1988

MACHINE LEARNING: A Critique of Research Efforts and Suggested Research Strategy

William S. Davis

David B. Murrell

Report Number:
88-774

http://docs.lib.purdue.edu/cstech/663

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
MACHINE LEARNING: A CRITIQUE OF RESEARCH EFFORTS AND SUGGESTED RESEARCH STRATEGY

William S. Davis
David B. Murrell

CSD TR-774
May 1988
MACHINE LEARNING: A Critique of Research Efforts and Suggested Research Strategy

William S. Davis
David B. Murrell

Purdue University, West Lafayette, Indiana

May, 1987
(Modified May, 1988)

ABSTRACT

Learning is constructing or modifying representations of what is being experienced, and then using those representations to perform better a second time on repetition of the same task, or another task drawn from the same population. Machine Learning is concerned with developing computational theories of such learning, as well as constructing learning systems. This document is three-fold in nature:

1) a brief overview of machine learning;
2) a critique of some accomplishments in machine learning research;
3) a proposed strategy for directing future machine learning research.

The authors' overall view of machine learning is that, while much of the research results have provided a good foundation in learning theories and taxonomies of learning strategies, a true learning element for the machine is clearly lacking. This viewpoint will be evident from the critiques. The proposed strategy for research will be to narrow the focus of research to the two main problems in the field: the lack of full integration of learning strategies, and the lack of real creativity/intelligence in learning.

May 17, 1988
1. Reasons for the Necessity of Machine Learning

A practical defense for the pursuit of machine learning research can be found in the need to reduce tedium in algorithm specification. Present methods dictate that a complete and correct algorithm be coded to instruct the computer to perform a task. This is a time consuming effort and must be done by trained personnel.

As computational methods worm their way into more and more application frontiers, the threat of duplicating past efforts to transport proven algorithms into new environments increases. In order to avoid the redundancy involved in teaching these "old dogs" new "tricks" the need for self-improving algorithms must be addressed. Present systems have no capability to learn to perform a task through examples or by analogy to similar tasks whose strategy has already been determined. Neither do current systems have the capability to improve their performance significantly by observing the consequences of past errors, or to acquire new abilities. Hence, much effort is expended in re-coding existing algorithms in order to graft them into new task environments or to upgrade them to meet new requirements.

The third commonly proposed reason for machine learning is that because computer systems must interact with human beings, they should closely parallel human abilities. This argument lacks strength in comparison to the others mentioned earlier. It could be debated that because of the "internal" position and "transparent" nature of learning processes, that mimicry of human learning strategies is not essential. Also, it is conceivable that human learning strategies are non-optimal, which produces the undesirable consequence of teaching the computer "bad habits". Even in applications where computers must be taught by human instructors, human learning strategies could be a hindrance. Admittedly, these strategies would be favorable at the onset due to the familiar flavor they convey to the tutor. But it is not known whether the initial advantages will persist in the long run, when tutors themselves may be instructed in better methods for communicating their knowledge.

2. History of Machine Learning

The chronological development of machine learning is reflected in three major paradigms — the neural network paradigm (pre-eminent in the 1950's and early 1960's), the logic-based paradigm (of central attention in the 1970's), and the knowledge-rich paradigm (at present). Each of these historically prevalent themes influenced the direction of research in its particular decade.

2.1. Neural Networks

Also known as "self-organizing" and as "connectionist" approaches this paradigm attempted to capture the operation and overall structure of the web of neurons and synapses in the human brain. No real emphasis was made to discover the methodological procedures of reasoning (a top-down approach). Instead, it was hoped that if the mechanical aspects of the brain could be replicated, that reasoning capabilities would somehow appear.
Some of the largest shortcomings in these early attempts centered around the primitive state of the technology used. Hardware was not yet mature enough to meet the challenges and a few electronic neurons required a roomful of equipment. Software algorithms could not practically simulate the complex paths of signal pulses and feedback effects. Hence, early systems, such as PERCEPTRON (Rosenblatt, 1958) met with limited success.

Recently, however, with the advent of dense, inexpensive integrated circuit technologies, the neural network is regaining popularity. And, thanks to contributions from researchers like Hopfield (1982), Hinton (1986), and Fahlman (1979) neural nets are beginning to exhibit favorable prospects for the future of machine learning.

Research in neural networks led to the birth of the “Decision Theoretic Approach” to machine learning. In this paradigm, a set of training examples was used to acquire linear, polynomial, or related discriminant functions. The machine’s interaction with its environment adjusted parameter values to these functions in order to produce stable performance in the midst of disturbances. Samuel’s checkers program (Samuel, 1959) is one of the best examples of a system based on these ideas.

2.2. Logic Based Paradigm

Rather than utilizing the statistical and numerical methods prevalent in the decision theoretic approach, the logic-based paradigm attempts to symbolically describe the acquired concepts using logical axioms or graph structures. Hence, rules are generated or augmented to produce new knowledge. Meta-Dendral (Buchanan, 1978) — an extension to Buchanan’s Dendral system — is one such example. Another prime example is Michalski’s Star Methodology presented later.

2.3. Knowledge Rich Paradigm

Characteristics of this approach include constraining the scope of learning to the intended task. The earlier tabula rasa and knowledge poor systems contributed the fact that in order to acquire new knowledge, a system must already possess a preponderance of initial information. Another notable characteristic of this paradigm is that a number of learning strategies are being pursued (eg. learning from instruction, by analogy, by observation and discovery, and so on) in addition to the more traditional learning by example. Lastly, the knowledge rich paradigm is marked by the incorporation of search heuristics. These are particularly valuable to inductive learning methodologies which can generate a potentially infinite set of generalizations to describe phenomena and which tend to perform poorly in environments fraught with incomplete or inconsistent training data. Heuristic procedures serve to limit the “focus of attention” under these circumstances.

3. Objectives of Learning

Perhaps the most socially relevant objective of the study of machine learning is the gaining of insight into human learning techniques. The simulation of human learning, however inefficient, gives valuable insight into the underlying principles human beings employ to acquire knowledge. The discoveries made in this area could be used to advance education and to improve the capabilities of computer tutoring systems to adjust the presentation of their materials as they “learn” about the abilities of their students.

By and large the most popular objective among the current research community is the exploration of alternative learning methods. Much research involves the proposition of novel learning schemes and the discovery of new induction methods. There is no reason to restrict the horizons of learning to those exhibited by human beings. As Ryszard Michalski states, “In fact, common sense suggests that human learning represents just one point in an uncharted space of possible learning methods” [Michalski, 1983].

Clearly the most under-developed objective of machine learning is the analysis of existing learning theories. Typically, the concentration of the research effort has fallen within the boundaries of the second objective, stated above. Very little attempt has been made to chart the scope and limitations of these infant theories or to establish the quantity and quality of information that must be available to the learner. Queries concerned with quantity take the form of measuring the amount of exemplary training data or in
initial (background) knowledge needed for the system to arrive at useful (i.e. unexpected) conclusions. Likewise, questions centered with quality are targeted at measuring the performance of the learning system when immersed in environments riddled with imperfect or conflicting data — can the system focus its attention upon the pertinent information and avoid the temptation to incorporate irrelevant facts? The practicality and flexibility of a learning scheme must also tested — is it possible to generalize the methods to suit a peculiar task domain? The "Old Woman in the Shoe" phenomena is certainly descriptive of machine learning research, and the solution is to strive to bring the more promising of these algorithmic children to maturity so that their potential can be ascertained.

4. Strategies of Machine Learning

The task of a learning system is to transform information provided by a teacher into some new form that can be used by the system to learn. This transformation is the inference that the system performs. The teacher can be anything that provides the system with information (ex. human, environment, text input device, etc.). The increasing complexity of a system's learning capabilities is related to an increasing effort on the part of the student and a correspondingly decreasing effort on the part of the teacher. Thus, the more "intelligent" a learning system, the more responsibility it has to make the transformation mentioned above. In other words, the less dependent a system is on the teacher, the more inference the system must perform.

Learning systems are classified according to the amount of inference the system performs. When inductive strategies are employed, another consideration is the amount of background knowledge needed. Since induction is the process of generalizing and making assumptions, a certain amount of background knowledge is necessary to perform this task. The learning strategies discussed are (in order of inference performed): rote learning, learning by instruction, learning by deduction, learning by analogy, learning by induction. Learning by induction decomposes into two other strategies: learning from examples, and learning by observation and discovery.

4.1. Rote Learning

Rote learning is directly accepting and memorizing knowledge (information) from a teacher. The teacher in this case is any highly organized source that can supply information that the system can immediately accept into its memory (i.e. making essentially no transformation). An example of this type of system is a simple database update/query system which accepts facts and stores them for later processing. Another area of rote learning is called caching. Caching is storing answers to frequently occurring problems to avoid repeating expensive effort. While the main concern in rote learning is simply indexing the stored knowledge for later retrieval, systems performing caching might be thought of as using some inference techniques. Nonetheless, this learning strategy performs the least amount of inference of any of those mentioned in this document.

4.2. Learning by Instruction

Learning by instruction, or learning by being told, is acquiring knowledge from a teacher (again some organized source such as a human, textbook, etc.), requiring the learner to transform the knowledge into an internal representation, which is integrated with prior knowledge for effective use. While this strategy also involves basic memorization as in rote learning, it must transform the teacher's responses into the form of its understanding. An example of this strategy is an expert system that gives advice concerning bank loans to single persons. Given information about loans to married persons, it must incorporate this new knowledge with its prior knowledge to better its advice. Another example system is one which learns vocabulary from example sentences by generating its own sentences containing words in question from the input sentence, and checks them with a human to see if these words were used correctly. The user's response (yes or no) would be "transformed" into the system's representation by updating the dictionary depending upon the response. In this strategy, while the learner uses some inference, the burden of learning lies with the teacher to present and organize the knowledge in a way that can easily be augmented to the rest of the existing knowledge.
4.3. Learning by Deduction

Learning by deduction is drawing deductive, truth-preserving inferences from given knowledge and storing those conclusions that are useful for later processing. The given knowledge is any knowledge previously entered or provided by a teacher. These deductions follow simple conventional logic. An example of such a system would be an expert system with the capability to draw deductive conclusions from its known facts to generate other facts (ex — *any spouse of a professor lives in West Lafayette; x is married to professor y; new fact: x lives in West Lafayette*). Another type of learning by deduction is called *chunking*. In chunking, lower-level descriptions are grouped into higher ones that preserve the lower-level functions, but reflect the overall function of its subcomponents. For instance, suppose a system knows how to add two numbers, and that it also knows how to divide a number by two. By chunking these two functions, the system can "learn" that it knows how to compute the average of two numbers. In learning by deduction, the teacher does not play as major a role as in the previous two strategies, rather the burden lies in the bias mechanism used to determine which items should be deduced (i.e. the criteria that says which conclusions are worth making and keeping). Thus some amount of inference is performed in this aspect, as well as in chunking.

4.4. Learning by Analogy

Learning by analogy is matching descriptions from different domains to determine a common substructure, which then serves as the basis for analogical mapping. No teacher is really necessary here, rather it must be the system's responsibility to perform the analogy. A system that performs analogies might transfer the properties of a "solar system domain" (*sun attracts planet, sun is more massive than planet, planet revolves around sun*) to that of an "atomic system domain" (*nucleus attracts electron, nucleus is more massive than electron, electron revolves around nucleus*). Hence the system might "learn" about the atomic subworld by making use of what it knows about the solar system subworld. This type of learning is both deductive and inductive combined. Finding a common substructure involves inductive inference, whereas performing the analogical mapping is deductive in nature. The burden of the learning in this case is the amount of *a priori* knowledge (background knowledge) needed, and how it is used. The main drawback to this learning is that analogies are, almost by definition, imperfect correspondences between situations. That is, they rarely match perfectly from one domain to another (hence the need for the analogy). So, a system must have the capability to check its results for inaccurate mappings. A fair amount of inference must be performed, since a large amount of transformations must take place — not from teacher to learner, but rather from one domain to another. Making generalizations of the two domains that determine which properties are shared also requires that a very large amount of background knowledge be present.

4.5. Learning by Induction

Learning by induction is reasoning that starts with assertions (facts and observations) supplied by a teacher (human, environment, etc.) and concludes with more general, plausible assertions which still explain the initial assertions. It is in induction where the greatest level of inference is performed. Generalization is the key aspect of induction, and in order to generalize one must make creative inference. This area of learning decomposes into two major strategies of learning: Learning from Examples, and Learning by Observation and Discovery.

4.5.1. Learning from Examples

Learning from examples is determining a general description of a target concept which explains all the positive examples given, but excludes all negative examples given. There are two further decompositions of this strategy: instance-to-class generalization, and part-to-whole generalization.

4.5.1.1. Instance-to-Class Generalization

This is inducing a description of some class of objects given *independent* instances (examples) of the objects to be generalized. For example, given information about a fork, spoon, knife, hammer, wrench, and saw, a system might generalize a description of the first three called "eating utensils." This would adequately explain these examples, while effectively excluding the latter three. "Objects" in this
sense can be anything — from geometric shapes to organic cell types to problem solutions. A significant amount of inference must be done to form such a general description. A fair amount of background knowledge is needed as well in order to manipulate the objects. However, the initial example descriptions should themselves provide enough information to distinguish one object from another, and therefore aid in the generalization.

4.5.1.2. Part-to-Whole Generalization

This generalization is hypothesizing a general description of whole object (scene, situation, sequence) given selected parts of it. An example of this type of induction is a system which reconstructs a total view of room given select parts of it. Another example is, given a sequence of numbers, predicting the next number in the sequence, as well as determining a formula which derives the sequence. As in the instance-to-class generalizations, a significant amount of inference and background knowledge is used. However, this class of learning by example generally requires more inference since the generalizations are more broad. In part-to-whole the generalizations are more theoretical in nature, as opposed to the descriptive nature of the generalizations in instance-to-class.

4.5.2. Learning by Observation and Discovery

This learning is searching (independent of a teacher) for regularities and general rules explaining all or most observations. This is very much like the instance-to-class generalizations of learning by example. It is, however, more general than the previous strategy, as well as the fact that learning by observation and discovery must realize on its own which observations serve as "positive examples" and which ones are "negative." The main criterion classifying this strategy is the fact that no teacher is involved. While the system's environment has been included in the list of possible teachers for a system, in this case the environment may be present to provide the system with observations. However, the environment is only allowed to act and react to the system, it is not involved in any active teaching role.

This is the most complex form of learning since, by having no active teacher, the responsibility for learning falls completely on the system. This means the system must create its own classifications for generalizing. There are two types of this form of learning: that of passive observation (merely examining available examples, and that of active experimentation (system perturbs its environment to add to its observation space). Under these guidelines, the system must encounter regularities in the observations made, or else generate any further observations needed by perturbing its environment. An incredible amount of inference must be performed to make such generalizations; and these generalizations must be more "novel" than any of the generalizations made in other strategies. A large amount of background knowledge must also be employed to drive the inference process.

5. Learning Orientations

There are three interdependent orientations for machine learning:

1) Theoretical Analysis of General Learning Systems
2) Computational Models of Human Learning
3) Task Oriented Learning Systems for Specific Applications

The first orientation deals with developing standard theories for learning and implementing these theories in algorithms. There is no restriction to the type of algorithm developed. However, some researchers feel that the representations used and generated should be similar to those a human might use, since humans have to understand the output. The second orientation is concerned with developing a cognitive model for implementation to solve a broad spectrum of problems. The target of this research is developing experimental models of human learning. The third orientation is concerned with developing practical learning systems for specific tasks. Engineering systems are developed to perform these tasks. Much of the effort in developing these systems actually goes toward solving issues other than learning; that is, issues specific to the problem only. Any solutions to such specific problems are usually generalized to include a broader class of machine learning problems.
Experiments with a Pleasure-seeking Automaton

J.E. Doran
Department of Machine Intelligence and Perception
University of Edinburgh

1. Introduction

Doran’s pleasure-seeking automaton was designed to demonstrate learning principles in classifying environments according to suitability (that is, semblance to a goal condition) subjectively determined from the automaton’s perspective. Specifically, when placed at any location in an unfamiliar room, the automaton seeks a point of greatest "desirability" (that is, the "nest"). Then, having been reinitiated at that same point in a later trial, the automaton is able to optimize its previous path to the nest. The ability to optimize paths to the nest in successive runs is regarded as an indication of "learning" a route.

2. Operation Scenario

Consider figure PA-1. A ten by ten grid is used to simulate the room’s objective environment. Walls are composed of letters of the alphabet, each of which is associated with a relative degree of desirability ("Z" being the least preferred and "A" being the nest). Desirability is measured as a function of euclidian distance from the nearest wall and of this wall type factor. The automaton may make one of four moves in order to advance towards the nest — STEP forward one square, turn 90 degrees to the LEFT or to the RIGHT, and to STAND still (taking no action at all).

Figure PA-2 shows a typical initial effort at finding the goal. The automaton may explore a relatively circuitous path, naively choosing locally desirable points. As can be seen from figure PA-3, successive trials yield a more direct route to the nest, indicating that learning has taken place.

3. Methodology

A five step process is used in Doran’s algorithm (see figure PA-4). At first, the automaton reads current status information from the objective environment — a procedure termed "Sampling". This data is returned in the form of a five-tuple containing the wall type ("A" through "Z"), the euclidian distance to the aforementioned wall, the last action taken, the degree to which this location resembles the nest (i.e. the "desirability"), and the system time. The desirability formula is a simple linear function which intersects the abscissa about midway through the alphabet at maximum distance and has no meaning outside the context of the simulation (figure PA-5).

From this state vector, a reduced state can be constructed using the first two components (the wall and the distance). Notice that the reduced state cannot uniquely identify a location in the grid — a fact which Doran uses to introduce uncertainty into his model. His claim is that the reduced state may serve detrimentally as a point of confusion (since the objective environment may return different state vectors for the "same location" — as perceived by the automaton) or as an aid to generalization (since the automaton may correctly behave in "new" situations — the new reduced state being indistinguishable from a previous state which causes the activation of a favorable plan).

Next the automata "Stores" the reduced state in a tree of consequence pairs (a duplet composed from the reduced state of the arrival location and the system time — see figure PA-6). From the root, the branch on which to descend is given by the wall type of the source reduced state, and from this child node the distance portion of the source reduced state selects the path to the next level. Finally, the action taken to arrive at the destination reduced state guides the traversal to the leaf nodes where the consequence pair is stored. Remember that there may be several consequence pairs stationed at a particular leaf node due to the reduced state ambiguity already mentioned. In this way, the automata is able to record a history of its past actions.
Having stored the consequences of the last action, the automata must choose the next action to take. The "Findact" selector illustrated in figure PA-7 is responsible for this task. The decision to carry out an existing plan ("Doplan"), to formulate a new plan ("Makeplan"), to abandon further seeking ("Sleep"), or to randomly pick an unexplored action ("Explore") is determined according to whether or not the reduced state has been previously visited, whether or not a plan exists for the reduced state, and whether or not the desirability of the reduced state is at the target value (that of the nest).

Whenever "Findact" chooses to formulate a new plan using "Makeplan" (figure PA-8), a lookahead tree is constructed from viable alternatives. These take the form of the relative desirability of reduced states reachable from the current location (the set of all these attainable reduced states forms a connected component of the graph of all visited reduced states for which the current location is a member — the so called "option graph"). Growth of the lookahead tree commences with the terminal (leaf) nodes which record the desirability of sleeping at the corresponding state versus making a new plan from that state. Non-terminal nodes represent a composite desirability based upon the desirability of each of the options proposed by their children including the suitability of "exploring" or "sleeping" at that point.

Desirability factors for each of these actions is computed as shown in figure PA-8. Of particular interest is the inclusion of an "EXPVAL" variable (valued within the range of 0 to 1) in the "explore" desirability formula which serves modify the behavior of the automata to act "impulsively" (EXPVAL → 1) or "cautiously" (EXPVAL → 0). Doran believes caution to be a favorable characteristic and has weighted the "makeplan" desirability to supercede exploration whenever the two options "neighbor" (within a certain proximity) each other in the tree.

Once a particular action is selected, it is passed to the objective environment (by the "Act" procedure) which in turn delivers a new state vector to the automata for "sampling".

4. Critique

This late 1960's algorithm favorably demonstrates the contribution of heuristic tree search and pruning techniques to machine learning. Such techniques are useful to improve algorithm performance over a number of runs — which could loosely be described as a "learning" behavior. In fact, however, improvement over time is only a characteristic of learning. True learning incorporates a number of more complex processes, such as the capability to generalize, the capability to draw information from examples and analogies, and the capability to observe and make relevant discoveries about the environment and about the system's own operation. Such learning aspects require much more than heuristics alone can offer.

Certain additional criticisms arise in overview of the system. First, the assignation of desirability factors to walls should be questioned. Although this provision has no real bearing on the heuristic procedures that form the basis for the program, they markedly detract from environmental realism which does affect the quality of any observational and experimental attempts made. Second, these provisions and the simplistic approach to learning used seriously limit the degree to which the algorithm may be generalized to fit other contexts and limit the amount of insight into machine learning that can be drawn from this approach. Third, the algorithm is unable to make observations about its own behavior. For instance, the "STAND" action should eventually be eliminated from consideration by "Explore" or "Makeplan" since it has no effect on advancement to the nest. Finally, it is questionable whether static desirability forms a valid constraint upon the problem. Few forms of desire remain unchanged with the passage of time. Of more interest are algorithms that can address the complexities involved with dynamic (satisfiable) desires (like hunger). Such programs would have to reflect the fluidity of the subjective environment in their data structures and in their methods.
Automaton in Action

Automaton (⊗) is placed in 10 x 10 grid which simulates the room environment.

Automaton may take one of four actions:
- **STEP**: to move forward one square
- **LEFT**: to turn 90 degrees to the left
- **RIGHT**: to turn 90 degrees to the right
- **STAND**: to remain still

Walls (composed of letters) have desirability value (a = highest, j = lowest).

Automaton seeks point of greatest desirability (the next letter "a").

---

**Figure PA-1**

---

Automaton in Action

- Automaton makes circuitous route to nest:
  - moving from places of lower desirability to points of higher desirability
- Desirability is a function of distance to a particular wall and the "value" of that wall

---

**Figure PA-2**

---
Automaton in Action

- On successive trials, automaton makes more direct route to nest.
- It has thus "learned" the optimal path to the nest when started at the same initial grid point.

Figure PA-3

Methodology

Figure PA-4
Methodology

SAMPLE

- Read a state vector from the OBJECTIVE environment:
  \[ \langle \text{WALL}, \text{DISTANCE}, \text{LAST-ACTION}, \text{DESIRE}, \text{TIME} \rangle \]
  
  \[ \text{WALL} \in \{ A (\# 1), B (\# 2), \ldots, Z (\# 26) \} \]
  
  \[ \text{DISTANCE} = \text{Euclidian Distance to WALL} \]
  
  \[ \text{LAST-ACTION} \in \{ \text{STEP, LEFT, RIGHT, STAND} \} \]
  
  \[ \text{DESIRE} = 50 - (\text{DISTANCE} \times 3 + \text{WALL}) \]
  
  \[ \text{TIME} = \text{System Time} \]

STORE

- Update consequence tree (representation of experienced environment)
- Leaf nodes are consequence pairs (time, reduced state)
- Example: At time 0, STEP from C3 to C2

---

Page 10
Pleasure-seeker

Methodology

1. **FINDACT**
   - Decide to **DOPLAN / MAKEPLAN / SLEEP / EXPLORE**

   - Reduced State Previously Encountered?
     - Yes: Plan Set for This State?
       - Yes: **DOPLAN**
       - No: **MAKEPLAN**
     - No: **SLEEP**

   - Desirability At Target?
     - Yes: **DOPLAN**
     - No: **MAKEPLAN**

   - Figure PA-7

Pleasure-seeker

Methodology

2. **MAKEPLAN**
   - Forms **LOOKHEAD TREE** from Option Graph (connected component of environment graph containing state S)
   - Grown up from leaf (terminal) nodes -- set of all (reachable) nodes in option graph
   - Estimates desirability of nodes, greatest valued option to parent

   - Non-Terminals
     - SL = Desirability of Node Unchanged
     - EX = D(S) + ( Target - D(S) ) * EXPVAL
     - MP = Heavily weighted EX to produce "caution"

   - Figure PA-8
1. Introduction

PROSE is a program which is capable of adaptively improving its linguistic behavior by conversation with a human being. Namely, it is able to expand its vocabulary by inferring syntactically correct word usages as it assesses the results of a battery of test sentences. These sentences exercise words whose usage is uncertain, and each trial elicits response from a human tutor who either reinforces the valid word usages or cues the system to correct improper usages.

2. Operation Scenario

Consider figure PR-1. Given the input sentence, "The computer and the program in its store become an integrated whole which can perform marvelous feats", PROSE constructs a dictionary during its first pass (this is actually the task of "SPUD" — the Sentence Parser Using Dependency division of PROSE). Having lexically classified the words, SPUD is able to construct the dependency tree of figure PR-2. A few observations should obviate the semantics of this arrangement. Here, it can be seen that the sentence structure hinges upon the verb, "becomes" and that determiners (eg. "the", "an", etc.), prepositional phrases (eg. "in its store"), and adjectives (eg. "integrated", "marvelous", etc.) depend upon the nouns they modify.

After SPUD has completed its duties, it passes the dependency tree to the "GASP" (Grammatically Analyzed Sentence Producer) which uses the dependency tree and the dictionary (described below) to produce random sentences testing words for which usage has not yet been determined. In figure PR-3, the words "marvelous", "feats", "integrated", and "whole" are under scrutiny. For each sentence GASP generates, the human tutor is obliged to either reinforce the usage or to direct the system in its attempts to arrive at a correct classification for the word. These cues may take one of two forms.

The first of these is an "ISA" response which causes PROSE to construct a new classification for the ISA subject with the label given by the ISA object. All the attributes accumulated by the ISA subject during the GASP session are transferred to this new lexical category. For instance, the ISA response, "sat ISA past-participle", causes a new lexical class "past-participle" to be formed in the dictionary with all the attributes currently ascribed to "sat". In the event that the ISA object class already exists, the attributes of this category are updated by the addition of the tags adhering to the ISA subject.

The second cue is an "ISP" response which causes PROSE to tag the ISP subject with attributes of the class indicated by the ISP object. This is the reverse of the procedure triggered by ISA. The ISP response "whole ISP abstract" shown in the example session of figure PR-3 would cause the characteristics stored for the noun class "abstract" in the dictionary to be attached to the word "whole", thus narrowing the scope of its usage.

Once the GASP session has reached completion, the "FINISHER" portion of PROSE is invoked to compile the results of the testing and place the word into the dictionary under the suitable classification. The authors claim that the FINISHER exhibits "learning" behavior as it assesses the content of the GASP session and ascertains the correct lexical category for the words in question.
3. Methodology

As can be concluded from figure PR-4, PROSE is a composition of distinct modules (SPUD, GASP, and the FINISHER) which are integrated around a central data structure, the dictionary. SPUD receives control initially to parse the words of the input sentence into general lexical categories and consequently to construct a dependency tree. GASP is then invoked to generate a battery of sentences to test words whose usage is not sufficiently constrained by the dictionary.

This usage dictionary is the core of PROSE. It is decomposed into a hierarchy of sub-dictionaries. The properties of classes located at the leaf nodes of the tree propagate to the ancestral nodes in the following manner (refer to figure PR-5). Each parent (non-terminal) node is associated with a combinatorial logic function which determines how the attributes of the child classifications are to be joined. If the parent node is an XOR node, the attributes of the classes for the subtrees are disjoint and no meaningful generalization can be made. If the parent node is an AND node, the attributes of the classes for the subtrees are part of a broader class represented by the parent. In this case the characteristics of the sub-tree dictionaries can be generalized to describe the parent classification.

The process used by the FINISHER to place a word within the the usage dictionary described is given in figure PR-6. Whenever SPUD encounters a new word in the input sentence, it temporarily assigns the word a position within the dictionary hierarchy. The word then migrates (under the guidance of the FINISHER) toward the root (taking a more general usage) or toward the leaves (taking a more specific usage) accumulating attributes (tags) of the classes it encounters according to the XOR or AND directives. If the test sentence for a word is rejected by the human tutor the dictionary entry for the word moves toward the leaves, effectively constraining its usage. If the test sentence for a word is accepted by the human tutor, the dictionary entry for the word climbs toward the root, which serves to generalize its usage. This process repeats as GASP generates another test sentence from the latest placement.

ISA and ISP responses (described earlier) are shortcuts to this process, and are necessary since the FINISHER may be unable to locate a suitable resting place for the word's entry.

4. Critique

PROSE's strength lies in its ability to demonstrate smooth integration of several modules (SPUD, GASP, and the FINISHER) to accomplish a task. When individual programs are linked together, the groundwork laid in prior research can be extended. The strengths and weaknesses of constituent modules (which may, as in the case of PROSE, be modifications of existing programs) and the scope of their purpose can be ascertained. Current machine learning research efforts have compromised exploratory depth earned through integration to become content with highly individualistic and shallow trends. The lateral expanse of learning research needs to be exchanged for the vertical growth afforded by the integration and further scrutiny of existing models.

PROSE's greatest shortcoming can be found in its implicit definition of learning as an accumulation of class attributes. The analytical facet of machine learning is largely absent. PROSE is incapable of defining distinct usage categories without prompting from a human tutor (i.e. the ISA response). It is also incapable of detecting when the direction of its questioning (the GASP session) has taken a fruitless fork. In these situations, the ISP response is necessary to provide the system with discriminant information. Without the autonomous acquisition of facts presented externally in the form of ISA and ISP cues, PROSE deserves little more than the title of "electronic pack rat".
"The computer and the program in its store become an integrated whole which can perform marvellous feats."

First Pass: (from SPUD)
- The noun determiner: Store
- Substrate noun: Computer
- Substrate noun: Program
- Integrated verb or adjective: which
- Whole noun: Whole
- Pl. noun or smg. verb: Feats

Second Pass: (from SPUD)
- Becomes
- Computer and program
- The in an integrated store
- Its
- Which perform can feats marvellous
PROSE in Action

Output: (from GASP)

Words in double are "marvellous", "feats", "integrated" and "whole"

The feat is performed. Right
The feat is marvellous. Right
The feat is. Feat ISP abstract.
The whole becomes the computer. Whole ISP abstract.
The whole is integrated. Right.
The computer and the program perform. Right.
The whole computer performed the feat. Right.
The whole is marvellous. Right.

Methodology

SPUD \rightarrow GASP \rightarrow Example

Dictionary

\rightarrow FINISHER

\rightarrow Response

Figure PR-3

Page 15
Methodology

Dictionary

- Composed from a hierarchy of dictionaries, general to specific
- Class properties propagate from leaves to root
  - XOR: Subtree properties are mutually exclusive
  - AND: Properties of this class are applicable to subtrees

```
Verb
  XOR
  Noun
  AND
  Concrete
  Abstract
```

Figure PR-5

Methodology

Dictionary

- Words are placed into tentative dictionaries by SPUD
- GASP forms test sentences from these classifications
- If REJECTED...
  - Move word down tree
  - Inherit parent class properties
  - Create new class if needed
    - (Serves to constrain usage)
- If ACCEPTED...
  - Move word up tree
  - (Serves to generalize usage)

```
Verb
  XOR
  Noun
  AND
  Concrete
  Abstract
```

Figure PR-6
Explanatory Schema Acquisition
Gerald DeJong
University of Illinois, Urbana-Champaign

1. Introduction

Explanatory Schema Acquisition (ESA) is a system which learns by observation and discovery. Its goal is to recognize significant events, those that are relevant to understanding later events (i.e. part of the planning of another event), and then generalize the new events into a new concept. A natural language system which acquires new schemata has been implemented. A schema can be best thought of as pre-stored solutions to problems. In the context of a natural language story, these schemata can best be thought of as predefined situations that are part of the story's plot. So, when presented with a new story that exhibits a problem-solving behavior (i.e. has a logically sequential plot with motives, reactions, etc.) the system generalizes the situations of the story and stores the generalization to aid in processing later stories.

2. Methodology

The goal of ESA is to recognize significant events and generalize them. The significance of an event is realized by use of the background knowledge in the system. This reflects DeJong's opinion that human adult learning is largely explanation driven. Included in the background knowledge are known schemata and knowledge specific to the problem domain. Generalizing is basically a matter of grouping known and/or acquired schemata into a new schemata. In the context of problem solving, this is represented in figures ESA-la & ESA-1b. Figure ESA-la demonstrates applications of a series of operators (op1, op2, and op3) to move from an initial problem state \( I \) to a goal state \( G \). Figure ESA-1b shows this same problem being solved using a schema generalized from the operators.

DeJong classifies four basic generalizations to be performed on schemata. They are schema composition, secondary effect elevation, schema alteration, and volitionalization. These techniques are used to create new schemata from known schemata.

2.1. Schema Composition

Schemata have certain preconditions which must be met in order to "trigger" them. Schema composition involves an essentially unchanged schema, whose precondition is met in a novel way. The basic notion of the precondition is the same, however, which still allows the particular schema to be triggered. Nonetheless, the manner in which the precondition is satisfied is different than the original precondition of the schema. Figure ESA-2a shows an example of an original schema, Bargain. It states that if both \( X \) and \( Y \) have something the other wants, then \( X \) and \( Y \) can Bargain. Figure ESA-2b shows the secondary effect of dating, which is to make someone jealous. Thus, Date can now be used with other schemata to create new schemata from known schemata.

2.2. Secondary Effect Elevation

Schemata are used in conjunction with each other to satisfy conditions that describe new schemata. Secondary effect elevation is using an existing schema in a new way which brings out an otherwise secondary effect. Certainly it can be suggested that one should just let all effects of a schema be of equal value. However, this approach is impractical computationally since every schema has many secondary effects. Hence, it is valuable to single out a primary effect and use the the technique of secondary effect elevation to make use of the non-primary effects. Figure ESA-3a shows a primary use of the schema Date. \( X \) dates \( Y \) because \( X \) wants \( Y \)'s companionship. However, figure ESA-3b shows the secondary effect of dating, which is to make someone jealous. Thus, Date can now be used with other schemata to
build upon the concept of jealousy.

2.3. Schema Alteration

When the system develops a schema, it is rarely the perfect "fit" for the concept being established. Hence, schema alteration is necessary to modify or refine an existing schema to better fit the requirements of a new situation. This is usually a matter of improving the initial generalization that created the schema through observing situations in which the schema is used. Figure ESA-4a demonstrates an initial configuration of the schema Farming. Figure ESA-4b shows this same schema modified to better fit the "real-world" model (which is acquired through the background knowledge).

2.4. Volitionalization

This technique is used to transform a schema which normally occurs without being planned to one which has an active agent (i.e. a "planner"). This is similar, in a way, to secondary effect elevation in that the primary effect is the spontaneous nature of the schema, and the secondary effect being elevated is a volitional counterpart to this. Figures ESA-5a and ESA-5b demonstrate this technique with the schema of Car-Accident. What is normally a spontaneous event (figure ESA-5a), becomes a planned event (i.e. one with an active agent who has a motive) in figure ESA-5b. Note the two new preconditions and the one new expected outcome and how they relate to the planner.

3. Operation Scenario

A sample of part of a "run" of the ESA system is given in figure ESA-6. The four input lines are the story to be generalized by the system. The ensuing output generated on input of the second line shows the internal representation once the input has been translated by the natural language front end. The word "Processing ---" denotes the start of a new input line. New schemata are triggered as certain predicates are recognized. This is shown by the schemata $Feed and $Naive-Poison. The poison schema is called "naive" to denote that it is an incomplete schema containing little more that what is needed for this story. The names beginning with "$" are the result of DeJong's re-writing the actual output, replacing the system's internal names with mnemonic symbols.

The processing continues until all four lines have been input. The system does basically as much processing as possible with each line input. When the processing for one line has been exhausted, it reads the next line. After input of the fourth line, the system has had enough schemata activated to trigger the volitionalization step. This creates a new schema, murdering to inherit, using the schemata of $Poison, $Death, $Possess, etc. along with a volitional actor Agrippina. A partial result is shown in figure 7.

4. Critique

While the most of the four taxonomies of schema generalization are good, the system must know so much about the schemata used that it basically has the final generalization preprogrammed into it. This is exemplified in the volitionalization example given (see figure ESA-6). The system seems to have "keyed" on the input "poisoned", since in New Schema S00003 the object is poison, not mushroom. Natural processing of the second input sentence would find "poisoned" modifying "mushrooms", making "mushrooms" the object. It is unclear how "poison" and "mushrooms" exchange syntactic roles. What if the modifier were not "poisoned" but rather "spoiled"? Certainly the effect might be the same (Agrippina knew spoiled mushrooms are toxic, etc.). However, now the system must volitionalize even though there is no way for Agrippina to have "spoiled" the mushrooms. Thus, it appears that the system already knows the very thing it is trying to learn as a planned event. Hence, it is the authors' opinion that this system performs no real induction. Other more secondary shortcomings of the system include the fact
that no natural language processing is employed in the system. Everything is done in terms of the internal representations of the system, which become so specific as to be impractical. Dejong even admits to some of the representations being ad hoc. Also, no later stories are shown to benefit from the system's results, and initial claim of the system's benefits.

What should be done is to generalize broader concepts. This would involve using the background knowledge in a less specific manner. In order to develop truly new schemata, one needs some aspect of genuine creativity coupled with more general knowledge. While this "real" generalization may seem too ambitious, anything less will result in a nominal system that performs no actual learning.
**ESA Schema Composition**

- Composing known schema in a novel way

Schema: Bargain

both $X$ & $Y$ have something
the other wants

precondition

Bargain

*Figure ESA-2a*

**ESA Schema Composition**

- Composing known schema in a novel way

Schema: Bargain

Fred has $$$

John takes Fred's wife

John has Fred's wife

Bargain $\equiv$ Extortion

**New Schema**

Note: This is done with no prior knowledge of "kidnapping" or "extortion"

*Figure ESA-2b*
**Secondary Effect Elevation**

- Using an existing schema in a new way to bring out an otherwise secondary effect

**Schema: Date**

- X wants Y's companionship
- X DATES Y

**Figure ESA-3a**

---

**Secondary Effect Elevation**

- Using an existing schema in a new way to bring out an otherwise secondary effect

**Schema: Date**

- Bob wants to make Sue jealous
- Bob DATES Mary

**an Alternate Use of Date**

**Figure ESA-3b**
Schema Alteration

- Modifying a nearly correct scheme to fit the requirements of a new situation (refining incomplete schema)

Schema: Farming

Figure ESA-4a

Figure ESA-4b
Volitionalization

- Transforming a schema which has no planner into one which can be used by a planner to attain a specific goal
  - like "Secondary Effect", with the primary effect being that the event occurs on its own, and secondary effect being that the event is planned

Schema: Car-Accident (unplanned event)

X loses brakes in car

(expected effects)

CAR-ACCIDENT (X crashes and dies)

Figure ESA-5a

Volitionalization

- Transforming a schema which has no planner into one which can be used by a planner to attain a specific goal

Schema: Car-Accident (unplanned event)

John wants to get rid of his wife, Mary

John cuts Mary's brake lines

Mary loses the brakes in her car

CAR-ACCIDENT

(expected events)

Mary dies

John's a widower & can marry his secretary

* used by planner to attain goal

Figure ESA-5b
ESA  

**ESA in Action**

**Input:**
1) Claudius owned an island estate.
2) Agrippina fed Claudius poisoned mushrooms.
3) Claudius died.
4) Agrippina inherited the island estate.

For instance, the output produced by the input of #2 is:

Processing --- ($Feed (Actor #Agrippina#) (object #poison# type #mushrooms#))

New Schema --- $00003 ($Feed (Actor #Agrippina#) (object #poison#) (to #Claudius#))

New Schema --- $00006 ($ naive-Poison (Actor #Agrippina#) (instrument #mushrooms#) (object #Claudius#)) activated by $00003

**Figure ESA-6**

**ESA**  

**ESA in Action**

**Final Output:**

New Schema U00035:

(vars (Benefactor (Var U00001)) (Prior-Heirs (Var U00003)) ...)

(Activate ($Individual-Inherit ...))

($Premeditated-Murder ...)

The system has learned a new schema of murdering to inherit something

**Figure ESA-7**
1. Introduction

In this work, Michalski offers a formal mathematical scheme of inductive processes that serves as the theoretical basis for the STAR learning methodology. Michalski formally defines the scope of induction with the following premise:

*Given a set exemplary instances (E) for a class (K), a tentative inductive assertion (possibly null), and a pool of background knowledge (stated as hypothesis preference criteria) obtain a complete and consistent description (H) for the class. H must satisfy the constraints imposed by the background knowledge.*

2. Operation Scenario

As a demonstration of the capabilities of the STAR methodology, consider the two cells depicted in figure SM-1. Within the context of this scenario, cells of the class ONC are presumed to be cancerous, while cells of the class DNN are thought to be normal. The object is to find descriptions which characterize the cells in each class as well as to find descriptions which discriminate cells of one class from cells of the other.

To begin, the user enters descriptions for each cell in the form of "C-expressions" which are conjunctions of annotated predicate calculus descriptors and selectors (explained later). A sample C-expression describing a DNC (cancer) cell might be the following:

\[
\exists \, \text{CELL}_1, B_1, B_2, \ldots, B_6 \left[ \text{contains} (\text{CELL}_1, B_1, \ldots, B_6) \right] \left[ \text{circ} (\text{CELL}_1) = 8 \right] \quad & \quad \left[ \text{plasm} (\text{CELL}_1) = A \right] \left[ \text{shape} (B_1) = \text{ellipse} \right] \left[ \text{texture} (B_1) = \text{stripes} \right] \quad & \\
\left[ \text{weight} (B_1) = 4 \right] \left[ \text{orient} (B_1) = \text{NW} \right] \left[ \text{shape} (B_2) = \text{circle} \right] \quad & \\
\left[ \text{contains} (B_2, B_3) \right] \left[ \text{texture} (B_2) = \text{blank} \right] \left[ \text{weight} (B_2) = 3 \right] \quad & \\
\left[ \text{shape} (B_3) = \text{circle} \right] \left[ \text{texture} (B_3) = \text{shaded} \right] \left[ \text{weight} (B_3) = 5 \right] \quad & \\
\therefore \left[ \text{class} = \text{DNC} \right]
\]

Rendering the above expression in English would yield the following description:

There exists a cell, \( \text{CELL}_1 \), which contains bodies \( B_1 \) through \( B_6 \). The number of segments in the perimeter of \( \text{CELL}_1 \) is 8, and it is composed of type "A" protoplasm. Body \( B_1 \) is a striped ellipse pointing northwest with weight 4. Body \( B_2 \) is blank with weight 3, and it houses a second body \( B_3 \). The C-expression continues on in this vein until the description for \( B_6 \), a shaded circle of weight 5, is reached.

Finally, the "links to" operator, \( \therefore \), informs the system that \( \text{CELL}_1 \) belongs to the class of cancerous cells (DNC).

In addition to providing C-expression class member descriptions, the user may also supply observational formula which the system will use to complement these descriptions. For example, if the measurement of cell weight would be an advantageous observation for the system to perform on the cells of each class, the user would enter the cell weight formula:

\[
\text{weight}(\text{CELL}_1) = \sum_i \text{weight}(B_i)
\]

Finally, the user constrains the number of descriptions generated by specifying the desired form of the hypothesis. This last is expressed in terms of a "LEF" (Lexicographic Evaluation Function — described
further on) and is considered a type of background knowledge. An exemplary hypothesis preference criterion would be:

Minimize the complexity of each description, and maximize the number of positive facts (examples for the class) covered.

After the necessary information has been input, the STAR methodology produces two types of descriptions for each class. The first of these types of descriptions discriminates members of one class from members of all other classes. A few instances of discriminant descriptions for cells of the class DNC could be:

\[ \text{circ} = \text{even} \]  
\[ \exists (1)B \ [\text{texture}(B) = \text{shaded}] \ [\text{weight}(B) \geq 3] \]  
\[ \exists (\geq 1)B \ [\text{shape}(B) = \text{boat}] \ [\text{orient}(B) = \text{N} \lor \text{NE}] \]  

When translated into English, equation 1a states that one distinguishing attribute of cancer cells (class DNC) is that their perimeters have an even number of segments. Equation 1b gives another unique feature of cancer cells — that each cell of the class has one shaded body of weight 3. The discriminant description of equation 1c says that cancer cells also differ from normal cells in that each one contains at least one boat shaped body which is pointed in either a north or north-easterly direction.

The second type of description, characteristic descriptions, point out the factors that members within a certain class share. Characteristic descriptions are not necessarily discriminant, since other class members may also have these attributes. A few instances of characteristic descriptions might be:

\[ \exists (1)B \ [\text{weight}(B) = 5] \]  
\[ [\text{plasm} = A \lor D] \]  
\[ \exists (2)B \ [\text{shape}(B) = \text{circle}] [\text{texture}(B) = \text{solid-black}] \]  

Equation 2a states that one common attribute among cancer (class DNC) cells is the existence of a body with weight 5. Equation 2b lists another shared characteristic — that cancer cells are composed of type "A" or type "D" protoplasm. The last equation (2c) shows that class DNC cells contain 2 solid black circular bodies.

3. Methodology

In order to pave the way for his STAR methodology, Michalski formally develops several key concepts which serve to excavate the basis of induction as well as to crystallize the contributions of background knowledge to the development of the final hypothesis. A condensation of these concepts is provided below, accompanied by comments where thought to be helpful.

To begin, it is necessary to define the sets which actively participate in the theory.

1. Let \( F \) be the set of all training instances (i.e. examples, initial class member descriptions, or facts). Also define \( K \) to be the set of all classes for which descriptions must be found.

2. Let \( E_k \subseteq F \) be the set of training instances for a particular class \( K_k \). Likewise, define \( D_k \) to be the set of descriptions covering the examples \( E_k \) for the class \( K_k \).

3. Let \( e_{k,i} \subseteq E_k \) be a single training instance (example, initial class member description, or fact).

4. \( F \) is therefore the union of all class training instance sets, and each fact in a certain class is exemplary of that class. Formally:

\[
F: \bigcup_k E_k \Rightarrow F: \{ e_{k,i} \Rightarrow K_k \}, i \in | E_k |
\]

5. \( H \) the final inductive hypothesis, is a disjunction of all class descriptions \( D_k \).
Set definitions having thus been laid in place, Michalski is able to formally describe the concepts of *completeness* and *consistency*. Completeness is a condition in which all instances of a particular class support the description of that class:

\[ \forall k, (E_k \Rightarrow D_k) \]

The condition of consistency arises when the description for a certain class refutes all examples of other classes:

\[ \forall k, l (D_k \Rightarrow \overline{E_l}), \text{ if } k \neq l \]

3.1. Annotated Predicate Calculus

The expression language used throughout the STAR methodology is an annotated version of the predicate calculus, termed "APC" (Annotated Predicate Calculus). APC elements are loosely divided into two categories — selectors which are terms in relational expressions (<, >, =, ≤, ≥, etc.) and descriptors. These last are further partitioned into three functional types:

- **Nominal** Also called *categorical* descriptors, these predicates have no structure. They simply associate a domain name with the parameters they govern. For example:
  
  - `bloodtype(person) = AB+
  - `color(Barchetta) = red

- **Linear** Linear (or ordinal) descriptors map parameters from a totally ordered set (i.e., variables measured on ordinal, interval, ratio, and absolute scales) onto another ordered set. For instance:
  
  - `distance(P1, P2) = 0.65
  - `avetemp(May) = 68° F

- **Structural** Structural descriptors convey a hierarchical relationship between the predicates and their arguments. Such dependencies may be viewed pictorially in the form of a *tree oriented generalization hierarchy graph*. Examples can be seen in figure SM-2. Note that descriptors themselves may be arranged according to a hierarchy, as is the case with the linear spatial dimension descriptors of figure SM-2.

APC basic assertions are written as conjunctions of these selectors or descriptors, called "C-expressions" (for Conjunctive expressions). A sample C-expression is given below with the descriptor and selector types tagged for clarity:

\[
\exists P_0, P_1, P_2, P_3 \left( \begin{array}{l}
\text{contains}(P_0, P_1, P_2, P_3) \\
\text{ontop}(P_1 \& P_2, P_3) \\
\text{length}(P_1) = 3 \ldots 5 \\
\text{weight}(P_1) > \text{weight}(P_2) \\
\text{color}(P_1) = \text{red} \lor \text{blue} \\
\text{shape}(P_1 \& P_2 \& P_3 = \text{box})
\end{array} \right)
\]

An English rendition of the above C-expression would be:

An object \( P_0 \) contains parts \( P_1, P_2, \) and \( P_3 \). The first two parts, \( P_1 \) and \( P_2 \), lie on top of the third \( P_3 \). The length of the first part is between 3 and 5, and its weight is greater than the weight of the second. The color of the first part is either red or blue, and all three parts are box-shaped.
3.2. Lexicographic Evaluation Function

The Lexicographic Evaluation Function (or "LEF") provides a means to specify preference criteria for the form of descriptions (hypotheses) generated. The LEF is needed because for any $F, |H|$, is potentially infinite.

Focusing hypotheses is the primary contribution of background knowledge to the induction process. As was stated in the historical discussion of machine learning, the "Knowledge-Rich" paradigm prevalent at present is centrally concerned with the role of background knowledge in learning processes. Without a significant amount of initial knowledge, learning systems are incapable of producing novel results or (in the case of inductive learning systems) directing effort along profitable channels.

Michalski's STAR methodology is able to capture the two forms of background knowledge (explicit and implicit) as well as to clearly illustrate the subtle means by which they mold hypotheses. Implicit background knowledge is that which is inherent in the problem language. Incomplete languages (those incapable of describing all features of phenomena) naturally restrict the number and form of generalizations made, since the expressive limitations of the language used serve to shape (or bias) the outcome. Michalski claims that the annotations he proposes to the predicate calculus are representative of implicit background knowledge.

Explicit background knowledge is that which is stated literally to constrain the generalization process. Michalski's LEF is a prime example of this sort of background knowledge. Specifically, a LEF is a list of hypothesis preference criteria and associated match tolerances:

$$\text{LEF}: \langle C_1, T_1 \rangle, \langle C_2, T_2 \rangle, \cdots$$

where each tolerance $T$ is defined over the range $[0 .. 100\%]$. A LEF progressively filters the description list it receives in the following manner. Each description is tested against the first criterion, and only those which satisfy the criterion to within the associated tolerance are permitted to remain. The surviving descriptions are passed to the second criterion to be tested to within the second tolerance, and so forth. In this way, the descriptions which disjunctively combine to form the hypothesis can be tailored.

3.3. Generalization Rules

The actual inductive steps performed by the STAR methodology are applications of generalization rules. When supplied with descriptors, these rules produce additional more globally relevant predicates. Note that induction is falsity preserving (in contrast to deduction which is truth preserving). Formally:

$$F \triangleright H$$

That is, the examples or facts generalize to form the hypothesis.

Generalization rules fall into two camps depending upon the type of descriptors for which they are defined. Selective generalization rules are designed for use with class member descriptions present initially, as opposed to constructive generalization rules which are applied to derived descriptions.

A few examples of selective generalization rules follow to give the reader a taste of the core of Michalski’s theory. In the instances provided, CTX refers to a "context" — a list of assertions which has been accumulated, and $S$ denotes any arbitrary expression.

1. **Dropping Condition Rule:** This rule allows $S$ to be removed from its context CTX. Logically speaking, if CTX is false, then CTX AND $S$ is certainly false.

   $$\text{CTX} \& S \implies K \mid \text{CTX} \implies K$$

2. **Adding Alternative Rule:** Two contexts may be combined disjunctively to describe a class $K$ if one of them presently covers $K$. Logically, if the disjunction is false, then either component context is also false.

   $$\text{CTX}_1 \implies K \mid \text{CTX}_1 \lor \text{CTX}_2 \implies K$$
3. **Extended Reference Rule:** Nominal descriptors whose values are taken from a subset \( R_1 \) of domain \( L \) may be generalized to a larger subset \( R_2 \) of that same domain \( L \). Formally:

\[
\text{CTX} \& [L = R_1] \Rightarrow K \land \text{CTX} \& [L = R_2] \Rightarrow K
\]

where:

- \( L = \{\text{nominal descriptor set}\} \)
- \( \text{DOM}(L) = \text{domain of } L \)
- \( R_1 \subset R_2 \subset \text{DOM}(L) \)

4. **Closing Interval Rule:** A context restricted independently by linear points "a" and "b" may be relaxed to conform to the interval \([a .. b]\) as long as \( a \leq b \). The formalization for this rule is given in figure SM-3.

5. **Climbing Generalization Tree Rule:** A context restricted over each of the child cases \([a, b, c, \ldots, i]\) of a parent structural description, \( s \), may be generalized to be governed by that parent description. The formal climbing rule is depicted in figure SM-3.

6. **Conjunction to Disjunction Exchange:** The power available in generalization can be seen in this rule. Falsity preservation allows a conjunction of facts \( F_1 \) and \( F_2 \) to be turned into a disjunction since, if \( F_1 \lor F_2 \) is false certainly \( F_1 \land F_2 \) is false. Formally:

\[
F_1 \land F_2 \Rightarrow K \land F_1 \lor F_2 \Rightarrow K
\]

An example of a constructive generalization rule can be seen in figure SM-4. If a context is bound by a set of facts \( F_1 \) and those facts imply a second more general set of facts \( F_2 \), then the context may be generalized to the binding constraints of the second set.

### 3.4. Simple Observations

The initial set of class instance descriptions may be augmented by the inclusion of simple observations upon these facts. Such observations are provided by the user in the form of APC formulae and may encompass any measurable property of those facts. Some examples include:

- A count of the number of elements in an assertion
- Statistical calculations (e.g. mean, median, max, min, etc.)
- Monotonicity relationships

### 3.5. The STAR Algorithm

So far, the gist of Michalski's induction theory has been stated. The actual STAR algorithm based upon these premises is now the subject of scrutiny. First off, Michalski defines the operator, STAR in the following manner:

1. Partition the set \( F \) of all training instances into subsets \( POS \subset F \) and \( NEG \subset F \) which are composed of all strictly positive (supporting) and strictly negative (conflicting) instances, respectively.

2. Given \( e \in POS \) define \( G(e \mid NEG) \), the STAR of \( e \) against the set \( NEG \), to be the set of all maximally general complete and consistent C-expressions satisfied by \( e \) (i.e. complete) that do not satisfy any events in \( NEG \) (i.e. consistent).

3. Further define a bounded STAR, \( G(e \mid NEG, m) \), of \( e \) against the set \( NEG \) to be a number of no more than \( m \) maximally general complete and consistent C-expressions. That is:

\[
G(e \mid NEG, m) \rightarrow D_e, \text{ such that } |D_e| \leq m
\]

The restraint of \(|D_e| \leq m\) is accomplished by the use of a suitable LEF.
Hence, the scope of obtaining complete descriptions for all positive instances \((oppA e \in POS)\) is limited to finding complete and consistent descriptions for a single positive example, \(e\). The final maximally general class description, \(H\), of all \(e \in POS\) is a disjunction of all \(G(e \mid NEG, m)\) for each \(e \in POS\). Formally:

\[
G(e \mid NEG, m) \rightarrow D_e \subset D_k \subset H, \ e \in POS \subset F \ \\
\forall D_e, \ * e \in POS \rightarrow H
\]

The general flow of the inductive procedure using bounded STAR's progresses as follows:

1. Randomly select \(e \in POS\)
2. Generate \(G(e \mid NEG, m)\) using the "INDUCE" method (summarized later).
3. In the bounded STAR derived in step 2, find \(D_e\) with the highest LEF preference.
4. If \(D_e\) covers \(POS\) completely, proceed with step 6.
5. Reduce \(POS\) to contain only those \(e \in POS\) which are unsatisfied by \(D_e\). Repeat the process with step 1.
6. Form \(H\) from a disjunction of all \(D_e\).

Finally, the steps involved in the "INDUCE" method to generate a bounded STAR are listed below:

A. Dissect the C-expressions of \(e\) using the dropping condition rule. This produces a partial STAR, PS, since some elements of the resulting selector list will cover \(NEG\).
B. Order these selectors using \(LEF_1\):

\[
LEF_1; < (-negcov, T_n), (poscov, T_p) >
\]

\(LEF_1\) serves to minimize the number of negative elements covered (by maximizing -\(negcov\)) and to maximize the number of positive instances covered (by maximizing \(poscov\)).
C. Expand the list \(PS\) using constructive generalization rules, simple observation formulae, and heuristics defined by the background knowledge.
D. Use \(LEF_1\) again to constrain \(|H| \leq m\).
E. Test the resulting \(PS\) for consistency and completeness. In other words, assure that \(negcov\) (the number of negative instances covered by \(PS\)) is 0, and that \(poscov\) (the number of positive instances covered) is \(|POS|\), respectively.
F. Append selectors in \(PS\) to specialize them (forming an implicit conjunction). Return to step E until \(PS\) is complete and consistent.
G. Apply extension, close interval, or climbing generalization tree rules to generalize the resulting expressions.
H. Apply the user-provided explicit background knowledge LEF to rank the expressions according to description form preferences. An example preference LEF would maximize the number of events covered in \(POS\) and minimize expression complexity (as determined by a count of the descriptors and selectors in the final assertion).

4. Critique

Michalski’s STAR methodology has enormous merit in formalizing the pertinent issues of machine induction, giving substance to several vaporous aspects:

- It discloses the contributions of explicit and implicit background knowledge.
- It defines and functionally catalogs the various generalization rules.
- It captures the interplay of \(POSitive\) and \(NEGative\) training examples.
• It reduces the task of forming a maximally general class description to finding hypotheses for bounded STAR's.

• It formalizes the role of observation and its contribution to the background knowledge pool.

Due to the resolution with which these facets of machine induction are focused, the STAR methodology is able to expose problematic learning areas. The methodologies reviewed previously offered no robust foundation capable of supporting further research, and their short-sightedness gave no promising avenue for additional study. Michalski's STAR, however, has enough theoretical substance to withstand the weight of subsequent investigations. Namely, research is necessary to find methods to allow STAR to autonomously construct its own generalization rules and to discover ways to gift STAR with the ability to produce its own relevant observational formulae. As difficult as these two tasks may seem, it is conceivable that STAR with these revisions would satisfactorily address the majority of machine induction issues.

Nevertheless, STAR is not without its shortcomings. It still contains no "innovative" element. Results obtained are not those exhibiting any novelty due to inspiration but are the fruit of perspiration from exhaustive rules application. The autonomous construction of new induction rules mentioned above as an addition to STAR would supply a much needed stroke of innovation — as would the power to make relevant observations on the initial data.

But it is a matter of debate as to whether true innovation could be bestowed upon a machine. Philosopher Karl Popper (1968) doubted the possibility of formalizing inductive inference for performance by a machine because, "inductive inference requires an irrational element."
STAR

STAR in Action

DNC

DNN

Figure SM-1

STAR

Methodology

Structural — tree-oriented generalization hierarchy graph

place(U.S.A) = \{Indiana, Illinois, Iowa...\}

shape(polygon) = \{triangle, rectangle...\}

dimensions(space) = \{length(), width(), depth()\}

Figure SM-2
Closing Interval Rule (for linear descriptors):
\[
\text{CTX} \land [L = a] \implies X \quad (\text{CTX} \land [L = b] \implies X
\]

Climbing Generalization Tree Rule (for structural descriptors):
\[
\text{CTX} \land [L = a] \implies X \\
\text{CTX} \land [L = b] \implies X \\
\text{CTX} \land [L = c] \implies X \\
\text{CTX} \land [L = s] \implies X
\]

Constructive Generalization Rules (use derived D(i) )

Example:
\[
\text{CTX} \land F_1 \implies X \\
\quad \text{(CTX} \land F_2 \implies X
\]

Simple Observations on the Basic Assertions

- Counting number of arguments
- Statistics observed on properties
- Elementary Monotonicity Relationships

Add to Assertions
1. Introduction

In the past, the task of arranging objects into classes has usually been based on some predefined measure of similarity. The task of conceptual clustering, however, is a matter of grouping the objects into classes that represent simple concepts describing the objects. Properties that characterize these clusters as a whole are not derivable from properties of the individual members of the classes. Therefore, there is a need for a system that is equipped with the ability to recognize configurations of objects which represent certain global concepts. This is the object of conceptual clustering. The work in this area is based on Michalski's previous work in the Star Methodology for Inductive Learning. This work also employs some new ideas, which basically extend the concepts of the star methodology.

2. Methodology

Conceptual clustering generates classes by first generating conceptual descriptions of the classes (descriptors* from the star methodology), and then classifying the objects according to these descriptions. This is known as the clustering phase and the hierarchy building phase, respectively. The system uses background knowledge to fuel the clustering phase in developing the descriptors. Among other things, this knowledge includes inference rules and a general goal or purpose of classification (represented in a Goal Dependency Network). Before discussing an example clustering session, the areas of the background knowledge, goal dependency network, clustering phase, and hierarchy building phase will be examined.

2.1. Background Knowledge

To create meaningful classifications, the system must be equipped with sufficient background knowledge. This background knowledge must include goals of classification, classification evaluation criteria, and deductive and inductive inference rules. There exists two types of background knowledge: general purpose, and domain specific. Each contributes to both phases of clustering.

The general purpose background knowledge includes fundamental constraints and criteria specifying general properties of classifications. Three main items exist in this area. The first item is the specification of the domain of each descriptor. The second is the domain type (unordered, linearly ordered, tree-structure ordered)†. The third item in the general purpose knowledge is a sequence of elementary criteria for selecting the most preferred viewpoint of the goal. In terms of the star methodology, this is the Lexicographical Evaluation Function (LEF). This is also used to control the combinatorial explosion that can result from having too many classifications from which to choose.

The domain specific background knowledge contains the inference rules specific to the particular problem. These include the deductive (i.e., truth-preserving) rules which follow simple standard logic. Also included are the generalizing inductive rules (like those of the star methodology). Another element of the domain specific knowledge is the goal dependency network (GDN) which states the general goal of classification for the system. This guides the application of the inference rules toward developing those descriptors which are most likely to be relevant.

* a descriptor is an attribute (or property) that is used in describing a class
† see the section on Star Methodology for Inductive Learning
2.2. Goal Dependency Network

There exists a large number of different but equally meaningful classifications which can be created. The problem of deciding which classification to select is resolved by assuming a general goal or purpose. This is configured in a structure of goals leading to subgoals, and then to attributes relevant to each goal. This is the goal dependency network (GDN). An example overall goal for a human might simply be to survive. This would lead to the subordinate goal of ingesting food, which would in turn lead to drinking liquid and consuming food. Relevant characteristics (or attributes) of these latter goals that support the concept of surviving might be that the food be edible, the liquid be potable, and perhaps even that the food taste good. This is summarized in the example GDN of figure CC-1.

2.3. Clustering Phase

The purpose of the clustering phase is to derive descriptors relevant to the overall goal(s) of the system which are later used for making the classifications. There are two means by which descriptors are derived. The first is by logical inference. Here the predicates/functions representing descriptors are obtained by applying the general knowledge and problem specific inference rules to the initial descriptions of the objects. These new descriptors are appended to the object description and are now attributes that can be used in building further classifications. The second method of deriving descriptors is by special computation. This special computation can be the result of an experiment, a certain device returning results from the environment, etc. It is largely an allowance for ad hoc measures.

2.4. Hierarchy Building Phase

The purpose of the hierarchy building phase is to classify all the given objects according to the descriptors (classifications) generated. It starts with building first-level conceptual classifications of all objects, and then recursively builds a classification for each sibling group of objects until some stop growth criterion is met. This is done by one of two methods. The first is called repeated discrimination, in which the problem of classification is reduced to a series of conceptual acquisition problems. This method closely follows Michalski's previous work in the star methodology. The second means of classifying is known as classifying attributes. In this, candidate classifications are generated from the initial pool of attributes (i.e., the object descriptions) or from those attributes derived by application of the inference rules and the GDN.

Both methods use the lexicographic evaluation function (LEF) to measure the quality of the generated candidate attributes. Included in the LEF are measures of aspects such as how well the objects fit the classification, the simplicity of the class description, the number of attributes that singly discriminate all classes, and the overall number of attributes that is needed to make the classification. Whereas the GDN helped to narrow the focus of the clustering phase (generating descriptions), the LEF finishes the job of narrowing by reducing the final group of classifications to those which apply best to the initial objects.

3. Operation Scenario

The problem of classifying trains, a reformulation of the "East- and Westbound Trains" problem (Michalski and Larson, 1977), is considered using conceptual clustering. Figure CC-2 shows four trains, and the object is to classify these according to some simple conceptual pattern. In this case the pattern will be visual (i.e., by shape, structure, and color). Figure CC-3 shows the goal dependency network associated with this problem. It is unclear from the work as to whether the system starts with this GDN in its entirety, or whether some of the descriptors have been developed along the way. Nonetheless, one can see how the GDN decomposes the main goal into relevant subgoals, each subgoal containing relative attributes. The final result of the classification is shown in figure CC-4. The final classification of wheels on all cars have the same color has been selected. While this is an extremely limited example, it does demonstrate the main idea of how conceptual clustering groups objects according to global concepts of a
configuration of objects.

4. Critique

Conceptual clustering makes good use of the previous work done on inductive learning. The methodology of dividing the background knowledge into general purpose and domain specific knowledge does a good job of distributing these two types of knowledge to those parts of the system where they are needed most. This use of background knowledge guides the system well toward finding relevant descriptors. However, the major problem in conceptual clustering is the use of the GDN. Both the GDN and the LEF perform well in guiding the system, but the final result seems to be nothing more than the exhaustive search capability of the computer applied toward the selection criteria (LEF) and the goals (GDN). It is the author’s opinion that the GDN is really serving as a substitute for human inference. It is very limited in scope, and is very tightly fixed to the data. This basically serves to "preprogram" the inference made by the system.

Once again, the need for some genuinely creative aspect of making inference arises. Nonetheless, the process described in conceptual clustering is a good contribution to machine learning, for it clearly shows where the advances in the field need to be made, and where such innovations could be applied.
Example Goal Dependency Network

![Diagram showing goal dependency network with nodes labeled survive, ingest, drink liquid, consume food, is_potable, and is_edible.]

- General goal
- Subordinate goal
- Relevant attribute (descriptor)

Figure CC-1

The TRAINS problem

A.

B.

C.

D.

Figure CC-2
The **TRAINS** problem

A.

D.

B.

C.

- Wheels on all cars have the same color
- Wheels on all cars do not have the same color

![Diagram](image)

**Figure CC-3**

**Figure CC-4**
Suggested Strategy for Machine Learning Research

1. Motivation

As was revealed in the systems reviewed, two major problems with current machine learning research seem to surface. These are a lack of full integration of learning strategies, and a lack of genuine inference (or "intelligence"). The primary concern of the suggested strategy, then, is to alleviate the lack of these two necessary components of a learning system. While many of the current knowledge-rich schemes of learning today employ multiple strategies (e.g., learning by instruction coupled with learning by examples), none attempt to incorporate all the strategies. In human learning, all of these strategies are employed at some time in the learning processes. Also, at the core of learning is making generalizations; and at the core of generalizing is making creative or novel inferences. Hence, no real learning system can afford to be without this capability. However, as has been shown among the critiques, no system to date possesses such an ability. The authors thereby propose to narrow the focus of research to these two issues. While short term goals are necessary in furthering research efforts, we feel that research should be limited to only those problems which ultimately in some way address the long term goal of resolving the two problems stated.

One reason for the lack in the two areas mentioned is that primarily all the research in machine learning is done independently. That is, every researcher is searching within the confines of his own limited environment trying to develop a complete system for learning. What is needed is a starting point for further research. Hence, we suggest a panel be composed of experts from the areas of Machine Learning, Cognitive Psychology, Computer Science, Electrical Engineering, and other related fields to investigate avenues for addressing the two problem areas. This panel would establish the foundation for learning research. We see three principal approaches to targeting the research: conceptual, structural, and a combined approach. The first two approaches would actually be explored in parallel until enough advances were made to combine the results. The panel would then distinguish the merits of each of the three approaches and narrow the research focus to concentrate on the best approach. The details of each approach will be discussed throughout the remainder of this document.

2. Conceptual / Theoretical Approach

This approach to research is a top-down, algorithm-based approach. It is proposed that the panel of experts develop a full cognitive model of human learning to serve as a basis to direct research efforts. By utilizing the model for a research foundation, the lack of integration of learning strategies would be alleviated. This approach is top-down in nature because the model is the overall concept which governs further areas of concentration. It is algorithm-based in that the methods developed and implemented are not hardware-specific.

Figure SS-1 shows a hypothetical model that exemplifies the objectives of the model's development. This model is developed out of the realization that human learning exists in a cycle. This cycle starts with a primary concept, derived either by example of a new concept or by making analogy to a previous one. This then prompts the individual to experiment with this new concept on his own, observing the results of this experimentation. Once discoveries have been made, the individual generalizes (i.e., clusters, classifies) the new concepts attained. This completes the cycle, ready to use the new concepts in further learning. Through the process of learning, certain knowledge bases are updated (denoted by the dotted lines in the diagram).

Once again, the model does no good if it does not aid in resolving the two main issues of integration and intelligence. Hence, given such a model, research efforts would be emphasized in the areas of the module links in the model, and in different modules to be applied toward creativity in inference. Concentration on the links between modules (e.g., interface between analogy and experimentation, or between experimentation and observation) would address the issue of integration of strategies. In order to develop
complete learning systems, it is imperative that the dependency of one strategy on another be explored. Not only will such exploration help to develop the system as a whole, but it will also result in extracting the benefits of the individual modules (strategies) in the proper perspective.

The investigation of different modules must be aimed at finding some way to introduce creativity in the inference of the system. This involves exploring new learning strategies as well as any new component which can add an element of innovation to the system. This may also mean developing new knowledge bases which significantly deviate from current technology — something like a strategies base, goals base, or a concepts base. As was stated previously, generalization is at the core of learning, and creativity/novelty is as the core of generalization. Without it, any learning system developed will be no more than a fancy program that performs complicated routines to achieve a nominal, highly predictable result.

3. Structural Approach

The central concern of the structural approach is not to proceed in a "top-down" manner with the development of a cognitive model. Instead, a strictly "bottom-up" progression is preferred. The connectionist philosophy strives to capture the physical structure of the human brain — utilizing parallel architectures to reap the benefits of information dispersion and of simple, iteration-bound algorithms.

Fahlman and Hinton [1987] make several crucial observations about elusive human learning properties that traditional AI methodologies have been unable to capture.

- Human memory appears to be associative in nature — that it is able to manage and effortlessly recall a vast quantity and variety of knowledge.
- Human pattern recognition abilities — whether involving the sensory realm or some other more abstract realm like analogy formation — far exceed present serial AI algorithm capabilities, and are relatively immune to the distractions of noisy or distorted data.
- It seems that in many cases, humans manage information using representations other than traditional symbolic assertions. This can be seen from the effort needed for human beings to describe phenomena in axiomatic terms.

In contrast to conceptual modeling approaches, connectionist schemes exhibit several strengths which address these issues. These "neural-like" architectures can automatically sculpt unit (processing element) roles to conform to the training environment — there is constraining predetermined cognitive structure. They display a natural tolerance to imperfect (noisy) environments due to the large number of contributors to the final action. They can easily draw on large distributed knowledge bases. They show an inherent fault tolerance akin to that present in holograms (if some elements malfunction, much of the knowledge can be reconstructed. This property arises because of the degree to which the data is dispersed among the processing units).

3.1. Brief Overview of Connectionist Methods

Fahlman and Hinton [1987] describe the primary characteristics of connectionism accordingly:

"The system's collection of permanent knowledge is stored as a pattern of connections of connection strengths among the processing elements, so the knowledge directly determines how the processing elements interact rather than sitting passively in a memory, waiting to be looked at by the CPU."

Figure SS-2 depicts the arrangement of functional units typical in connectionist strategies. Each of these units receives input from significant others through the "synapse-like" connections, and delivers its results to dependent units. Although not shown in the diagram, these connections may eventually return the signal flow (through circular indirection) to a previous unit or may directly feed the output back to the originator (in order to perform direct recursion).
The "exploded" functional unit of figure SS-2 shows more clearly the operation of a single processing element. From a general standpoint, the overall strategy is highly reminiscent of data flow techniques. Each functional unit, F, is responsible for a single elementary procedure (for instance, single boolean or arithmetic instructions, data manipulation operations, or device driver signals) and "fires" depending on a weighted combination of its inputs. These weights are adjusted as part of the learning process to vary the amount of contribution needed from the inputs — effectively molding the strength of these connections.

Connectionist networks can be classified according to the degree to which each unit participates in the final outcome. Local Representation schemes assign a single concept to each unit, causing the arrangement to resemble semantic networks. Because the part each unit plays in the operation of the system is well-defined and conceptually modular, local representations are well suited for the study of connectionist methods. However, the size of the informational "chunk" with which each unit is burdened causes each unit to become indispensable to the overall system operation. Hence, fault tolerance advantages are forfeited.

Figure SS-3 graphically displays a local connectionist representation describing Clyde the elephant used in the NETL system [Fahlman, 1979]. In order to process a query like:

\[
\text{Colorof(Clyde)} = ?
\]

the "Clyde" node is first activated which fires signals all adjacent units ("Male" and "Elephant"). Then, all stimulated nodes resident on the "Colorof" arc spread their influence to the head of the arc. This serves to activate the "Gray" unit — which is the answer to the query.

Distributed Representation schemes finely disperse knowledge over a great many units. Each unit then plays only a small part in the derivation of the final answer. While fault tolerance is inherent to this approach, clarity of operation is not. Still, though difficult to trace, the distributed representation has a unique property to "develop" data representations in its connections as it is tuned to the training instances.

A sample of the process used to tune a distributed connectionist representation is provided in figure SS-4. Consider a speech synthesis system which is being trained to correctly articulate English diphthongs. Each functional unit could be assigned a frequency in the sound spectrum. Another functional unit, G, responsible for producing the sound wave would accept inputs from these units and weight their contributions (in the form of amplitudes) to the overall curve. At first G tries to reproduce a "B" sound for the diphthong "BUH". The first attempt is matched against the training wave and the differences are recorded as a gradient. The components of this difference vector instruct G to strengthen or weaken the contribution (by adjusting the input weighting) of each of the corresponding frequency units for the wave. G fires again and this time is trained by another diphthong, "BAH". The resulting gradient is back-propagated to again adjust the input weightings for G. This process continues over a number of different diphthongs (eg. "BEH", "BOO", etc.) until G is able to accurately reproduce the consonant "B" sound.

4. Conclusion: A Combined Approach

The combined approach attempts to glean the strengths of both the conceptual modeling scheme and the connectionist scheme. Cognitive modeling techniques clearly define the division of labor among the various learning strategies, as well as to provide a control flow template useful for tracing learning procedures. On the other hand, connectionist strategies are free to develop their own cognitive representations as the network dynamically tunes contributions from constituent processing units. These strategies are also able to draw on vast knowledge bases distributed throughout the network.

Two research avenues stretch ahead as an integrated solution is sought. The first of these is to cognitively arrange connectionist modules. Founded on the observation that the human brain is apparently partitioned into regions responsible for certain processes, this path attempts to overlay a cognitive template upon a largely connectionist architecture (see figure SS-5). The well-defined modules and orderly interconnection specifications of the cognitive model superstructure are hoped to reduce the burden on purely connectionist schemes as they attempt to grasp abstract concepts, such as induction, observation, and so forth.
The second avenue proposes a connectionist arrangement of cognitive modules (see figure SS-6). Here, it is hoped that the dynamic properties of the connectionist method can be adopted by the cognitive modules when arranged in a network scheme. Each module may weigh the contributions from sibling modules in order to arrive autonomously at an overall interconnection scheme. It is highly probable that the human brain follows no one overall cognitive model, although its individual cognitive processes (reflected in individual learning modules) are relatively well defined. The questions involved in deciding the degree to which each module contributes to the overall process of learning can thus be settled by the system itself.

It is our belief that research along these avenues will not only result in deriving a successfully integrated learning system, but will also focus machine learning research upon fundamental issues in hopes of establishing a solid foundation for machine learning technology.
An Integrated Model for Machine Learning

Connectionist Scheme

Each unit contributes to the final result

$F(x \cdot w_0, y \cdot w_1, z \cdot w_2)$

$F$ is a simple arithmetic or boolean function, or a device signal

Figure SS-1

Figure SS-2

Page 44
Connection

NETL

Net Color of Clyde = ?

Connectionist Scheme

Local Representation

Living Thing

Needs

Oxygen

Animal

Mammal

Color of

Gray

Elephant

Clyde

Bertha

Male

Female

1

1

Connection

Connectionist Scheme

Distributed Representation

"b"

f_q

BUH

f_q

\{B A H, B E H, B O O, \ldots\}

f_q

Figure SS-3

Figure SS-4
Cognitive Arrangement of Connectionist Modules

Connectionist Arrangement of Conceptual Modules
REFERENCES


B.G. Buchanan and E.A. Feigenbaum, *DENDRAL and Meta-DENDRAL: Their Applications Dimension*.

G. DeJong, *An Approach to Learning from Observation*.

J.E. Doran, *Experiments with a Pleasure-Seeking Automaton*.
   In: D. Michie (ed.), *Machine Intelligence, Vol. 3*.


S.E. Fahlman and G.E. Hinton, *Connectionist Architectures for Artificial Intelligence*.

   In: Rumelhart and McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*.

J.J. Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*.
   In: *Proceedings National Academy of Sciences, USA, Vol. 79, No. 8*.

   In: Michalski, Carbonell, Mitchell (eds.), *MACHINE LEARNING, An Artificial Intelligence Approach*.

   In: Michalski, Carbonell, Mitchell (eds.), *MACHINE LEARNING, An Artificial Intelligence Approach*.

   In: Michalski, Carbonell, Mitchell (eds.), *MACHINE LEARNING, An Artificial Intelligence Approach*.
R.S. Michalski, *Understanding the Nature of Learning: Issues and Research Directions*  

R.E. Stepp III and R.S. Michalski, *CONCEPTUAL CLUSTERING: Inventing Goal-Oriented Classifications of Structured Objects*  


D.B. Vigor, D. Urquhart, and A. Wilkinson, *PROSE — Parsing Recognizer Outputting Sentences in English*.  
**ESA Schema Composition**

- Composing known schema in a novel way

**Schema:** Bargain

\[
\text{both } X \text{ and } Y \text{ have something the other wants}
\]

\[
\text{precondition}
\]

**Schema:** Bargain

\[
\text{Fred has $$$}
\]

\[
\text{John takes Fred's wife}
\]

\[
\text{John has Fred's wife}
\]

\[
\text{Bargain} \rightarrow \text{Extortion}
\]

**New Schema**

Note: This is done with no prior knowledge of "kidnapping" or "extortion"