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1992.

Wisniewski, E. J. and L. M. Medin, The Fiction and Nonfiction of Features, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Wnek, J. and M. Hieb, Bibliography of Multistrategy Learning Research, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

Zhang, J., Learning Graded Concept Descriptions by Integrating Symbolic and Subsymbolic Approaches, in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (eds.), Morgan Kaufmann, San Mateo, CA, 1993.

RYSZARD S. MICHALSKI
GHEORGHE TECUCI