OCT 13 1969

UNDERSTANDING THE INFORMATION PROCESSES OF SCIENTIFIC INFERENCE ~~REDISCOVERING SOME PROBLEMS OF ARTIFICIAL INTELLIGENCE~~ IN THE CONTEXT OF ORGANIC CHEMISTRY*

B. G. Buchanan, G. L. Sutherland

E. A. Feigenbaum

J.L./ Part I is being subjected to numerous minor textual revisions. EAF/

PART I: THE MASS SPECTROMETRY PROGRAM

The set of computer programs known as Heuristic DENDRAL is an attempt to develop machine intelligence in a scientific field. In particular its task domain is the analysis of mass spectra, chemical data gathered routinely from a relatively new analytical instrument, the mass spectrometer. Heuristic DENDRAL has been developed as a joint project of the departments of Computer Science, Chemistry, and Genetics at Stanford University. This collaboration of chemists and computer scientists has produced what appears to be an interesting program from the viewpoint of artificial intelligence and a useful tool from the viewpoint of chemistry.

----

* This research was supported by the Advanced Research Projects Agency. the HEURISTIC [illegible] DENDRAL project is the joint effort of the authors and ~~We gratefully acknowledge the collaboration of~~ Professor Joshua Lederberg, Mr. Allan Delfino, Dr. Alan Duffield, Dr. Gustav Schroll, and Professor Carl Djerassi.

For this discussion it is sufficient to say that a mass spectrometer is an instrument into which is put a minute sample of some chemical compound and out of which comes data usually represented as a bar graph. This is what is referred to here as the mass spectrum. The instrument itself bombards molecules of the compound with electrons, thereby producing ions of different masses in varying proportions. The x-points of the bar graph represent the masses of ions produced and the y-points represent the relative abundances of ions of these masses.

The Heuristic DENDRAL process of analyzing a mass spectrum by computer consists of three phases, the first, preliminary inference (or planning), obtains clues from the data as to which classes of chemical compounds are suggested or forbidden by the data. The second phase, structure generation, enumerates all possible explicit structural hypotheses which are compatible with the inferences made in phase one. The third phase, prediction and testing, predicts consequences from each structural hypothesis and compares this prediction with the original spectrum to choose the hypothesis which best explains the data. Corresponding to these three phases are three sub-programs. The program(s) have been described in previous publications, primarily in the volume of Machine Intelligence 4, and in a series of Stanford Artificial Intelligence Project Memos (4, 5, 7).

The Preliminary Inference Maker program contains a list of names of structural fragments, each of which has special characteristics with respect to its activity in a mass spectrometer. These are called "functional groups". Each functional group in the list is a LISP atom, with properties specifying the necessary and/or sufficient conditions (spectral peaks) which will appear in a mass spectrum of a substance containing that fragment. Other properties of the functional group indicate which other groups are related to this one - as special or general cases.

The program progresses through the group list, checking for the necessary and sufficient conditions of each group. Two lists are constructed for output: Goodlist enumerates functional groups which might be present, and Badlist lists functional groups which cannot be in the substance that was introduced to the mass spectrometer.

Goodlist and Badlist are the inputs to the Structure Generator, which is an algorithmic generator of all isomers (topologically possible graphs) of a given empirical formula (collection of atoms). Each Goodlist item is treated as a "super atom", so that any functional group inferred from the data by the Preliminary Inference Maker will be guaranteed to appear in the list of candidate hypotheses output by the Structure Generator.

The Structure Generator's operation is based on the DENDRAL algorithm for classifying and comparing acyclic structures.(9) The algorithm guarantees a complete, non-redundant list of isomers of an empirical formula. It is the foundation for the development of the whole mass spectrometry program.

The third sub-program is the Mass Spectrum Predictor, which contains what has been referred to as the "complex theory of mass spectrometry". This is a model of the processes which affect a structure when it is placed in a mass spectrometer. Some of these rules determine the likelihood that individual bonds will break, given the total environment of the bond. Other rules are concerned with larger fragments of a structure - like the functional groups which are the basis of the Preliminary Inference Maker. All these deductive rules are applied (recursively) to each structural hypothesis coming from the Structure Generator. The result is a list of mass-intensity number pairs, which is the predicted mass spectrum for each candidate molecule.

Any structure is thrown out which appears to be inconsistent with the original data (i.e., its predicted spectrum is incompatible with the spectrum). The remaining structures are ranked from most to least plausible on the basis of how well their spectra compare with the data. The top ranked structure is considered to be the "best explanation"

Thanks to the collaboration of Dr. Gustav Schroll, an NMR (Nuclear Magnetic Resonance) Predictor and Inference Maker have been added to the program. Thus the program can confirm and rank candidate structures through predictions independently of mass spectroscopy, bringing the whole process more in line with standard accounts of "the scientific method". Thus the Heuristic DENDRAL program is expanding from the "automatic mass spectroscopist" to the "automatic analytical chemist". Other analytical tools, such as infra-red spectroscopy will be incorporated eventually.

Three papers have appeared in the chemical literature (1, 2, 3) in the past year. The first paper describes the Heuristic DENDRAL program and tabulates numbers of isomers for many compounds. This is of particular interest to chemists because it indicates the size of the search space in which structures must be found to match specific data. The second paper explains the application of the program to ketones: the subclass of molecular structures containing the keto radical (C=O). The whole process from preliminary inference (planning) through structure generation and prediction of theoretical spectra was applied to many examples of ketone spectra. The results, in terms of actual structures identified, were encouraging. The third paper explains the application of the program to ethers.

Introducing the NMR Predictor contributed to the successful results which are described in the ether paper.

Acceptance of these papers by a chemistry journal is some measure of the program's capability, but indicates more its novelty and potential. A better measure of its performance level is provided by comparing the program with professionals. In July (1969) Professor Carl Djerassi, an eminent mass spectroscopist, asked the members of his graduate mass spectrometry seminar to interpret three mass spectra, giving them only the empirical formulas of the structures and stating the fact that they were acyclic structures - just the information given to the program. On the first problem, the program and one graduate student got the correct structure; another graduate student and a post-doctoral fellow were both close, but not correct. On the second problem, the program got the correct answer; two graduate students included the correct answer in undifferentiated sets of two and four structures; while the post-doctoral fellow missed the answer. On the last problem, the program missed the correct structure and the post-doctoral fellow included it in a pair of equally likely structures. The computer spent approximately two to five minutes on each problem; the chemists spent between fifteen and forty minutes on each. From this small experiment and their own observations, (admittedly sympathetic) mass spectroscopists have said the program performs as well as graduate students and post-doctoral fellows in its limited task domain.

Success of the mass spectrometry program is encouraging. One reason for this success is the large amount of mass spectrometry knowledge which chemists have imparted to the program. Yet this has been one of the biggest bottlenecks in developing the program. When there was only one theory of mass spectrometry in the program, viz., the complex theory in the Predictor, we were relatively insensitive to the difficulty of adding new information to the theory. Although it was a time-consuming process, it was still manageable by one programmer, working with one chemist, with most of the time spent programming as opposed to criticizing. By the time the planning phase was added to the program, it was easier to see how to shorten the task of programming by separating the chemical theory from the routines which work on the theory. The separation was by no means complete here, but it was successful enough to reduce the programming time drastically for the addition of new pieces of theory. Because the theory could be changed by changing an entry in a table, many iterations with the expert were now possible in a single one or two hour session at the console. The preponderance of time was now spent by the chemist deciding how to change the rules in the table to bring the program's behavior more in line with real data.

The organization of the Preliminary Inference Maker made the process of expanding its chemical knowledge relatively simple, compared to the process of putting knowledge into the Structure Generator and Predictor programs. Both of these programs are on their way to becoming "table driven" in much the same way as the Preliminary Inference Maker is now. (See Part IV.) Yet, re-designing the programs to allow easy additions and changes to the chemical knowledge will not solve all our problems. Because mass spectroscopy is a relatively young discipline, the theory does not exist in any sort of comprehensive codified form. Part II will discuss some of the problems of obtaining the chemical theory that has been incorporated into the programs so far. Further, the presence of any body of knowledge in the programs brings up questions of how and where this knowledge is to be represented, stored, and referenced within the programs. Part III will elaborate on these issues.

PART II: ELICITING A THEORY FROM AN EXPERT

As in the case of the Greenblatt chess program, the proficiency of the mass spectrometry program is due in large measure to the great number of times the behavior of the program has been criticized by good "players", with subsequent modifications to the program. In both cases, the heuristics of good play were not born full-blown out of the head of the programmer; they were built up, modified and tuned through many interactions with persons who were in a position to criticize the performance of the program. Yet one of the greatest bottlenecks in our total system of chemists, programmers and program has been eliciting and programming new pieces of information about mass spectrometry. One problem is that the rate of information transfer is much slower than we would like. And another problem is that the theory itself is not as well defined as we had hoped. Since these two problems are common to a broad range of artificial intelligence programs, our encounter with them will be described in detail.

It should be understood from the start that there presently is no axiomatic or even well organized theory of mass spectrometry which we could transfer to the program from a text book or from an expert. The theory is in very much the same state as the theory of good chess play: there exists a collection of principles and empirical generalizations laced throughout with seemingly ad hoc rules to take care of exceptions. No one has quantified these rules and only a few attempts have been made to systematize them. Thus the difficulty in eliciting rules of mass spectrometry from the expert lies only partly in the clumsiness of the program; the primitive state of the theory certainly contributes to our difficulty too. In our case, this problem has been compounded by having the theory of mass spectrometry in two different forms in the program: one in the prediction phase, and a less complex -- but hopefully compatible -- theory in the planning phase. The implications of this added difficulty will be discussed in Part III.

The following dialog illustrates some of the difficulties we encountered at the console, apart from machine troubles and programming problems. It is not a literal transcript, but both parties to the actual dialog agree that it is a fair condensation of some of the sessions in which they focused on the Predictor's theory of mass spectrometry. The sessions, typically, were one or two hours long. Because much of the process depended on what the program could do, both parties sat at a teletype tied to the PDP10 time sharing system in which the LISP programs resided. The expert in this dialog is A, the programmer is B, and meta-comments are bracketed.

First Session:

A: 'Since El Supremo and the rest want us to work on ketones, I guess we should get started.

B: OK. Incidentally, why are ketones important?

A: Besides being very common in organic chemistry we also know something of their mass spectrometry because they've been studied a lot.

B: What subgraph exactly will cause a molecule to be classed as a ketone?

A: The keto, or carbonyl, radical. That is -C=O (noticing B's puzzled look).

B: Then all of these are ketones?

CH3 - CH2 - C=O - R

CH3 - C=O - R

H - C=O - R

A: Wait a minute. The first two are ketones, but the last one is a special case which we should distinguish in the program. It defines the class of aldehydes.

B: So can we formulate the general rule that a ketone is any molecule containing C - C=O - C (thinking of the LISP list '(C (2 O)(1 C)(1 C))').

A: That's it.

B: Now what mass spectrometry rules do you have for ketones?

A: Three processes will dominate: alpha-cleavage, elimination of carbon monoxide from the alpha-cleavage fragments, and the McLafferty rearrangements.

B: OK. I wrote those down -- now tell me exactly what each one means. Start with alpha-cleavage -- do you mean the bond next to the heteroatom?

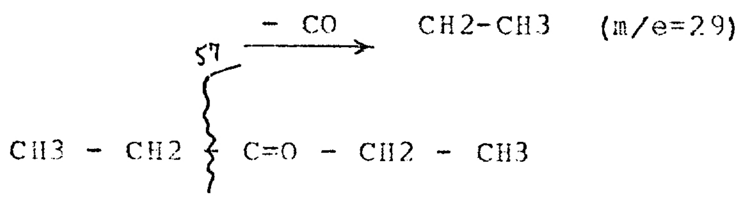
A: (Digression on notation -- often alpha-cleavage would mean this bond, but not for mass spectrometry.) ... Here alpha cleavage is cleavage of the C-C=O bond, i.e., cleavage next to the carbonyl radical -- on both sides don't forget.

B: All right. That's an easy rule to put in (translating to a new LISP function which defines alpha-cleavage as cleavage which results in a fragment (i.e., a list) whose first atom has a non-carbon atom on its property list). Shall we say the peaks are always high?

A: That will do as a start. We don't really pay much attention to intensities just as long as the peaks show up. (Reasons why exact intensities cannot be computed are explained briefly -- B's interpretation is that chemists just don't know enough about them.)

B: Now let's get on to the second process -- loss of carbon monoxide from the alpha-cleavage fragments. Would you write that out in detail? Exactly what happens to the fragment CH3-CH2-C=O for instance?

A: Let's see, that is mass 57. You will see a high 57 peak for this fragment and you'll also see a 29 peak because of this process:



B: Is that all there is to it, just drop off the C=O from the fragment (thinking of making a second call to the LISP function which breaks bonds and returns fragments). Does this happen in every case?

A: Yes it's that simple.

B: What about the intensities of these new peaks?

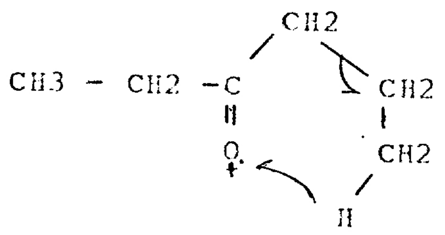
A: Well, as far as we know they'll be pretty strong all the time. Let me check some spectra here. (A looks through a notebook containing some mass spectra of simple ketones to check on the relative abundance of alpha-cleavage minus 28 peaks.) ... Well some of the time they're not recorded below mass 40 so it's a little hard to say. But it looks like the alpha-cleavage minus 28 peaks are about half as strong as the alpha-cleavage peaks in most cases.

... (A and B digress on the generality of the process; A thinks of the chemical processes, while B thinks of their LISP representation.)

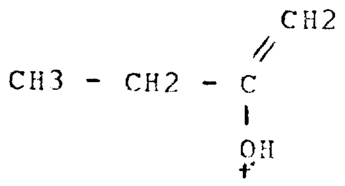
A: (Finally.) Now the last important process for ketones, and this also holds for aldehydes too, is the McLafferty rearrangement. That is just beta cleavage with migration of the gamma hydrogen.

B: You lost me again. Would you write down an example?

A: Take the case we've been working with, but with a normal propyl on the one side. Here's how we would show what's going on:



m/e=100



m/e=72

B: I guess I still don't understand. Would you mind going through that step by step?

A: We can't really say what the sequence of events is, just that from the molecular ion of mass 100 you get another ion of mass 72 -- the McLafferty rearrangement is just one way of explaining how that happens. (Digression on how chemists can be confident of what the process is, including some discussion of deuterium labeling, and meta-stable transition peaks.)

B: Suppose we wanted to tell the program about McLafferty rearrangements, as I guess we do. What do I tell it in this case?

... (A and B work out the details step by step as best they can. Both A and B suffer from B's lack of experience.)

B: Let's see if I have this straight with another example.

... (B picks an example which is too difficult for the first approximation to the rules which he understands at this point. This leads to a lengthy discussion of the conditions under which just one McLafferty rearrangement will occur, and conditions under which a "double McLafferty" will occur. At the end, B's most valuable possession is a piece of paper on which A has sketched several examples with cryptic notes. B promises to program these three rules, knowing full well that he won't get them right the first time but knowing that it will be easier for A to correct specific errors than to understand everything at once. A promises to review the published spectra of simple ketones to come up with some closer estimates of the relative intensities of the peaks resulting from these processes.)

Second Session:

B: The program and I are a little smarter than last time. But we both need some help. Let me show you what it does with a few specific examples. (B calls the program, and types in a few examples.)

... (At this point, A looks at the examples and their corresponding entries in the notebook of actual mass spectra. As he looks he diagrams the processes -- typically all processes for a molecule are superimposed on the graph structure of the molecule, with arrows pointing out of the main graph to the graphs of "daughter ions".)

A: In all these cases the alpha-cleavages are pretty good, the alpha-cleavage minus 28 peaks are OK most of the time, but I don't understand what the program is doing with McLafferty rearrangements. Also, there are a couple of things that I didn't mention last time -- I remembered them as I reviewed the ketone literature last night; so naturally the program doesn't know about them.

B: Let me write these down.

A: Two things: there is a difference in relative abundance of the alpha-cleavage peaks depending on whether it is major alpha or minor alpha, and second, very often you will see a McLafferty plus one peak after the McLafferty rearrangements.

B: Let's come back to those after you've told me what is wrong with the program as far as it goes.

A: (Looking at the examples run by the program.) In the first case you have the alpha-cleavage and alpha minus carbon monoxide peaks. But what are these others?

B: Let's see. (B inputs the example again with a switch turned on which allows him to see which major functions get executed and what their results are.) The program thinks it can do a double McLafferty rearrangement --- isn't that right?

A: It should do one McLafferty rearrangement, but I don't see the right peak for that. Here is the one it should do (sketching it out), It looks like you've tried to do something quite different.

... (After much time the errors are traced to a basic misunderstanding on B's part and some programming errors.)

B: Well I guess I'd better take care of those things before you look at more examples. Perhaps I can add those other things you mentioned earlier. What's this business about major alpha and minor alpha?

A: It is just a way of bringing the intensities predicted by the program more in line with the actual intensities. In these examples the major alpha cleavage is the alpha-cleavage in which the larger alkyl fragment is lost. (A sketches several examples to illustrate his point.)

B: What sort of general principle defines the minor alpha?

A: The larger alkyl fragment lost.

... (B agrees to put this in the program after getting it clear. A new LISP function is mostly conceptualized by now. Within a few months, however, some poor results were traced to this form of the principle, so it had to be reformulated to consider more than merely the size of the fragment.)

B: Now what about the other thing -- the McLafferty-plus-one-peaks?

A: Well, we don't know much about it, but it seems that in almost all cases where you see a McLafferty rearrangement peak you also see a peak at one mass unit higher. Of course we can't say where the extra mass comes from, but it doesn't really matter.

B: Suppose the program just sticks in the extra peak at x+1 for every x from a McLafferty rearrangement?

... (B's suggestion is motivated by the existing LISP code. The only time the program knows it has a McLafferty peak is inside one function. After a brief discussion of this, both A and B decide that the next step is to get the program to make more accurate predictions. The discussion switches, then, to adding this ketone information to the planning phase of the program.)

After deciding upon an interesting class of organic molecules, such as ketones, ethers, or amines, the first step toward informing the program about the mass spectrometry theory for that class is to ask a mass spectroscopist what rules he generally uses when considering molecules of the class. His first answer is that he expects specific fragmentations and rearrangements to dominate the entire process, with different mass numbers resulting in different contexts. He expects just four processes to explain all significant peaks in the mass spectra of acyclic ketones: (1) cleavage next to the C=O (keto) group, i.e., alpha-cleavage, (2) loss of carbon monoxide (CO) from the ions resulting from alpha-cleavage, (3) the rearrangement process known as the "McLafferty rearrangement" (migration of the gamma hydrogen to the oxygen with subsequent beta cleavage), and possibly (4) addition of a proton to ions resulting from McLafferty rearrangements. The last process is given far less weight than the first three, seemingly because there are still too many exceptions to put much confidence in it. But it is still useful enough of the time to warrant inclusion in the list. It is impossible to identify a process with any specific mass number because these processes result in different spectral lines when applied to different structures. For example, alpha-cleavage (next to the C=O) in C-C-C-C-C=O-C-C results in peaks at mass points 57 and 71 while in C-C-C-C-C-C=O-C-C the alpha-cleavage peaks are at mass points 57 and 85.

These four rules were put into the Predictor's complex theory and, in a different form, into the rough theory of the planning stage. The problems we encountered with these rules are typical of three fundamental problems we have learned to expect: (1) unanticipated concepts require additional programming, (2) counter-examples to the first rules force revisions, and (3) a false start leads to a change in strategy.

The first difficulty is just a variation on the old adage "Expect the unexpected". In our case one root of this problem is lack of communication between expert and non-expert. Because the expert tries to make his explanations simple enough for the layman he leaves out relations or concepts which very often turn out to be important for the performance of the program.

Initially the Predictor's theory treated each cleavage independently of the others. But the introduction of the concepts of major and minor alpha-cleavages destroyed this independence and forced revisions on the program. Since the expert measured the relative abundance of minor alpha-cleavage peaks in terms of the major peaks, it was essential to calculate the abundance of the major alpha-cleavage peaks first. The technique for handling this was to introduce a switch indicating whether the major alpha-cleavage had been encountered yet (with appropriate tests and settings in various places). The underlying reason for using this technique rather than another was to plug the hole as quickly as possible (and as a corollary to fix things with a minimun of reprogramming).

In the planning stage, the anticipated form of a rule was a list of peaks at characteristic mass points, (where these could be relative to the molecular weight). But in order to identify alpha-cleavage peaks in ketones the program needed to find a pair of peaks at masses x1 and x2 which satisfied the relation x1 + x2 = molecular weight + 28. So the program was extended in two ways to account for this: first, a LISP function was allowed to stand in place of an x,y pair as an acceptable rule form in the table of planning rules, and second, a function was added to the set of available rules. The function looks for n peaks x1, ..., xn which sum to the molecular weight plus k, where n and k are different for different functional groups (n=2, k= +28 for ketones).

The second fundamental difficulty in this whole process has come after the additional programming was completed to take care of new concepts, when we are in a position to try out the programs on real data. Typically these first trials uncover counter-examples to the initial set of rules: we have often been surprised at the low quality of the inferences on this first pass. For example, we quickly found that the theoretical ketone rules did not always hold for methyl ketones, i.e., for structures containing the radical -C-CH3. The alpha-cleavage on the methyl side produced a much weaker peak than was originally expected, and methyl ketones often failed to show significant McLafferty rearrangement peaks, contrary to expectations. Thus it was necessary to alter the original rule that both alpha-cleavage peaks for ketones must be high peaks, to allow for the virtual absence of the peak corresponding to loss of the methyl radical. Also, because of the methyl case it was necessary to alter the conditions which determined the strength of McLafferty rearrangement peaks in ketones.

Experimental mass spectra often contain peaks which the theory either cannot account for or would have predicted were absent and the spectra often fail to show peaks where the theory predicts there should be some. Because of this, the first attempts to use almost strictly theoretical rules in the context of real data often reveal counter-examples to the rules. A theoretical chemist, however, wants to sweep away these discrepencies -- we have heard such comments as "typing error", "recording error", "impure sample", "insensitive instrument", "uncareful operation of the instrument", and so on. In tracking down the source of the discrepancies we first check the original data to see that the computer has looked at what we wanted it to. Occasionally, our friends have even re-run samples in their own laboratory to check the reliability of the data. But our limited experience indicates that the data are seldom in error: it is the theory that needs more work.

From the chemists' point of view, the dialog process is also helpful for discovering gaps in the theory. Only when they started stating their theoretical rules as precisely as the computer program demands did they realize how little their theory of mass spectroscopy says about some simple classes of molecules. For example, when considering the class of amines, a chemist wrote out 30 interesting amine superatoms* which he believed exhausted the possibilities. A program which was developed later to generate superatoms convinced him there were, in fact, 31 possibilities. Even

Professor Carl Djerassi, author of a comprehensive book on mass spectroscopy, terms his exposition "woefully inadequate" in places because of the gaps discovered in the computer model. (Research is underway to fill these gaps.)

---

* As readers of the Machine Intelligence 4 description of Heuristic DENDRAL will remember, a superatom is a structural fragment which is treated as a single unit. For example, when given the amine superatom -CH2-NH-CH3, the program will use this structure as an atomic element without considering any structural variants of it such as -CH2-CH2-NH2. Thus several atoms in the graph can be replaced by a single superatom, at a considerable saving for the Structure Generator.

Making a false start is the third type of problem, which is usually discovered only after a few iterations of examining new results and patching rules. Because this requires backtracking and reprogramming, it is painful to realize that some early decisions were bad in light of subsequent developments. We have had courage enough to label only a few of our decisions as false starts. For example, in the planning phase we quickly got into trouble with identification rules for ether subgraphs by over-specifying the subgraphs. We had successfully attacked a previous class of molecules (ketones) by dividing the class into an elaborate hierarchy of subgraphs, each with its own set of identifying rules. But this approach was not transferrable to the new class, apparently because the mass spectrometry of ethers follows a different pattern. By the time we had defined rules for C-O-C, CH2-O-CH2, CH3-O-CH2, CH3-CH2-O-CH2 we were no longer able to make sound inferences. Thus it was necessary to start at the beginning and define a less hierarchical, broader and smaller set of ether subgraphs.

Typically it has taken weeks of interaction with a chemist at a console to proceed past the first two difficulties never knowing whether we were making a false start. However, the iterative process itself is not finished when a set of rules is found which seems to "do the right thing". Because of the number and the complexity of the subgraphs we often run into trouble because we do not have the patience to grind out the consequences of the inferences which the planning phase makes. For many examples of spectra our rules excluded so many subgraphs that, even though the program was properly instructed to put a particular superatom into every structure generated, it could not generate any structures at all. In these cases we have had to weaken the identifying rules still more -- with the result that we often let in incorrect classes of molecules to insure that we never excluded the correct ones.

The end of the iterative process to establish planning rules for a class of molecules comes when we have a set of rules which correctly identifies substructures contained in all available examples of mass spectra for that class, e.g., for all acyclic ethers. Similarly, the end of the process to establish the deductive rules comes when the chemists satisfy themselves that the predicted mass spectra agree in significant respects to the published mass spectra of a broad range of examples.

It should be mentioned that we recognize the need to clear up the bottleneck of getting new information into the computer. Here, as elsewhere, many alternative designs are open to us. For instance, we could get rid of the "middle man" in the information transfer by educating a programmer in mass spectroscopy or by educating a chemist in LISP. Or we could replace the middle man with a program designed to perform the same function as B (the layman/programmer) in the dialog above. In effect, we have been moving slowly in all three of these directions at once. But what we would most like to pursue is the design of a program to elicit information from an expert who is not also a programmer. (This seems especially attractive to the real-life B, needless to say.)

In many areas of science -- especially the rapidly expanding frontier areas -- the rules which will someday be incorporated into a unified theory exist only in an uncodified morass of recent papers and unpublished notes, and in the heads of researchers on the frontier. Because of the number and complexity of the rules, they are easy to forget, especially so in a collection that is messy. The process of codifying this collection is thus both tedious and important. For this reason automation of the dialog is of general interest: B is not the only one who stands to gain.

Because B's function is more than translating from chemical language to LISP, the program must be more than a compiler. Writing the compiler and, before that, designing a rich enough chemical language seem unavoidable in the general problem. B does even more than an interactive compiler which asks for clarifications of statements. B also asks questions to fill in gaps, he uses analogies (and occasionally even sees one), he constructs possible counter-examples, and he puts new information into all parts of the system which can use it.

Each one of these additional functions adds another level of complexity to the problem of automating the dialog. Yet the language of any particular science may be sufficiently formal and constrained that the whole problem is still tractable. In our task area these problems may be as well in hand as anywhere. The next few remarks will briefly show how they are manifested in the DENDRAL system. B's experience has been that the expert can easily overlook a logical possibility, for example, one of all possible permutations of carbon, hydrogen and nitrogen atoms in a terminal radical. Because of the exhaustive Structure Generator within the program -- in fact, at the heart of the program -- it is possible to enumerate all structures within a specified class. Thuse it is possible to use a program to check for gaps in any list of structures provided by a chemist. An important but non-trivial problem, then, is finding heuristics which will select "interesting" missing structures, that is, structures the chemist would like to know he missed.

Frequently the discussion of a new functional group will call in analogies with what has been discussed before. "Amines are like ethers", was one specific remark that B had to make sense of; a smart program should at least know what questions to ask to make sense of the analogy. It will take a much smarter program to recognize these analogies itself. The point is that the dialog will move much faster if the program can at least use analogical information.

Constructing counter-examples may often require a thorough understanding of the theory. But B has been of some help to A even though he has only a little knowledge of mass spectrometry. The dialog program might easily watch to see what kinds of cases the expert needs to patch up. This strategy now leads B to ask "But what about the methyl case?" for every set of rules that doesn't explicitly consider methyls. And, surprisingly, this reminder is often helpful.

Finally, the "middle man" in the process is sometimes expected to put pieces of theory in appropriate places of the program, and sometimes to shift information from one place to another. The difficulty here, of course, is that different parts of the program require different representations of the knowledge: the planning phase is written in terms of transforming spectral lines into structural pieces while the Predictor is written for transforming structural pieces into spectral lines. As the theory becomes more complex and as the representations diverge, it becomes more difficult to assess the consistency of the different representations. Human intelligence now decides the questions of where to put new information, how to represent it, and how to make it consistent with other statements. These questions will be discussed in the next section. Let it suffice here to say that a dialog routine cannot be blind to how and where the information will be used.

In sum, eliciting a theory from an expert is a tedious process that is worth automating. It has been our key to the wealth of knowledge not yet accessible in textbook packages. And it has benefited the scientist since it provides a means of codifying a loose collection of empirical generalizations into a theory. Automating half of the information transfer should add confidence in results as well as speed to the process. Our concern is not so much building a program which teaches itself mass spectrometry as building one which has the capacity to be taught.

PART III: GENERAL PROBLEMS OF DESIGN, SEARCH, AND REPRESENTATION

Behind the discussion of the information transfer process is the unquestioned assumption that the performance of the Heuristic DENDRAL system depends critically on the amount of knowledge it has about mass spectrometry. Thus it is necessary to be able to add more and more theory to the program in the easiest possible way -- through some such process as the dialog just discussed.

In addition to the amount of information the system has, the performance of the system also depends upon how and when that information is used during the problem solving process. Writing a program to use the theory of mass spectrometry presupposes making a choice about how and where to reference the theory. That is, it presupposes choosing one design for the systen over others, choosing an efficient search strategy, and choosing appropriate representations for the theory.

In systems science the best design is the one which maximizes the stated objective function. Thus an objective function provides a measure of performance for any design of the system, when the function is available. Unfortunately, there is no epistemological theory which allows us to define one objective function and alter the design of Heuristic DENDRAL systematically to bring its level of performance closer and closer to the objective. Our criteria for evaluating the performance of the system are admittedly intuitive: we say that a design, manifested in a computer program, is better the less time the program takes, the more compact the program is, and the more problems it can solve. (Also, an intuitive concept of elegance may lie below the performance measure as a means of judging between programs which seem to perform equally well with respect to the other measures.)

The larger problem of designing the system efficiently cannot be ignored by anyone writing complex computer programs. But design questions involve more than just programming considerations. As with other large programs, Heuristic DENDRAL is broken into segments, with each segment expected to contribute to the solution of the whole problem in such a way that the performance of the entire system is efficient over a broad class of problems. If we were given just one design to implement on a computer, the questions would be questions of coding and running efficiency. But we have been forced to realize that our first choice of design was not the best one after all, that we must concern ourselves with choosing among all possible designs for systems which perform the same task.

Apart from the fact that no completely satisfactory measure of performance is forthcoming, there remains a problem of relating the performance of the components of the system with the performance of the whole system. In some systems the parts are completely independent; thus maximizing the performance of each part results in maximizing the performance of the whole system. But in the case of this program, as in other complex systems, the components are so interrelated that the best total system is different from a collection of the "best" independent parts, because the measure of each part's contribution must bring in the goals of the other parts.

The problem of where to put theoretical knowledge into the system is one aspect of the design problem which is of particular interest to us. There are several components of this system which might profit from access to the theory of mass spectrometry if we chose to represent the theory suitably for each part. But we must balance benefits to a part of the system against cost to the whole system. For example, the addition of theory to the planning stage increases its contribution, and benefits the total system, as mentioned earlier, with only a small increase in program space. Approximately three-quarters of a second spent scanning the data to make a rough plan resulted in the saving of ten or more minutes of computer time in the successive stages of the program. By our intuitive measures of good performance, we took that as an improvement, as long as the reliability of the later parts was not undermined by hasty planning. However, in the case where we gave the planning program indentifying conditions for thirty amine subgraphs we did run into serious time trouble, but not where we expected it. We expected trouble to show up in a slow-down of the planning program, when it showed up at all. But in the amine case, the slow-down came in the generator because of the number of generation constraints added by the planning program: three to eight subgraphs, typically, would be added to Goodlist and the rest of the thirty subgraphs added to Badlist. The generator just had too much information to process. Our solution was to reduce the number of Badlist additions, since (a) this was the major source of trouble in the generator, and (b) we could be assured that we never deleted correct answers this way. Although we did increase the number of wrong answers from the generator, they would be ruled out when the predictive theory of mass spectrometry was applied later.

Woven through the pattern of alternative designs for the system are alternative search strategies which are available to the system designers. In the designs actually programmed, the over-all search strategy has been to define a subspace, generate all hypotheses in that subspace, and test each. But at least two different strategies are available to the program: (A) test each node in the subspace during generation (i.e., test partial hypotheses), and (B) generate one candidate hypothesis then use a GPS-like difference-reducing strategy to generate better hypotheses. Both of these alternatives will be discussed as a means of bringing out some of our design problems, and as a weak means of justifying the strategy used in the program.

The alternative strategy (A) has, in fact, been tried in one version of the program with only incomplete results so far. In the simplest application of this strategy, the generator consults the deductive theory at each node in the generation tree to determine whether the data indicate that an unproductive branch has just been initiated. That is, the theory is consulted to determine which partial hypotheses are not worth expanding. Unproductive branches are pruned, another node is added to each partial hypotheses, and the test is repeated. For example, part way down the search tree one branch (partial hypothesis) might be an oxygen atom with unbranched carbon atoms on either side (-CH2 - O - CH2-), and the next move for the generator might be to attach a terminal carbon to one of the carbons resulting in the partial hypothesis -CH2 - O - CH2 - CH3. Consulting the theory will tell the generator that this is a fruitful branch only if the data contains peaks at 59 and the molecular weight minus 15 (M-15), otherwise the branch would be pruned at this point. Because of the large number of nodes in the unconstrained hypothesis space, it was quickly evident that this strategy could be applied in this simple way only when the planning phase had indicated a relatively small subspace.

One reason why this alternative strategy (A) will not work well in this task area is that the theory of mass spectrometry in the program, as in the heads of chemists, is highly context dependent. The theory can say very little about the behavior of isolated atoms or small groups of atoms in the mass spectrometer without knowing their environment in the molecule. An ethyl group, (CH3-CH2-) for instance, usually produces some small peaks in the spectrum at masses 29 and M-29, but when it is adjacent to a keto radical (C=O) it will produce strong M-29 and 29 peaks (depending, of course, on the structure attached to the other side of the keto radical). When an ethyl is attached to an oxygen in an ether (CH3-CH2-O-), on the other hand, the theory predicts a peak at M-15 but not at M-29, and no peak at mass 29. More importantly, the theory can say very little about pieces of structure which do not contain at least one terminus. But the canons of structure generation begin with a node at the center of the structure, working down toward the termini. The theory can say almost nothing, for example, about a chain of carbon atoms in the center of a molecule without knowing what is at the ends of the chain. In short, it must know the context.

For any class of problems where it is difficult to validate partial hypotheses, the node-by-node search strategy is not the best of alternatives. The current design with no theory used inside the generator (and thus no node-by-node testing) is superior to the node-by-node test strategy with respect to confidence, and almost certainly with respect to time.* Only after branches of the search tree terminate, i.e., when complete chemical structures are generated, can the theory be called with confidence, for only then is the context of each piece of the molecule completely determined. But the intermediate calls to the theory will then either be incorrect or a waste of time.

---

* Those familiar with earlier versions of the Heuristic DENDRAL system may recall that a rough deductive test was once applied at each node, using what we called the "zero-order theory of mass spectrometry". The simplicity of the tests was both the beauty and the downfall of the zero-order theory. Because it was not a complex theory, the test was very cheap, and thus could be applied to every node. But it was such an oversimplified theory that it very often returned incorrect answers to the tests. We have not abandoned hope of finding heuristics which indicate circumstances under which cheap tests are reliable. We are also asking ourselves how to call the complex theory efficiently, as described in (A1) and (A2) of the text to follow. Just asking questions of this sort, and asking how to incorporate their answers (if found) into the LISP program, incidentally, have led to a successful reformulation of the program. The new code, designed to allow reference to a more general theory than the zero-order

theory, runs about twice as fast with about three-fourths the number of instructions.

Adding one or both of two levels of complexity to the node-by-node testing strategy (A), however, may make it competetive with the current test-at-the-end strategy for our problem. First, we can add some meta-theory to the testing routine or, second, we can reorganize the generator to make the theoretically significant nodes come at the top of the generation tree.

(A1) Adding meta-theory to the testing routine is relatively simple since it is possible to say a priori that the theory itself is uninformative or perhaps misleading on certain classes of partial structures. Thus the first test on a partial hypothesis is to determine whether the theory can say anything about it -- whether this partial hypothesis warrants the expense of calling the full deductive theory. In this way, the number of calls to the theory is considerably reduced. The moral seems to be that a little meta-theory goes a long way.

(A2) Reorganizing the Structure Generator is a second way to maximize the pruning ability of the deductive theory in node-by-node checking. As mentioned earlier, the canons of generation initiate each structure at the center so that generation is from the center out to the termini. So in most cases near the beginning of the generation process the testing routine provides no information which allows pruning. Testing begins to pay off only after termination of one of the branches of the partial structure. By starting the generator at a terminal atom (instead of at a central atom) the deductive theory could often prune very effectively at the top of the search tree where it is most desirable. One reason why we have not pursued this strategy, however, is that we now have no way to decide which end of the structure will make the most informative termial radicals. In those cases where the oxygen of an ether molecule, for example, lies close to one end and far from the others, as in CH3-CH2-O-CH2-CH2-CH2-CH2-CH3, the savings would be positive for the termial atom near the oxygen, but negative for the other choice.

(B) Another completely different search strategy which the program might have used is a GPS-like difference reducing strategy, mentioned above as the second alternative to the current test-at-the-end strategy. The Structure Generator could construct any molecule as an initial hypothesis -- preferably within some constraints set by a smart planning program -- and the rest of the time would be spent finding differences between the predicted and actual mass spectra and then reducing those differences by changing the structure of the candidate. Chemists find this suggestion attractive because they use somewhat the same strategy in analyzing mass spectra, since they are without the benefit of an exhaustive generator. However, they have been unable to articulate a measure of progess toward the goal or a description of the process of finding relevant differences.

Another reason the GPS strategy does not fit our problem is that unless the program keeps a precise record of hypotheses already considered, it will have trouble avoiding loops. The structural changes would be made in pieces, in response to the salient differences at any level. Thus it is quite likely that a sequence of changes, each meant to reduce one of a set of differences, would soon be in a loop because changing one piece of structure to reduce the one difference might well introduce other differences in the mass spectra.

Another important reason why the GPS framework is not suited for this problem is that the chemist does not necessarily work incrementally toward the goal, as GPS does. He may add a feature to the hypothesis at one stage which seems to introduce more differences than it reduces. And then, because of that, he may finish the problem in a few swift strokes. For example, shifting the position of a functional group in a candidate molecule may explain some puzzling spectral lines but introduce puzzles about other lines that the previous structure had explained. This strategy of temporarily retreating from the goal, so to speak, is also common in synthetic chemistry and in theorem proving. In both cases, expressions (or molecules) are introduced at one stage which are more complex than the one at the previous step, because the remainder of the problem-solving activity is thus simplified. In other words, there are certain problems for which step-by-step movement toward a goal is not the best strategy; mass spectrum analysis appears to be one of them.

Although the two alternative search strategies A and B introduce new difficulties, modifying the current strategy may well improve the program without adding serious problems. One extreme is to use a powerful enough theory in the planning stage to produce only a single unambiguous hypothesis. That is, plan the hypothesis generation process so carefully in light of data and theory that just one structure meets the constraints. This means adding much more new theory to the planning program. The planning stage now has a table of interesting and relatively common subgraphs each coupled with a set of identifying conditions. Pieces of structure for which the theory has too little context to identify their presence or absence are left out of the table entirely. The rest of the table is organized hierarchically.

However, using a powerful enough theory requires enumerating whole molecules (because the theory cannot be applied unambiguously to pieces of molecules out of the total context), resulting in an enumeration which would be far too large to catalog or search. On the other hand, enumerating subgraphs -- or pieces of molecules -- in a much more manageable list leaves ambiguities in the ways the pieces can be put together in a complete molecule. That is, if we want to plan carefully enough to isolate exactly one structure for any number of atoms, the entries in the table must specify the total context for each piece of structure. In this case the planning program must do a table look-up on spectrum-molecule pairs, obviating the need for the Structure Generator or Predictor at all. (Much work in the application of computers to analytic chemistry has this flavor.) Cataloging anything less than whole structures will result in looser constraints, since some contextual information must be omitted, and thus will result in generating more than one whole structure in those cases where there is more than one way to put the identified pieces together.

While we cannot rigorously justify our design decisions, and in particular our decision to use one search strategy over another, we have been able to explore some alternative designs. Perhaps more importantly, we have found that the Heuristic DENDRAL system is fertile ground for exploring these general problems.

Another class of problems which the system forces on us has been called "The Representation Problem". There appear to be several problems under this rubric: choosing a convenient representation for the theory, deciding when to proliferate representations, deciding when two representations are consistent, and switching from one representation to another. None of these appears to warrant the title 'the problem of representation' any more than the others; they all require solution in any system which admits any of them.

Initially, the only theory of mass spectrometry of any complexity in the program was the deductive theory in the Predictor. The most crucial aspect of the representation problem at that time -- and probably the only aspect we saw -- was choosing a convenient representation. And then, also, we held a simplistic view of what made a representation convenient. We meant, roughly, a representation that was easy to code and write programs for.

Since then it has become obvious that convenience is also conditional on the persons adding statements to the theory, as discussed in the second section. For the sake of communicating with the expert, for example, it may be necessary to cast the theory in terms of bonds and atoms at the level of the dialog, but then transfer those statements to a representation in terms of electron clouds and charge localization for the efficient operation of the program. That is, there may be a need for two representations even though there is only one theory. With only one representation it is very possible that either communication with the expert or execution of the program will become cumbersome. On the other hand, separating the internal representation from the one which is convenient for communication makes it more difficult to find mistakes in the program and to explain mistakes to the expert who must ultimately correct them.

With the addition of planning to the program, it was expedient to introduce a new representation of mass spectrometry theory which could be easily read by the planning program. Even though all of the information was already in the Predictor's theory, it was not in a form which could be easily used for planning. For example, the Predictor's theory indicates that a pair of peaks (at least one of which is high) will appear in the mass spectra of ketones as a result of breaks on either side of the keto (C=O) group. Thus, because of the appearance of C=O (mass 28) in each resulting fragment, the peaks will add up to the molecular weight plus 28. The theory in the planning program also knows this, but it uses the theory in reverse. The planning program looks for a pair of peaks in the data (at least one of which is high) which sum to M+28 as a necessary condition for the appearance of the keto group. That is, the Predictor uses structural information to infer pieces of the bar graph, while the planning program uses bar graph information to infer pieces of structure.

Duplication of information may be the preferred means to processing efficiency, even at an obvious cost in space, as it almost certainly is in this case where conditionals are read left to right in the prediction (deductive) phase and re-representations are read the other way in the planning phase. Even more critical than the space vs processing time question, though, is the question of consistency. The system has no way of checking its own theories for inconsistencies. Worrying about the consistency of different representations of the theory may be considered a waste of time, but we see this as a serious issue because of the complexity of the body of knowledge about mass spectrometry. We even have to be careful now with the internal consistency of each representation because of complexity. For example, the rules of the planning program have occasionally put a subgraph on Goodlist and a more general form of that subgraph on Badlist: to say something like "this is an ethyl ketone but it is not a ketone". Our solution to this particular problem avoids the consistency issue by allowing the planning program to check only as far as the first "no" answer in the family tree. In general, however, because of the complexity of the theory we are not confident that the programs are internally consistent, let alone consistent with each other.

The consistency problem would evaporate if there were just one representation of the theory which could be read by all parts of the system which use the theory. But it may be unreasonable to expect to find one representation which is suitable for all purposes. Another solution to the consistency question is to add either (1) a program which can read both representations of the theory to check for inconsistencies, or (2) a different representation to which modifications will be made and a program which writes the other two representations from the third after each set of changes. At the least the consistency of the whole system can be checked empirically by running examples. It may well be that this is also the best that can be done; there may be no logical proof of consistency for this vaguely stated body of knowledge. In any case, the system should be designed in such a way that the opportunities for introducing inconsistencies are minimized.

If the consistency problem is dismissed by disposing of all but one representation of the theory in a system, then the problems of representation become vacuous for that system. When different representations of the same body of knowledge remain, however, it is possible that switching from one to another inside the program will be desirable. In this system, for instance, it would be very desirable to be able to move information automatically from the Predictor's complex theory of mass spectrometry to the planning program's theory. The convenience and consistency questions just mentioned have directed attention to the benefits of switching representations. There are at least two ways of carrying it out here. First, and more generally, if the theory were suitably represented, for example in a table, a program could conceivably move pieces of information from one place to another making appropriate transformations on the way. This is very difficult for any complex body of knowledge, though, since it is difficult to put it into a perspicuous form and to write a program which can interpret it. The less general way of moving mass spectrometry theory from Predictor to Preliminary Inference Maker also appears slightly less difficult. In effect, the program can be asked to perform a "Gedanken experiment", i.e., to pose questions about mass spectrometry and answer them itself without outside help. The program already has almost all the necessary equipment for such an experiment. The major power of the idea is that there is already a systematic Structure Generator for producing the instances of molecules of any class, for example, all methyl ketones. Moreover, the Structure Generator can also produce the exemplars, or superatoms, which define the class. The Predictor tells what happens to each particular molecule in the mass spectrometer. All that remains is a program to classify the predicted mass spectra and find the common spectral features. These features are just what the planning program needs to identify the class. In this way the Predictor's theory is transferrable to the planning program.

Much of our current effort is directed to just these points: set up one central theory which the expert modifies and automatically move the new information to appropriate places. This effort requires much reprogramming, some of which is described in the next part of the paper, it requires improving the communication with experts as described in the second part, and it requires answering the critical design questions just discussed.

PART IV: TABLE DRIVEN PROGRAMS AND RECENT PROGRAMMING CHANGES IN HEURISTIC DENDRAL

Parts II and III have discussed the problems of obtaining and representing scientific theories for a computer program. Designing the actual computer programs to access the theory is another problem, which, fortunately, seems easier to solve than the others. The general programming approach, adopted after several trials, is summed up in the phrase "table driven program". The idea* is to separate the theory from the program which works with the theory by putting specific items of theory on lists and in global variables. Changing the theory, then, involves little actual re-programming. This allows experiments to be carried out with different versions of the theory, a very useful feature when dealing with a subject which is as uncodified as mass spectrometry.

----

* This idea is worked out in detail in Donald Waterman's program to learn the heuristics of draw poker (10).

A. The first of the DENDRAL programs to be written as a table driven program was the planning program (Preliminary Inference Maker) which bases most of its operation on a list of names and their associated properties. The planner has a list of functional groups and subgroups arranged in family hierarchies, e.g., (A) ketone, (A1) methyl-ketone, (A2) ethyl-ketone, etc. Associated with each group and subgroup is a set of identifying conditions. The program picks the first main functional group on its list and checks its identifying conditions against the given mass spectrum, e.g. for the subgroup C2H5 - C=O - CH2 - C - CH, we have X1 + X2 = M + 28 (alpha cleavage) and 72 high (McLafferty rearrangement). If any condition fails to be satisfied, the group and all its subgroups are ruled out - their structures are put on Badlist. If all conditions are satisfied, the structure of this group is put on Goodlist - a list of preferred subgraphs. Then subgroups will be checked in a similar way. All groups known to the program are thus considered either explicitly or implicitly. Modifying either the list of subgroups or their properties will drastically affect the behavior of the program. Yet all the theory of mass spectrometry in this program is contained in one or the other place.

B. The Structure Generator program has been table driven to a small extent; in particular, three lists, Orderlist, Badlist, and Goodlist, function as tables which determine the structures which will be generated and their order. Orderlist contains a list of all chemical atoms which the program can use. Each atom has properties such as valence, weight, symmetries, etc. Removing an atom from Orderlist effectively removes it from the domain of the Structure Generator. The relative order of atoms on Orderlist determines, to a small extent, the order of structures in the output list. Badlist is another table which controls output of the Structure Generator. If Badlist is nil, all topologically possible structures will appear. Otherwise, any structure containing one of the Badlist subgraphs is pruned from the generation tree as soon as the Badlist item first appears. This does not change the generating sequence, but rather eliminates structures from the unfiltered output list. Goodlist serves two purposes: it can determine the order in which structures are generated and it can limit generation to a specified class of structures. Those structures containing preferred substructures present on Goodlist will be generated first, while structures containing none of the preferred substructures will be generated last or not at all if generation is to be limited.

One of the basic problems inherent in the Structure Generator, however, has been its rigid insistence on following the canons of DENDRAL order as they existed four years ago when the program was written. These canons specified the canonical form of a structure, and thus the implicit generating sequence, by stating the following rules:

Count, degree, apical node, and afferent link are the attributes in decreasing order of importance.
1 is lowest count, increasing integer values are higher
The value of apical nodes follows Orderlist, usually $C < N < O < P < S$, with superatoms added at the end
1 is minimum degree, the highest degree is the maximum valence of all the atoms on Orderlist)
1 is the minimum link, 3 is the highest link

These specifications were programmed into the Structure Generator LISP code in such a widespread way that changing even the allowable ranges for attributes (let alone trying to change the order of attributes) required many separate small programming changes. Thus, it was difficult to determine all the places to change the code whenever even slight variations of generating strategy were desired.

The rigidity of the program in this respect made it very difficult to change the generating order for structures. It had occasionally been suggested that non-branching structures should be given preference, but such a suggestion was difficult to implement with the former Structure Generator. This problem has now been overcome by a substantial reworking of the Structure Generator program. A basic change in operating procedure made this possible.

This is the evaluation, at each level of structure generation where a node and link are picked and recursion is about to occur, of each choice of partial structure, and a consequent ordering of choices in a plan list. The program follows the DENDRAL canons through all values of node, link, and degree and makes a plan list of all possible ways to add the next node to the emerging structure. It orders these plans according to plausibility scores calculated by a single LISP function. Some plans may be eliminated because of "implausibility". Only then does the recursion take place, operating according to a single one of these plans, and then the process is repeated for the next node to be added to the emerging structure.

The result of this reorganization is a tremendous simplification of the generating algorithm. Instead of having six functions to generate the complete list of structures, two are now sufficient. Of the six functions (Genrad, Makerads, Uprad, Uplinknode, Upcompnode, and Updegnode), only two remain. The other four, whose jobs were to change a single structure, have disappeared. Previously Genrad constructed the single "lowest" canonical structure which could be made from an empirical formula. This structure had to be "incremented" by Uprad many times in order to obtain the entire output list. The current version of Genrad does all this for itself and returns a list of structures as its answer. Incidentally, this reduced the size of the Structure Generator by about 25%, a substantial savings; and cut execution time about in half.

This reorganization quickly caused us to notice that it would now be relatively easy to make the generator into an almost completely table driven program, by putting the DENDRAL canons (attributes and their values) on a global list. This is now possible because the canons are mainly invoked by the function Genrad and only a few other utility functions. The new idea is to form a global list of the form

```
((link 1 2 3)(node C N O)(degree 1 2 3 4))
```

which will be accessed during the process of making plans about how to enlarge the structure that is being built. In the example of the list above, the link is the least important attribute, and 1 is its least value; thus link=1 is always the first thing to be tried in generating structures. If, for some reason, it was felt that highly branched structures with heteroatoms (non-carbon atoms) near the center of the structure were the most likely, the revised form of this global list might appear as

```
((degree 4 3 2 1)(node O N C)(link 1 2 3))
```

or if desired, unbranched structures could be eliminated entirely by revising the list as

```
((degree 4 3 2)(node O N C)(link 1 2 3)).
```

This table driven program will have great use whenever some data or some chemist's special application indicate that structure generation should be limited to a very specialized class of structures.

C. The Predictor program is currently being revised in the form of a table driven program. This will permit a great simplification in the process of adding new chemical theory, as well as making the program easier to understand and correct. One large part of the effort of re-programming the Predictor is in switching representations of structures. Previously, three different representations of structures had existed there: the list notation which is characteristic of the Structure Generator (and the graph matching algorithm which the Predictor inherited), a variant of the list notation with unique numbers assigned to the nodes of the graph, and a connection list representation of structures. In the connection list representation the unique names of nodes are stored as global LISP atoms with properties declaring the bonds coming to and from each atom. Five reasons are given for switching to a complete connection list representation in the Predictor:

1. Keep the legal move generator simple.

The primary motivation for using connection lists was to represent bonds uniquely, because the legal move generator in the Predictor is of the form "move to the next bond and decide whether it breaks." In the connection list, the directedness of acyclic chemical graphs is maintained with separate indicators for the links to other nodes and the one link from another node. The list of links under the "from" indicator for all nodes, then, is a complete and irredundant list of the links in the graph. The list notation puts bonds and atoms in a hierarchy which makes this process difficult.

2. Represent fragments uniformly.

Since the Predictor sometimes needs to know what was connected to a new fragment over the broken bond, it was necessary to keep track of the names of the atoms connected by that bond. So connection lists were necessary even when the list structure of a fragment was available. But the connection list representation of structures alone is sufficient for these purposes.

3. Avoid building up and tearing apart list structures.

All connections are represented once and for all in the connection lists; temporary changes, e.g. the result of removing an atom and breaking a bond, can be represented by temporarily "pushing down" the appropriate properties. Previously, the Predictor built new list structures for each primary cleavage result and for each result of rearrangements. Then each of these had to be searched for such features as the number of double bonds one or two bonds removed from any atom in the structure. Even the common function of assigning a mass number to a fragment was messy in the list structure, partly because of the branching list structure and partly because the number of implicit hydrogens in the list structure had to be calculated each time.

4. Speed up graph matching.

In the Predictor, atoms in the list structure needed node numbers in order to specify the places at which a match occurred. This was essential because the secondary processes being modeled in the Predictor affect specific atoms. And the structure of the result is important because the result is itself checked for important subgraphs. Besides adding node numbers to the atoms in the list, it was also essential to put all hydrogen atoms into the list explicitly each time a new fragment was produced. Hydrogen atoms are often important conditions for the occurrence of secondary processes. So the list structure was no longer easy to search with the modified graph matching algorithm of the Structure Generator. A new algorithm has been written for the connection list representation.

5. Represent rings in the same notation as trees.

Since circular lists are generally undesirable, a fragment containing a ring could not be represented in the same way as an acyclic fragment. Thus the functions which searched for structural features could not be the same in both cases. Adding one additional property to show the links which make the acyclic structure into a cycle allow us to retain a list of unique bonds. At the same time, we can still find all connections for any atom quickly.

D. Interaction and interdependence of the three sub-programs of Heuristic DENDRAL must also be considered when writing and revising these computer programs. Because of the size of the combined programs, it is more practical to run them separately than to run them together. One supervisor takes care of the interaction by having each sub-program write an output file which is then the input file for the next phase of program operation. The Preliminary Inference Maker writes the file containing the empirical formula and the Goodlist and Badlist to be used by the Structure Generator. That program, in turn, reads this file, and writes another file containing the single output list of structures which it generates according to the Goodlist and Badlist specifications. The Predictor, then, reads this file to obtain its input, and calculates a mass spectrum for each structure in the file. If other tests such as an NMR prediction are to be made on the candidate structures, the supervisor interfaces the appropriate program to these others in the same way.

Although it is painful to rewrite a set of programs as large as those in Heuristic DENDRAL, the cost of modifying old programs seems to increase sharply as the number of new ideas increases. The primary motivation for completely rewriting large portions of the LISP code is to increase the program's flexibility. The major emphasis is on separating the chemical theory and heuristics from the rest of the code by putting chemical information into tables.

PART V: CONCLUSION

A few general points of strategy have emerged from the DENDRAL effort for designing a program which will explain pieces of empirical data. With regard to the theoretical knowledge of the task domain in the program, we believe that the following six considerations are important.

(1) CONVENIENT REPRESENTATION. As discussed in Part Two, the effort of eliciting a theory from an expert can be alleviated by choosing a representation of the theory in which he can converse easily. Although this may not be the best representation for internal processing, our experience has been that it is expeditious to write interface routines between the communication language and the internal one, rather than force the expert to converse in the scheme which suits the machine. This is also preferable to forcing the machine to carry on its problem solving in the framework of the dialog.

(2) UNIFIED THEORY. For reasons of consistency, the theory (or set of facts, or axioms) should be collected in one place in the program, with modifications made to this unified collection. This is compatible with having different representations of the theory for different applications, if this is desirable, as long as there are lines of communication between the special representations and the central one. If changes to the theory must be made by hand to every special representation there is a strong possibility that inconsistencies will be introduced between two representations which are intended to be equivalent. Having just one central theory to change from the outside will greatly reduce this possibility.

(3) PLANNING. In this program there is no question of the desirability of using some knowledge of the task domain, mass spectrometry, to construct a plan for hypothesis generation. However, it is not clear how much knowledge to use nor where to use that knowledge. Our one experience with using too much knowledge in the planning stage, when we were using 31 amine (nitrogen-containing) subgraphs, indicated that the planning stage could accomodate a great number of rules; but the generator was the part which became overburdened. This is only one example of the problems caused by the lack of a meta-theory for system design.

(4) DEDUCTIVE TESTS. Despite the efficacy of the planning stage, there remain ambiguities in the data which

cannot easily be resolved prospectively. In task areas such as this one, where testing at each node in the search space is not feasible, deductive tests on the terminal nodes become especially important. The Structure Generator often constructs several structures consistent with the plan because the planning stage does not reference an exhaustive table of subgraphs. Thus it is necessary to bring in deductive tests upon specific hypotheses to resolve ambiguities. The program deduces consequences of a hypothesis (together with the theory) and looks at the available data for confirmation or disconfirmation.

(5) GENERATION OF PLANNING CUES. Because the theory in the planning phase is part of the more complex theory in the Predictor it should be possible to generate planning cues automatically from the more comprehensive theory. Not only does this relieve (if not remove) the consistency worry, it also opens the possibility of generating cues which might not otherwise have been noticed. Although our own work is barely under way on this problem, the potential benefits are encouraging. In effect the program is asked to look at its theory to say what would happen if structures of a specified class were put in a mass spectrometer. Its answer is a set of identifying conditions for structures of the given class. Hitherto it has been necessary to gather experimental data to answer this question, but here exists the apparatus to generate identifying rules independently of the laboratory data.

(6) TABLE DRIVEN PROGRAMS. Separating the theory from the routines which use it facilitates changing the theory to improve it, on the one hand, or to experiment with variations of it, on the other. Although embedding the theory in the program's LISP code increases running efficiency, it seems more desirable, at this point, to increase the program's flexibility. In the Structure Generator it is useful to be able to change the canons of generation. In the Preliminary Inference Maker, the identifying rules for groups, as well as the groups themselves, change frequently and so should be easily manipulated. The Predictor's theory also needs modifying frequently, which cannot easily be done if all the theoretical statements are scattered throughout the code. A complex body of knowledge is rarely easy to modify with confidence that the result is accurate and consistent. But the confidence should increase if the statements of the theory are at least separable from the rest of the program.

Although each one of these general points provides direction for future research, each gives rise to numerous problems ranging from global design, search and representation problems to minute programming considerations. We'll know we are making progess in artificial intelligence when we can look back on these problems and wonder why they seemed difficult.

BIBLIOGRAPHY

(1) J. Lederberg, G. L. Sutherland, B. G. Buchanan, E. A. Feigenbaum, A. V. Robertson, A. M. Duffield, and C. Djerassi, "Applications of Artificial Intelligence for Chemical Inference I. The Number of Possible Organic Compounds: Acyclic Structures Containing C, H, O and N". Journal of the Americal Chemical Society, 91:11 (May 21, 1969).

(2) A. M. Duffield, A. V. Robertson, C. Djerassi, B. G. Buchanan, G. L. Sutherland, E. A. Feigenbaum, and J. Lederberg, "Application of Artificial Intelligence for Chemical Inference II. Interpretation of Low Resolution Mass Spectra of Ketones". Journal of the American Chemical Society, 91:11 (May 21, 1969).

(3) G. Schroll, A. M. Duffield, C. Djerassi, B. G. Buchanan, G. L. Sutherland, E. A. Feigenbaum, and J. Lederberg, "Application of Artificial Intelligence for Chemical Inference III. Aliphatic Ethers Diagnosed by Their Low Resolution Mass Spectra and NMR Data". Journal of the American Chemical Society (in press).

(4) G. Sutherland, "A Family of LISP Programs", to appear in D. Bobrow (ed), LISP Applications (also Stanford Artificial Intelligence Project Memo No. 80).

(5) B. G. Buchanan, G. L. Sutherland, and E. A. Feigenbaum, "Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry". In Machine Intelligence 4 (B. Meltzer and D. Michie, eds) Edinburgh University Press (1969), (also Stanford Artificial Intelligence Project Memo No. 62).

(6) C. W. Churchman and B. G. Buchanan, "On the Design of Inductive Systems: Some Philosophical Problems". British Journal for the Philosophy of Science, (Autumn 1969).

(7) J. Lederberg and E. A. Feigenbaum, "Mechanization of Inductive Inference in Organic Chemistry", in B. Kleinmuntz (ed) Formal Representations for Human Judgment, (Wiley, 1968) (also Stanford Artificial Intelligence Project Memo No. 54).

(8) E. A. Feigenbaum, "Artificial Intelligence: Themes in the Second Decade". Proceedings of the IFIP68 International Congress, Edinburgh, August 1968 (in press), (also Stanford Artificial Intelligence Project Memo No. 67).

(9) J. Lederberg, "DENDRAL-64 - A System for Computer

Construction, Enumeration and Notation of Organic Molecules as Tree Structures and Cyclic Graphs", (reports to NASA, unpublished).

(10) D. A. Waterman, "Machine Learning of Heuristics", PhD Dissertation (Stanford University Computer Science Department), (also Stanford Artificial Intelligence Project Memo No. 74).