Stanford Heuristic Programming Project
Memo HPP-77-6

March 1977

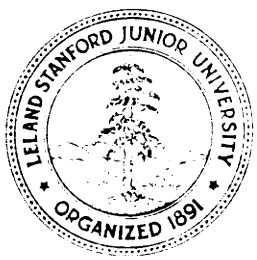
Computer Science Department
Report No. STAN-CS-77-597

MODEL-DIRECTED LEARNING OF PRODUCTION RULES

by

Bruce G. Buchanan and Tom M. Mitchell
Meta-DENDRAL Group

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY



Model-Directed Learning of Production Rules

STAN-CS-77-597

Heuristic Programming Project Memo 77-6

Bruce G. Buchanan and Tom M. Mitchell

ABSTRACT

The **Meta-DENDRAL** program is described in general terms that are intended to clarify the similarities and differences to other learning programs. Its approach of model-directed heuristic search through a complex space of possible rules appears well suited to many induction tasks. The use of a strong model of the domain to direct the rule search has been demonstrated for rule formation in two areas of chemistry. The high performance of programs which use the generated rules attests to the success of this learning strategy.

KEY WORDS

ARTIFICIAL INTELLIGENCE, LEARNING, INDUCTION, PRODUCTION RULES, **META-DENDRAL.**

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. **2492,** Contract No. DAHC-15-73-C-0435, and by The National Institute of Health under Contract No. NIH **5R24** RR **00612-07**

Model-Directed Learning of Production Rules (1)

by
Bruce G. Buchanan and Tom M. Mitchell
Heuristic Programming Project
Department of Computer Science
Stanford University
Stanford, CA 94385

ABSTRACT

The Meta-DENDRAL program is described in general terms that are intended to clarify the similarities and differences to other learning programs. Its approach of model-directed heuristic search through a complex space of possible rules appears well suited to many induction tasks. The use of a strong model of the domain to direct the rule search has been demonstrated for rule formation in two areas of chemistry. The high performance of programs which use the generated rules attests to the success of this learning strategy.

1 INTRODUCTION

Knowledge-based artificial intelligence programs derive their power from the richness and depth of their knowledge bases. It follows that careful construction of the knowledge bases is an obvious prerequisite for high performance in such systems, yet we have few alternatives to hand-crafting these for each new program. We are better off than we were several years ago, however, for it is no longer necessary to hand-craft a whole program. A rather general program, e.g., a production rule interpreter, can constitute the problem solving machinery for common problems in a variety of domains. The task-specific knowledge is then encoded in tables of inference rules, definitions, and procedures that test predicates in the domain and execute task-specific actions.

Waterman's early work [13] showed the advantages of using a production rule encoding of knowledge. It also provided a model for

---

(1) This work was supported by the Advanced Research Projects Agency under contract DAHC 15-73-C-0435, and by the National Institutes of Health under grant RR 00612-07.

learning productions by a program. Davis has made a significant contribution to our understanding of interactive knowledge acquisition [3] in which a human expert's knowledge is elicited and checked by a sophisticated acquisition program.

The Heuristic DENDRAL programs [4] are structured to read much of their task-specific knowledge from tables of production rules and to execute the rules under rather elaborate control structures. These programs interpret analytic data from organic chemical samples in order to help chemists determine the molecular structures of the samples. For a number of reasons, we made little progress with the interactive approach to building a knowledge base for DENDRAL. Instead we constructed another set of programs, collectively called Meta-DENDRAL, that aid in building the knowledge base. Meta-DENDRAL is described below in general terms that are intended to clarify the similarities and differences to other learning programs (see [12]).

2 THE TASK DOMAIN

2.1 Rule Formation

The rule formation task that Meta-DENDRAL performs is similar to the tasks of grammatical inference, sequence extrapolation, and concept formation [6],[5],[15]. Programs that perform these tasks can all be characterized as "induction" programs. Broadly speaking, the induction task is to find a general rule that can generate, classify, or explain a training set of specific instances, and correctly predict new instances. The training set can be thought of as a set of I/O pairs from a "black box" machine; the induction program is supposed to discover the generating principle used in the machine.

2.2 Mass Spectrometry

As described previously [1], the black box whose behavior we are attempting to characterize is an instrument for chemical analysis known as a mass spectrometer. The mass spectrometer bombards a small sample of an unknown chemical with high energy electrons breaking individual molecules into many fragments and causing atoms to migrate between fragments. Results of these processes are observed in a recording of the masses of the fragments that are collected. The data are usually presented in a bar graph of the relative abundance of each fragment (Y-axis) plotted against fragment mass (X-axis). From these data and a strong model of mass spectrometry, a skilled chemist can reconstruct much of the molecular structure of the unknown compound.

Throughout this paper we will use the following terms to describe the actions of molecules in the mass spectrometer:

1) Fragmentation - the breaking of an individual graph (molecule) into fragments by breaking a subset of the edges (bonds) within the graph.

2) Atom migration - the detachment of nodes (atoms) from one fragment and their reattachment to a second fragment. This process alters the mass of both fragments.

3) Mass spectral process, or process - a fragmentation followed by zero or more atom migrations.

One I/O pair for the instrument is considered to be: (INPUT) a chemical sample with uniform molecular structure (abbreviated to "a structure"), and (OUTPUT) one X-Y point from the bar graph of fragment masses and relative abundances of fragments (often referred to as one peak in the mass spectrum, or spectrum):.

Since each structure spectrum contains 50 to 100 different data points, each structure appears in many I/O pairs. Thus, the program must look for several generating principles, or processes, that operate on a structure to produce many data points. In addition, the data are not guaranteed correct because these are empirical data from an electronic instrument that produces some background noise. As a result, the program does not attempt to explain every I/O pair. It does, however, choose which data points to explain.

2.3 Syntax of Rules

The model of mass spectrometrv used by chemists is often expressed in sets of production rules. The rules (when executed by a program) constitute a simulation of the fragmentation and atom migration processes that occur inside the instrument. The left-hand side is a description of the graph structure of some relevant piece of the molecule. The right-hand side is a list of processes which occur: specifically, bond cleavages and atom migrations. For example, one simple rule is

(R1) N - C - C ----> N - C * C

where the asterisk indicates breaking the bond at that position and recording the mass of the fragment to the left of the asterisk. NO migration of atoms between fragments is predicted by this rule.

Although the vocabulary for describing individual atoms in subgraphs is small and the grammar of subgraphs is simple, the size of the subgraph search space is immense. For example, for subgraphs containing 6 atoms, each with any of roughly 20 attribute-value specifications, there are roughly 20**6 possible subgraphs. In addition to the connectivity of the subgraph, each atom in the subgraph has four attributes specified: (a) Atom type (e.g., carbon), (b) Number of connected neighbors (other than hydrogen), (c) Number of hydrogen neighbors, and (d) Number of doubly-bonded neighbors.

The language of processes (right-hand sides of rules) is also simple: one or more bonds from the left-hand side may break and zero or more atoms may migrate between fragments.

2.4 Semantic Interpretation of Rules

The interpretation of rule R1 in the above example is that if a molecule contains a nitrogen atom and two carbon atoms bonded as N-C-C then it will fragment in the mass spectrometer between the two carbon atoms, and the piece containing the nitrogen will be recorded. In a

large molecule, this rule may apply more than once. For example, CH3-CH2-CH2-NH-CH2-CH3 will show two fragments from the application of this rule:

CH3-CH2-CH2-NH-CH2
and CH2-NH-CH2-CH3 .

For a number of reasons the data points are not uniquely associated with a single fragmentation and atom migration process (rule). For example, a single process may occur more than once in a molecule (as in the above example), and more than one process may produce identical fragments (and thus produce peaks at the same mass points in the bar graph).

2.5 Space of Instances

In order to learn rules of this form, the Meta-DENDRAL program is presented with many examples of actual I/O pairs from the mass spectrometer. Each I/O pair is described as a molecular graph structure, together with a data point from the mass spectrum for that structure. The rules to be learned constitute a description of the relevant transformations in the black box. Typically we start with a training set of six to ten related molecules and their associated bar graphs, each containing 50-150 data points, or 300-1500 I/O pairs. These are drawn from an infinitely large space of possible instances, of which only a few for each structural class of molecules are available from libraries of spectra.

3 THE WORLD MODEL

3.1 Reasons for Introducing Strong Biases

Purely statistical learning programs find associations that are indicated by the data without introducing judgments about the meaningfulness of those associations. This is an advantage at times

when an investigator's bias inhibits seeing associations or when an investigator is merely looking for all associations. It is a disadvantage, however, when the number of associations is so large that the meaningful ones are lost in the chaff. Statistical pattern recognition programs have been applied to mass spectrometry with some success. Clusters of data points are found to be associated with families of molecules 80-90% of the time [7]. These programs, however, produce no meaningful explanations of why the associations are found.

In contrast to statistical approaches, Meta-DENDRAL utilizes a semantic model of the domain. This model has been included for two important reasons. First, it provides guidance for the rule formation program in a space of rules that is much too large to search exhaustively, especially when the input data have ambiguous interpretations. Second, it provides a check on the meaningfulness of the associations produced by the program!, in a domain where the trivial or meaningless associations far outnumber the important ones.

3.2 The Half-Order Theory

The base-level, or zero-order theory of mass spectrometry states that every subset of bonds within a molecule may break, and that the resulting fragments plus or minus migrating atoms will all be recorded. This zero order model of mass spectrometry is not specific enough to effectively constrain the rule search. Therefore, some general guidelines have been imposed on it in the so-called "half-order" theory.

The half-order theory asserts that bonds will break and atoms will migrate to produce data points, according to the following constraints.

Constraints on fragmentations:
- Double bonds and triple bonds do not break.
- No aromatic bonds break.
- Only fragments larger than 2 carbon atoms show up in the data.
- Two bonds to the same carbon atom cannot break together.
- No more than 3 bonds break in any one fragmentation.
- No more than 2 complete fragmentations occur in one process.
- At most 2 rings fragment in a multiple step process.

Constraints on atom migration:
- At most 2 hydrogen atoms can migrate after a fragmentation.
- At most 1 H_2O unit is lost after any fragmentation.

At most 1 CO unit is lost after any fragmentation.

One of the most helpful features of this model is its flexibility. Any of the parameters can be easily changed by a chemist with other preconceptions. Any of these assumptions can be removed and, as discussed in the following section, additional statements can be added. This power to guide rule formation will result in the program discovering only rules within a well-known framework. On the other hand, it also results in rules that are meaningful for the domain.

3.3 Augmenting the Half-Order Theory

A chemist will often know more about the mass spectrometry of a class of molecules than is embodied in the half-order theory. In these cases it is important to augment the program's model by specifying class-specific knowledge to the program. This also provides a way of forming rules in the context of additional intuitions and biases about mass spectrometry. A chemist can thus see the "most interesting" rules (as defined by the augmentations) before the other rules. For example, one might be interested first in rules that mention at least one oxygen atom before the numerous (and generally less interesting) rules that mention only carbon and hydrogen substructures.

4 THE LEARNING STRATEGY

We began with the assumption that numerical parameter estimation methods were not sufficient for the kinds of rules we wanted the program to discover in this domain due to the large number of variables required to describe subgraphs. We also wanted a chance to explore the power of heuristic search in a learning program, in the belief that efficient selection of alternative explanations is a large part of scientific discovery. As mentioned above, we also wanted to make rule discovery a model-directed procedure.

As described in more detail below, the learning program is based

on a generator of production rules of a predetermined syntax operating under the constraints of a semantic world model. In common with other induction programs, it also contains an instance selection component and a critic for evaluating potential rules.

4.1 Instance Selection

Unlike the sophisticated instance selection procedure described by Simon and Lea [11], Meta-DENDRAL merely looks at the next I/O pair, which is the next data point for the current molecule or, when there are no more for this molecule, the first data point for the next molecule. For each iteration through the learning cycle, training data are presented several spectra at a time, and are then interpreted and summarized before any rule formation takes place. In Hunt's terms [6] the data are presented in parallel, and not sequentially, for each iterative step.

Some interesting variations can be introduced to improve the instance selection procedure. For example, we have suggested elsewhere [1] allowing the program to request new data that will answer specific questions raised upon examination of the current best rule set. However, the cost of obtaining new data can be prohibitive in cases where chemical samples are difficult to obtain. Thus, the program cannot assume that it will receive each training instance which it requests.

4.2 The Critic

Any learning system must employ a critic to compare current performance with some desired standard. In Meta-DENDRAL there are two critics - one associated with rule generation and the other with rule modification. Both critics rely heavily upon examining evidential support for rules in the training data. Each rule is evaluated in terms of its positive evidence (correct explanations of data points) and its

negative evidence (incorrect predictions associated with the rule). Both critics treat evidence which is uniquely explained by a rule (unique positive evidence) differently from evidence which is shared by several rules. In particular, a data point which can be explained by only one rule is stronger evidence for the rule than a data point which has several alternate explanations.

The rule generation critic analyses candidate rules in terms of their positive evidence only; for reasons of efficiency it does not consider negative evidence. If the positive evidence of a candidate rule exhibits characteristics typical of good rules, then the critic adds this candidate rule to the list of output rules. Otherwise it decides whether the candidate rule should be further refined and reconsidered or should be abandoned.

The rule modification critic analyses both positive and negative evidence of individual rules in order to fine-tune each rule. Since rule modification involves several distinct tasks (explained below) the critic makes several types of decisions. The criteria used for making all of these decisions can be summarized as follows.

1. The set of rules as a whole should be made as compact and correct as possible without decreasing the positive evidence of the rule set.
2. Rules should be modified to increase their positive evidence without increasing negative evidence.
3. Rules should be modified to decrease their negative evidence without decreasing their unique positive evidence.

4.2.1 Credit Assignment

After evaluating performance, the critic must assign credit (or blame) to specific rules or components of rules. This credit assignment problem is an instance of a large class of such problems which have been recognized for some time [8] as important to learning programs. When blame for poor performance can be assigned to a component of a rule, modifications to that component are attempted.

For the rule generation critic, credit assignment is quite simple. During the rule search it must credit individual features in the left hand side of a rule for the evidence collected by the rule. Therefore, as each new feature is added to a rule its effect on the rule's supporting positive evidence is examined. If the effect is unfavorable (see section 4.3.2) the new feature receives the blame and is removed immediately from the rule.

There are three credit assignment problems during rule modification corresponding to the three decision criteria listed above.

(A) In order to make the rule set more concise, the critic must assign credit among redundant rules for explaining a specific data point. Credit is assigned to the rule with the strongest evidence over the entire training data set. Strength of evidence is a measure of a rule's positive and negative evidence weighted by the average intensity (Y-component) of the data points which the rule explains. In the event that two redundant rules have equally strong evidence, credit is given to the rule with the simpler left hand side.

(B) In order to increase the positive evidence of a rule, some attribute value in the left hand side of the rule must be made less specific. The critic must search for an overly specific feature to blame for excluding additional positive evidence for the rule. Currently the critic must search by trial and error for such a feature.

(C) In order to remove negative evidence from a rule, the critic must assign blame to some overly general feature. The set of attribute values common to positive evidence instances provides a menu of possible rule attribute values. Attribute values from this list are added to the rule to remove the negative evidence.

4.3 The Learning Cycle

The learning cycle is a series of **"plan-generate-test"** steps as found in many AI systems [4]. After pre-scanning a set of several

hundred I/O pairs, the program searches the space of rules for plausible explanations and then modifies the rules on the basis of detailed testing. When rules generated from one training set are added to the model, and a second (or next) block of data examined, the rule set is further extended and modified to explain the new data. That is, the program can now iteratively modify rules formed from the initial training set (and add to them), but it is currently unable to "undo" rules. Details of each of these processes are provided below.

4.3.1 Data Interpretation

The planning step in the procedure is reinterpretation of all the 'given I/O pairs in terms of the vocabulary of the specified model (the augmented half-order theory). That is, the output half of each I/O pair is reinterpreted to be a list of fragmentation and atom migration processes (potential right hand sides of rules) which are feasible explanations of the data point within the specified model. This must be done since we want the final rules to propose processes that produce data points, not just the X and Y components of the data points. This step is called INTSUM, for interpretation and summary of the initial data. For each molecule in a given set, INTSUM produces the plausible mass spectral processes which might occur, i.e., breaks and combinations of breaks, with and without migration of atoms. INTSUM then examines the spectra of the molecules looking for evidence (spectral peaks) for each process. Finally it produces a summary showing the total evidence associated with each possible process.

4.3.2 Rule Generation

After the data have been interpreted in INTSUM, control passes to a heuristic search program known as RULEGEN, for rule generation. RULEGEN creates general rules by selecting "important" features of the molecular structure around the site of the fragmentations proposed by INTSUM. These important features are combined to form a subgraph

description of the local environment surrounding the broken bonds. Each subgraph considered becomes the left hand side of a candidate rule whose right hand side is INTSUM's proposed process. Essentially RULEGEN searches within the constraints of the half-order theory through a space of these subgraph descriptions looking for successively more specific subgraphs that are supported by successively "better" sets of evidence.

Conceptually, the program begins with the most general candidate rule, X*X (where X is any unspecified atom and where the asterisk is used to indicate the broken bond, with the detected fragment written to the left of the asterisk). Since the most useful rules lie somewhere between the overly-general candidate, X*X, and the overly-specific complete molecular structure (with specified bonds breaking), the program generates refined descriptions by successively specifying additional features. This is a coarse search; for efficiency reasons RULEGEN sometimes adds features to several nodes at a time, without considering the intermediate subgraphs.

The program systematically adds features to subgraphs, always making a "parent" subgraph more specific, starting with the parent X*X. (Recall that each node can be described with any or all of the following attributes: atom type, number of non-hydrogen neighbors, number of hydrogen neighbors, and number of doubly bonded neighbors). Working outward, the program assigns one attribute at a time to all atoms that are the same number of atoms away from the breaking bond. Although different values may be assigned to each of these atoms, the coarseness of the search prevents examination of subgraphs in which this attribute is totally unimportant on some of these atoms. In addition, each of the descendants of the parent X*X is checked to see if the supporting evidence is "better" (see below) than the evidence for the parent. Those which satisfy the test become new parents for a next level of descendants with one more feature specified. For example, from the rule X*X the program will arrive, after several steps, at rule (R1)

(R1) N - C - C ----> N - C * C

In (R1) the only important features are the atom types and the connections of three atoms; the other features and atoms have been generalized away. The point of generalizing is to abstract away unimportant attributes of atoms and unimportant atoms.

The program adds specifications to candidate rules until it finds a rule that is (a) specific enough to make correct predictions and (b) general enough to account for more than a few special cases. (2)

4.3.3 Rule Modification

The last phase of the program (called RULEMOD) evaluates the plausible rules generated by RULEGEPJ and modifies them by making them 'more general or more specific. In order to extend the range of applicability of the rules, RULEMOD uses a less constrained model than RULEGEN. Rules generated by RULEGEN under an augmented half-order theory, e.g., in which only fragments containing an oxygen atom were considered, cannot immediately be applied by a performance program useing a more general model. Therefore RULEMOD refines the rule so that it can stand on its own under a more general model. In contrast to RULEGEN, RULEMOD considers negative evidence (incorrect predictions) of rules in order to increase the accuracy of the rule's applications within the training set. RULEGEN performs a coarse search of the rule space for reasons of efficiency, leaving the fine tuning to RULEMOD.

RULEMOD will typically output a set of 8 to 12 rules covering substantially the same training data points as the input RULEGEN set of approximately 25 to 100 rules, but with fewer incorrect predictions. This program is written as a set of five tasks (corresponding to the five subsections below) which we feel are closely analogous to this aspect of human problem solving.

Selecting a Subset of Important Rules. As a first step, the selection procedure is applied to the whole set of rule candidates produced by RULEGEN. The local evaluation in RULEGEN has ignored negative evidence and has not discovered that different RULEGEN pathways

may yield rules which are different but explain many of the same data points. Thus there is often a high degree of overlap in those rules and they may make many incorrect predictions.

To select rules, scores are calculated, the rule with the best score selected, and the evidence peaks supporting that rule removed from the supporting evidence for other rules. Then the whole process is repeated until either (i) all scores are below a selected threshold or (ii) all evidence has been explained. The scoring function (3) applies the standard of performance of the RULEMOD critic discussed above.

Merging Rules. Although most of the redundant rules have been deleted in the first step of RULEMOD, there may still remain sets of rules that explain many of the same data points. For any such set of rules, the program attempts to find a slightly more general rule that (a) includes all the evidence covered by the overlapping rules and (b) does not bring in extra negative evidence. If it can find such a rule, the overlapping rules are replaced by the single compact rule.

Deleting Negative Evidence by Making Rules More Specific. RULEMOD tries to add attribute-value specifications to atoms in each rule in order to delete some negative evidence while keeping all of the positive evidence. This involves local search of the possible additions to the subgraph descriptions that were not considered by RULEGEN. Because of the coarseness of RULEGEN's search, some ways of refining rules are not tried, except by RULEMOD. For example, rule (R2) below would be a specification of (R1) that RULEGEN would miss because it specifies different attributes (not just different values) for atoms that are the same distance from the broken bond (asterisk):

(R2) N - CH2 - C ----> N - CH2 * C .

In this case, the number of hydrogen neighbors is specified for the first left-hand atom but not for the first right-hand one.

Making Rules More General. RULEGEN often forms rules that are more specific than they need to be. At this point we have a choice whether

to leave the rules as they are or to seek a more general form that covers the same (and perhaps new) data points without introducing new negative evidence. Rule (R1) for example, could be made more general by removing the atom type specification on one of the first atoms next to the asterisk:

(R1′) N - C - X ----> N - C * X .

Again, because of the coarseness of its search, RULEGEN could not have considered this form of the rule. We assume here that RULEGEN produces good approximations and that RULEMOD can refine them.

Selecting the Final Rule Set. The selection procedure described above is applied again at the very end of RULEMOD in order to remove redundancies that might have been introduced during generalization and specialization.

Evaluating the Rules. Rules may be evaluated by measuring how well they explain, or "cover", the given spectra. We call this the "explanatory power" of the rules. We also want to be able to estimate how well they can be used to discriminate the most plausible structures from the rest in a list of candidate explanations of an unknown spectrum (from a known class). We call this the "discriminatory power" of the rules.

4.3.4 Integrating Subsequent Data

A requirement for any practical learning program is the ability to integrate newly acquired data into an evolving knowledge base. New data nay dictate that additional rules be added to the knowledge base or that existing rules be modified or eliminated. New rules may be added to the rule base by running RULEGEN on the new data, then running RULEMOD on the combined set of new and previously generated rules.

When an existing rule is modified, the issue is raised of how to maintain the integrity of the modified rule on its past training instances. To see this consider an example. A new training instance is acquired and, after credit assignment questions are resolved, it is

decided that rule R incorrectly "triggered" on some situation S. The left hand side of rule R must be modified so that it will no longer match S. In general there will be many possible changes to R which will disallow the match to S, but some will be better choices than others. The correct changes to R are those which do not alter past correct applications of R. Of course there is no way of knowing which of the possible changes to R will turn out to be correct upon examining still more data, and once a single change is selected the possiblity exists that backtracking will be necessary at some future point. This whole issue nay be viewed as a problem of credit assignment among the features which nake up the left hand side of R.

Different learning programs have taken different approaches to this problem of insuring that rule modifications are consistent with past training instances. Some [10] have assumed that the correct performance of each rule on past data need not be preserved. Other programs [14] keep past training instances in memory so that they may be reexamined to evaluate later changes to rules, and to allow backtracking in cases where incorrect changes to rules were made. Still other programs [15] use domain specific heuristics to select the most likely change to R.

We are currently developing a method for representing all versions of the left hand side of a rule which are consistent with the observed data for all iterations thus far. This representation is referred to as the "version space" of the rule. By examining the version space of R, we can answer the question "Which of the recommended changes to R will preserve its performance on past instances?". The answer is simply "Any changes which yield a version of the rule contained in the version space". By using version spaces we avoid the problem of selecting a single unretractable modification to R. Instead all the elements of the version space which do not match some negative instance, S, are retained, and those which do match S are eliminated. Similarly, when new data are encountered in which a situation S' is found to correctly

trigger R, only those elenents of the version space which match S' are retained.

5 RESULTS

One measure of the proficiency of Meta-DENDRAL is the ability of the corresponding performance program to predict correct spectra of new molecules using the learned rules. One performance program ranks a list of plausible hypotheses (candidate molecules) according to the similarity of their predictions (predicted spectra) to observed data. The rank of the correct hypothesis (i.e. the molecule actually associated with the observed spectrum) provides a quantitative measure of the "discriminatory power" of the rule set.

The Meta-DENDRAL program has successfully rediscovered known, published rules of mass spectrometry for two classes of molecules. More importantly, it has discovered new rules for three closely related families of structures for which rules had not previously been reported. Meta-DENDRAL's rules for these classes have been published in the chemistry literature [2]. Evaluations of all five sets of rules are discussed in that publication. This work demonstrates the utility of Meta-DENDRAL for rule formation in mass spectrometry for individual classes of structures.

Recently we have adapted the Meta-DENDRAL program to a second spectroscopic technique, 13C-nuclear magnetic resonance (13C-NMR) spectroscopy [9]. This new version provides the opportunity to direct the induction machinery of Meta-DENDRAL under a model of 13C-NMR spectroscopy. It generates rules which associate the resonance frequency of a carbon atom in a magnetic field with the local structural environment of the atom. 13C-NMR rules have been generated and used in a candidate molecule ranking program similar to the one described above. 13C-NMR rules formulated by the program for two classes of structures have been successfully used to identify the spectra of additional

molecules (of the same classes, but outside the set of training data used in generating the rules).

The rule based molecule ranking program performs at the level of a well educated chemist in both the mass spectral and 13C-NMR domains. We view this performance as indicative of the quality of the rule base discovered by Meta-DENDRAL.

6 SUMMARY

We believe that automated knowledge base construction is feasible for constructing high performance computer programs. The functional components of Meta-DENDRAL are common to other induction programs. The Meta-DENDRAL approach of model-directed heuristic search through a complex space of possible rules appears well suited to many induction tasks. The use of a strong model of the domain to direct the rule search has been demonstrated for rule formation in two areas of chemistry. The high performance of programs which use the generated rules attests to the success of this learning strategy.

FOOTNOTES

(2). The program judges a rule to be an improvement over its parent if three conditions hold: (a)the new rule predicts fewer fragments per molecule than the parent (i.e. the new rule is more specific); (b) it predicts fragmentations for at least half of all the molecules (i.e. it is not too specific); and (c) either the new rule predicts fragmentations for as many molecules as its parent or the parent rule was "too general" in the following sense: the parent predicts more than two fragments in some single molecule or, on the average, it predicts more than 1.5 fragments per molecule.

(3). The scoring function is Score = I * (P + U - 2N), where: I = the average Y-component (fragment abundance) of positive evidence data

points; P = the number of positive evidence instances for the rule; U = the number of unique positive evidence instances for the rule; N = the number of negative evidence instances for a rule.

References

1. Buchanan, B.G. Scientific theory formation by computer. in Proceedings of NATO Advanced Study Institute on Computer Oriented Learning Processes, Noordhoff, Leydon, 1976.

2. Buchanan, B.G., Smith, D.H., White, W.C., Gritter, R.J., Feigenbaum, E.A., Lederberg, J., and Djerassi, C. Automatic rule formation in mass spectrometry by means of the Meta-DENDRAL program. Journal of the American Chemical Society, 98, 20, September 1976.

3. Davis, R. Applications of meta-level knowledge to the construction, maintenance, and use of large knowledge bases. Ph.3 thesis (STAN-CS-76-552), Stanford University, July 1976.

4. Feigenbaum, E.A., Buchanan, B.C., and Lederberg, J. On generality and problem solving: a case study using the DENDRAL program. Machine Intelligence 6, Meltzer, B. and Michie, D. eds., American Elsevier, New York, 1971, 165-190.

5. Hedrick, C. A computer program to learn production systems usinp a semantic net. Ph.D. thesis, Graduate School of Industrial Administration, CMU, July 1974.

6. Hunt, Earl B. Artificial Intelligence, Academic Press, New York, 1975.

7. Jurs, P.C. in Computer Representation and Manipulation of Chemical Information, Wipke W.T., et. al. eds., Wiley-Interscience, New York, 1974: p.265.

8. Minsky, M. Steps toward artificial intelligence. Computers and Thought, Fiepenbaum, E.A. and Feldman, J. eds., McGraw-Hill, New York, 1963, 406-450.

9. Mitchell, T.M. and Schwenzer, G.M. A computer program for autonated empirical 13C NMR rule formation. Submitted to Journal of the American Chemical Society, 1977.

10. Samuel, A. L. Some studies of machine learning using the game of checkers. in Computers and Thought, Feigenbaum, E.A. and Feldman, J. eds., McGraw-Hill, New York, 1963, 71-105.

11. Simon, H.A., and Lea, G. Problem solving and rule induction: a unified view. CMU Conplex Information Processing Working Paper 227 (revised), June 1973.

12. Smith, R.G., Mitchell, T.M., Chestek, R.A., Buchanan, B.G. A model for learning systems. Submitted to the 6th IJCAI, 1977.

13. Waterman, D.A. Generalization learning techniques for automating the learning of heuristics. Artificial Intelligence, 1, 1970, 121-170.

14. Waterman, D.A. Adaptive production systems. Complex Information Processing Working Paper 285, CMU Dept. of Psychology, December, 1974.

15. Winston, P.H. Learning structural descriptions from examples. Ph.D. thesis (MIT AI-TR-231), September 1970.