STANFORD ARTIFICIAL INTELLIGENCE PROJECT
Memo No. 49

February 15, 1967

# DENDRAL - A COMPUTER PROGRAM FOR GENERATING AND FILTERING CHEMICAL STRUCTURES

by Georgia Sutherland

Abstract: A computer program has been written which can generate all the structural isomers of a chemical composition. The generated structures are inspected for forbidden substructures in order to eliminate structures which are chemically impossible from the output. In addition, the program contains heuristics for determining the most plausible structures, for utilizing supplementary data, and for interrogating the on-line user as to desired options and procedures. The program incorporates a memory so that past experiences are utilized in later work.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183)

Table of Contents

## 1. DENDRAL REPRESENTATION OF CHEMICAL STRUCTURES

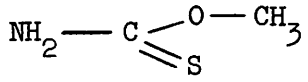
Dendral is a system of topological ordering of organic molecules as tree structures.* The essential features are detailed in the first of the references and are summarized here.

Proper Dendral includes precise rules to maintain the uniqueness and the non-ambiguity of its representations of chemical structures. Each structure has an ordered place, regardless of its notation; the emphasis is upon topological uniqueness and efficient representation of molecular structures. The principal distinction of Dendral is its algorithmic character. Dendral aims (1) to establish a unique (i.e., canonical) description of a given structure; (2) to arrive at the canonical form through mechanistic rules, minimizing repetitive searches and geometric intuition; and (3) to facilitate the ordering of the isomers at any point in the scan, thus also facilitating the enumeration of all of the isomers.

The Dendral representation of a structure is made up of operators and operands. The operators are valence bonds issuing from an atom. Each bond looks for a single complete operand. An operand is (recursively) defined as an unbonded atom, or an atom whose following bonds are all satisfied in turn by operands. Hydrogen atoms are usually omitted, but are assumed to complete the valence requirements of each atom in the structure. If the structure has unsaturations (one unsaturation for each pair of hydrogen atoms by which the structure falls short

*References: J. Lederberg, DENDRAL-64, A System for Computer Construction, Enumeration and Notation of Organic Molecules as Tree Structures and Cyclic Graphs, Parts I-V, Interim Report to the National Aeronautics and Space Administration, December 1964.

of saturation), these are indicated by locations of double and triple bond operators. Single, double, and triple bonds are represented by . , : , and ⁝ respectively. The operator : may be represented by = and the operator ⁝ by $ or < depending on the available character set.

As an example, the molecule 

has one unsaturation and may be written in many ways, including:

1) C.O.C.:NS
2) C.O.C:.SN
3) O..CC.:NS
4) O..C.:NSC
5) C..:O.CNS
6) C.:.O.CSN
7) C.:.NSO.C (canonical)
8) C:..SNO.C
9) S:C..O.CN
10) N.C.:O.CS

Each of these ten notations is a non-ambiguous representation of the molecule. However, proper Dendral also specifies that the representation be unique. The key to obtaining the unique or "canonical" representation is the recognition of the unique center of any tree structure and the subsequent ordering of successive branches of the tree.

The centroid of a tree-type chemical structure is the bond or atom that most evenly divides the tree. A molecule will fall into just <u>one</u> of the following categories, tested in sequence. Let V be the count of non-hydrogen atoms in the molecule. Then either

A. <u>Two central radicals</u> of equal count are either (1) united by a leading bond (V is even) or (2) sister branches from an apical node (V is odd); or

B. <u>Three or more central radicals</u>, each counting less than V/2, stem from a single apical node.

In the first case, the centroid is a bond, and the canonical representation is an operator followed by two operands. In the other two cases the centroid is an atom, and the canonical representation is an operand in the form of an atom followed by two or more bonds and operands. In every case where two or more bonds follow an atom, the operands must be listed in ascending Dendral order.

Dendral order (or simply "weight") is a function of the composition and arrangement of a structure and finds its primary use when comparing two operands (radicals). The weight of a radical is evaluated by the following criteria (in descending significance): count, composition, unsaturation, next node, attached substructures.

Count is the number of skeletal (non-hydrogen) atoms. Of two radicals, the one with the higher count is of higher weight.

Composition refers to the atoms contained in the radical. An arbitrary ordering of the atoms makes carbon less than nitrogen less than oxygen less than phosphorus less than sulfur, $C < N < O < P < S$. (This ordering is alphabetical as well as by atomic number.) When comparing two radicals of the same count, the one with the fewer number of carbons has lesser weight. If carbons are equal, the one with the fewer nitrogens is of lesser weight. And so forth.

Unsaturation counts the number of extra bonds (1 for a double bond, 2 for a triple bond) in the radical, including those (if any) in the afferent link (the bond leading into the radical). Of two radicals, the one with the greater number of unsaturations has the greater weight.

The next node or apical node refers to the first atom in the radical (the one connected to the afferent link). When comparing two apical nodes, the following three criteria are evaluated (in order of decreasing significance):

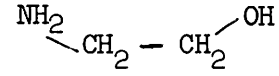
Degree is the number of efferent (attached) radicals. The apical node with the most radicals attached to it has the greater weight.

Composition refers to the type of atom. A carbon atom is the lowest apical node, while a sulfur atom is the highest.

Afferent link refers to the bond leading to the apical node. A single bond afferent link is the lowest, a triple bond is the highest.

If the above criteria fail to determine which of two racicals has the greater weight, then the radicals appendant on the two apical nodes must be arranged in increasing order and compared in pairs. The first inequality in weight of appendant radicals determines the relative weight of the original radicals.

The canonical representation for the molecule in the example given earlier is notation #7. It must be a central atom molecule since its count (ignoring hydrogen atoms) is 5; and the non-terminal carbon atom is the only atom which has all its appendant radicals with counts less than 5/2. Of the three appendant radicals, the one containing two atoms has the highest count and thus is the heaviest. Of the two radicals containing a single atom each, the one with the double bond is the heavier because it has more unsaturations.

Even-count molecules may have a bond for center, if the count of the molecule is evenly divided by cutting that bond. Thus, the canonical form for  is .C.NC.O, a leading bond, the first dot, calling for two operands.

The collection of rules and conventions described above provides a unique and non-ambiguous representation for any non-ringed chemical structure. (Ringed structures have been dealt with by more complex rules.) In addition, the rules also allow us to construct the "lowest" structure which can be made from a composition (collection of atoms). Once this lowest structure has been made, it may be transformed by a process of rearranging its atoms and unsaturations into the "next to lowest" structure. This "incrementing" process may be continued until the "highest" structure has been made.

The computer program which is described in later sections of this report is designed to do these operations and therefore to construct all of the (topologically possible) isomers of a composition.

## 2. DENDRAL IMPLEMENTATION

The task set forth in the Dendral Report is the manipulation of chemical graphs to produce all the isomers of a given chemical formula. The list processing language, LISP, was used to write a computer program implementing the proposed scheme. The choice of this language led to the representation of the chemical graphs as tree type lists.

The list representation is a straightforward translation of Dendral dot notation. The bonds are represented by the integers 1,

2, and 3. Each radical is a sublist and is enclosed in parentheses. The list notation for a structure is a three part list. The first part specifies the afferent link; the second part specifies the apical node; and the third part specifies the efferent radicals in list notation. If the structure is a molecule, its afferent link will be NIL . For a central-bond molecule the apical node is NIL also. Central-bond molecules have two efferent radicals; central atom molecules have two or more efferent radicals, and radicals may have any number (including zero) of efferent radicals.

As an example, the dot notation C.:.OOS.N is translated into (NIL C (1 O)(2 O)(1 S(1 N))); and the molecule represented in dot notation as .C.:SS C.O.P..CC:C becomes
(NIL NIL (1 C(1 S)(2 S))(1 C(1 O(1 P(1 C)(1C(3C)))))).

The list notation is easily manipulated by the computer program, so all operations are performed using this representation. Output, however, is given in Dendral dot notation. Certain functions are available within the Dendral program for converting back and forth between the two types of notation. The function INDOT reads dot notation and converts the dots to integers. (This is the so-called Dendral Polish notation.) The function UNSTRING converts Polish notation to list notation. The function DOTORD also reads dot notation, but converts it to cannonical dot notation. The functions MOLORD (for molecules) and RADORD (for radicals) convert list notation into cannonical list notation. The functions TOPMOL (for molecules) and TOPRAD (for radicals) convert list notation to dot notation and print the dot notation.

The Dendral program operates on a chemical composition in order to produce the structural formulas of all isomers with that composition. A composition is a list of numbers of atoms, such as: $C_4H_{10}$, or $C_2H_5NO$. The internal form of a composition is a list of dotted pairs in which the number of hydrogen atoms is replaced by the number of unsaturations (extra bonds) in the composition. The formula for obtaining the number of unsaturations (U) in a composition is the following:

$$N = 2U = \sum_{\substack{\text{all types of atoms}}} \left(\begin{matrix}\text{number of atoms} \\ \text{of this type}\end{matrix}\right) \times \left(\begin{matrix}\text{valence of this} \\ \text{type of atom}\end{matrix} - 2\right)$$

Examples of composition lists are ((U . 0)(C . 4)) and ((U . 1)(C . 2)(N . 1)(O . 1)).

Some compositions imply molecular structures while others imply radicals. If N = 2U is an odd integer, then the structures will be radicals. If N = 2U is even (including zero), the structures will be molecules. If N = 2U is negative, there are no structures possible for that composition.

In the process of generating all structures corresponding to a chemical formula, the program starts with the formula given as a composition list. The program then constructs the molecule* of lowest Dendral value for this composition. Once the molecule of lowest

* The current discussion will concentrate on molecular structures. Radicals are generated in an analogous fashion.

value is established, the process of generating all the isomers consists of incrementing this molecule to the next higher Dendral structure, then taking the new molecule and repeating the process until it is no longer possible to make a molecule of higher Dendral value than the previous one.

The following computer program is a precise statement of the generating algorithm mentioned in the Dendral Report, with the following added features:

a. The number and type of radicals that can be made from a given composition may be determined in advance and placed in a dictionary list for that composition. When a dictionary list is encountered (during structure generation) for a composition, the algorithm will generate only those radicals on the dictionary list. Thus a dictionary is a "memory" for past work. The program knows how to make use of its memory efficiently.

b. At any given node it is possible to represent the efferent radical in an implied format (by composition rather than by structure).

c. Several options are available which may limit the output by eliminating or bypassing some structures which are topologically possible but which are not of interest or perhaps are not chemically meaningful.

## 3. THE DENDRAL PROGRAM

The computer program to implement Dendral is written in

LISP. The program is made up of over a hundred separate LISP functions which call each other to perform certain tasks relevant to different parts of the structure generation. Most of the functions are simple utility programs. The main logic is contained in eight or ten primary functions which are described in the following pages.

The LISP function called GENMOL will make the molecule of lowest Dendral value from a given composition. If the composition contains an even number of atoms, GENMOL will attempt to make a central-bond molecule by making two radicals with equal count and identical afferent links. If this fails or if the composition contains an odd number of atoms, then a central node is selected, starting with the atom of lowest Dendral value ($C < N < O < P < S$) in the composition. MAKERADS is then given the remainder of the composition and instructed to make two efferent radicals of lowest Dendral value. If this fails, then GENMOL selects the next possibility for the central node and the process of generating two efferent radicals is attempted again. If no atom in the composition leads to a structure with degree two, then the lowest atom is again chosen for the central node, and MAKERADS attempts to make three efferent radicals, and so forth.

The function MAKERADS takes a composition and produces a list of a specified number of radicals. These radicals have the lowest Dendral value which is possible in view of the valence requirements specified in the arguments to MAKERADS. Each radical except the last must have a specified number of atoms, and the radicals must be listed in increasing Dendral order. First the total composition is split into

the proper number of compositions by MAKELSTCLS. These separate compositions (listed in increasing Dendral order) are then made into radicals by GENRAD. If any of the compositions cannot be converted to a radical then this composition is incremented to the next greater Dendral value. Any compositions which are greater than the one incremented are reset to compositions greater than or equal to the new one. The process of converting these lists to radicals is then continued as before. Once the compositions are all converted to radicals, the radicals are checked to see if certain valence requirements are met. If not, an attempt is made to decrease the afferent link of each radical in turn starting with the radical of greatest Dendral value. When a radical has the lowest allowable afferent link and the valence requirements are not satisfied, then the compositions of this radical and those of greater Dendral value are incremented by trading atoms among compositions. New radicals are generated from each composition and the process of decreasing afferent links continues.

GENRAD takes a composition and makes the radical of lowest Dendral value within valence limitations specified. If the composition consists of a single atom then this atom with an afferent link accounting for all the unsaturations in the composition will be the desired radical. Otherwise, the lowest apical node is selected from the composition, the lowest possible afferent link is chosen (considering the atoms and unsaturations in the remaining composition), and GENRAD is used (recursively) to generate a single efferent radical

from the remaining composition. If this fails, the afferent link will be increased and GENRAD tried again. If this too fails, then the composition of the apical node will be increased, the afferent link reset, and the whole process repeated until there are no more possibilities for a new apical node. If this process fails, then it is repeated allowing GENRAD to generate two efferent radicals. The number of efferent radicals continues to be increased if necessary and is limited only by the valence of the apical node.

UPRAD increments a given radical to its next highest Dendral value. This is done by first trying to increment one of the efferent radicals from the apical node of the radical (if there are any efferent radicals). Starting with the efferent radical of greatest Dendral value, UPRAD is applied to obtain a new efferent radical and to reset all of the efferent radicals of greater Dendral value so that they are as low as possible but still higher than the one just generated. If this proves unsuccessful, then an attempt is made to, first, raise the afferent link of the radical (using the function UPLINKNODE), or to increase the composition of the apical node (using the function UPCOMPNODE), or, finally, to increase the degree of the apical node (using the function UPDEGNODE). If none of these is successful, and there are radicals of higher Dendral value concurrent with the one being incremented, then MAKERADS is used to make a new set of efferent radicals in which the composition of the first radical is greater than that of the present radical being incremented.

UPMOL uses UPRAD in an attempt to step up one of the

efferent radicals from the center of a molecule. These efferent radicals are given to UPRAD one at a time starting with the one of greatest Dendral value. If it is not possible to step up any of the efferent radicals, then an attempt is made to find a central node of greater compositional value and then use MAKERADS with the remaining composition to find the set of efferent radicals for this central node. (If this is a molecule with a bond as its center, then instead of finding a new central node UPMOL tries to increase the value of the central bond.) If these steps fail, then UPMOL tries to raise the degree of the central node while resetting this node to its lowest value.

The three functions UPLINKNODE, UPCOMPNODE, and UPDEGNODE operate on a radical and attempt to make the next higher radical out of the same elements. In each case the function MAKERADS is the primary tool. UPLINKNODE is asked to return a radical with a higher afferent link. UPCOMPNODE is asked to return a radical with an apical node of higher composition. UPDEGNODE is asked to return a radical in which the apical node has an increased number of efferent radicals.

The main control function for the structure generation is a function called WORKLIST. WORKLIST causes generation of all structures from all composition lists which are subsets of a given composition list. The user specifies this composition list and requests either molecules or radicals to be generated. WORKLIST calls TESTDENDRAL (for molecules) or TSTRAD (for radicals). TESTDENDRAL first calls GENMOL to obtain the lowest molecular structure for that composition list. The resulting structure is printed, and TESTDENDRAL then alternately calls UPMOL and OUTDEN (a print function) until UPMOL

returns the value NIL, indicating that no higher structures can be generated. TSTRAD calls GENRAD and UPRAD in an analogous fashion for radicals.

The usual case is that the user has in mind a single chemical formula for which he wishes to see all the allowable structures. The function CHNOPSXVQW takes a chemical formula and converts it to a composition list by calculating the number of unsaturations. CHNOPSXVQW also determines whether molecules or radicals will result. Then WORKLIST is called and instructed to generate structures from this single composition list.

The input to the function CHNOPSXVQW is a list of ten elements, corresponding to the letters in the name of the function. The first six elements are the number of carbon, hydrogen, nitrogen, oxygen, phosphorus, and sulfur atoms in the formula. X is the name of any other atom in the formula. (If X is the word NIL, then no other atoms are present). V is the valence of X; Q is the number of X atoms in the formula; and W is the atomic weight of X.

Other functions have been supplied to accept different forms of input and call CHNOPSXVQW after constructing the appropriate list of arguments. The function ISOMERS takes a single argument which is a composition name (i.e., $C_3H_1O$, $C_2H_5NO$, $CH_3COOH$, etc.). The function DENDRAL takes no arguments, but later requests the user to specify desired options and to input a composition name. The function STRUCTURES alternately requests composition names and calls ISOMERS so that it is easy for the user to examine the isomers of many compositions in succession.

## 4. MODIFICATIONS TO THE DENDRAL PROGRAM

The program described above does indeed produce all the structural isomers from a chemical composition. Sometimes, however, the number of isomers is so large that a user may not want to see all of them. Thus, the program will pause after each N structures (N may be set by the user) and ask the user whether or not to continue generating structures.

The "model" of chemistry in basic Dendral includes the following subjects:

a) Atoms; there are seven distinct atoms (C,H,N,O,P,S and X). Of these, all are treated the same except H.

b) Valence of an atom; valence specifies the number of attachments an atom may acquire.

c) Unsaturations; the program knows that unsaturations indicate multiple order attachments and the program knows how to calculate the number of unsaturations for any composition.

Using the above concepts and the Dendral rules for building structures, the program constructs all of the topologically possible structures for any given composition. However, any chemist inspecting the output list for a composition would realize that the program knows little chemistry, since many chemically meaningless structures are included in the class of topologically possible structures. So additional information and programming was added to the Dendral Program in order to eliminate certain types of structures which are always chemically impossible.

This was done in two ways. The first uses the notion of illegal attachment. For instance, an oxygen atom cannot be attached to another oxygen atom. (Other such conditions hold for nitrogen and sulfur atoms.) Whenever the program picks an atom to be the apical node of a radical, it checks the partially built structure to be sure that this atom will not be attached to a forbidden atom or structure. The second and more general way of avoiding certain chemically impossible structures is by presenting the program with a list of impossible substructures. Each generated structure is then examined and rejected if it contains any of the forbidden substructures. The process of checking for certain substructures led to the incorporation of a graph matching algorithm in the Dendral program. The graph matching process is described in detail in later sections of this report.

Eliminating chemically impossible structures from the program output means that the program no longer creates all structural isomers of a chemical composition. Rather, the program now contains some knowledge of chemistry. The on-line user of the program may want to impart more knowledge to the program. A small step toward this goal is provided in three ways.

a) Supervising structure generation:

The program can be made to pause every time it is about to add an atom to a partially generated radical. The program prints out the current status and requests permission to continue. A "no" answer at this point should invoke some (as yet unwritten) learning dialog

to find out why not to continue.

b) Examining molecular partitions:

Certain groupings of atoms are more plausible (chemically speaking) than others. The program can be made to print (in advance of any structure generating) all the partitions of the composition. The user may then rearrange the list of partitions, eliminating any in which he is not interested. The program will then generate structures in the order specified by the list of partitions. The learning process which could be inserted at this point would interrogate the user as to why some partitions were left out and why some are more plausible than others.

An option is included at this point which allows the user to rearrange the partition list on the basis of a built-in plausibility function. This function calculates a "plausibility score" for each partition and then rearranges the partition list, putting the partitions with the highest scores at the beginning of the list. The criteria which are considered in calculating the score of a partition are built into the program.

The function GETSCORE examines a partition (a list of compositions) and returns a number between 1 and 10 which is an assessment of the value of the partition. If any of the compositions with more than one atom contains no carbon atoms, then the score for the whole partition is 1. Otherwise the score is the average of two numbers:

The first number is obtained by considering the proposed center of the structure. A central bond is assigned the value 10. A non-carbon central atom is assigned the value 3. A central carbon with degree two is assigned the value 10. A central carbon with degree three is assigned the value 6. And a central carbon with degree four is assigned the value 1.

The second number is the average of N values, where N is the number of compositions in the partition (aside from the central node, if present). If the composition is COOH then the value is 10. Otherwise the value is a number between 0 and 10 measuring how closely the proportion of carbon to non-carbon atoms matches the overall proportion for the whole partition.

c) Using other data (spectra):

Spectral information may be used to further shorten the list of isomers of a chemical formula. A spectrum is a list of numbers corresponding to weights of substructures. The Dendral program "knows" the atomic weight of each atom, and can calculate the weight of any structure or substructure. If a spectrum is present, then no structure or substructure will be generated unless its weight appears in the spectrum. A function called HSTGRM will calculate the spectrum of a structure in list notation. Another function called USESPECTRUM requires the user to input a spectrum or structure from which to generate a spectrum. This function then sets SPECTRUM and calls structure generating functions.

When structure generation is complete,

USESPECTRUM sets SPECTRUM to NIL .

Future plans for the Dendral program include expanding the dialog, partition, and spectral facilities, streamlining the graph matching algorithm, allowing several levels of memory, and providing more utility functions for the benefit of the program user.

## 5. GRAPH MATCHING IN DENDRAL

The Dendral program may be made to generate all topologically possible structures from a chemical formula, restricted only by the valence limitations on the atoms. Many of the structures generated by the Dendral algorithm are not chemically meaningful because they contain certain impossible substructures. The forbidden substructures were few enough in number that they could be listed; and a graph matching algorithm was introduced to check each generated structure against the list of forbidden substructures. If a structure generated by the rules of Dendral is found to contain even one of the forbidden substructures, it is not acceptable output, and the Dendral program attempts to find the next higher structure which does not contain a forbidden substructure.

The first graph matching algorithm that was implemented to compare Dendral-generated structures was essentially that of E. H. Sussenguth Jr. of Harvard. This algorithm is described in detail in an appendix to this report. After using this algorithm for some time, however, it was determined to be inefficient in the sense that previous work was constantly being repeated. This resulted from the

fact that graph matching was performed at each level of structure generation, whenever an atom was added to a partially built structure. The Sussenguth method considers the total structure and calculates characteristics of each node (atom). Yet, the nature of the structure generating algorithm implied that if any forbidden substructures were to be present, they would have to include the most recently added portion since the remainder would have been checked previously.

Thus, a simpler graph matching algorithm is now being employed. The structure to be checked is first put into the appropriate format. This format is identical with the list notation representation of the structure except that hydrogen atoms must be included. The function ADDH(S) converts the structure S to the appropriate format. The function NEWCHECK causes this conversion to be made, and then compares the structure with each element of BADLIST to determine whether any forbidden substructure is present.

BADLIST is merely a list of forbidden substructures. Each element is the list notation representation of a structure, with the following alterations:

1) The substructures have no afferent links.

2) If any node can be one of several types of atoms, the list of atoms is put in place of the usual single atom name.

NEWCHECK examines BADLIST and extracts those structures whose apical node is the same as the apical node of the structure being checked.* Then the structure is searched for each of the appendant

*Because of this, BADLIST must contain several entries for some structures, one entry for each possible apical node. Thus the elements of BADLIST are not all in canonical form.

radicals of the forbidden substructure. If all are found, then the substructure is present, and NEWCHECK returns the value T.

The primary tool used in this process is the function COMPAR. COMPAR takes two arguments, S and L . Each argument is a list of radicals representing some or all of the efferent radicals of the two graphs being matched. The radicals on L come from the forbidden substructure. If not all elements of L are on S , then COMPAR returns NIL. If all elements of L are on S then COMPAR returns the dotted pair (T.R) where R is the remainder of list S after the elements of L have been removed.

This graph matching algorithm is far simpler than the one described in the appendix. It requires much less code (and consequently uses much less core memory) and should prove to be quite a bit more efficient for the types of problems encountered in structure generation.

## 6. SUMMARY AND DISCUSSION

The Dendral program is constantly being modified as new and better ways of doing things are conceived. The basic structure generating functions are written independently in such a way that new supervisory functions can be easily inserted into the system. It is hoped that the Dendral program will eventually be able to benefit from the user-on-line characteristics of the time-sharing system in order to "extract knowledge" from chemists and other users of the program.

One goal is to be able to give the program a real mass spectrum and have the program predict only a small number of structures which are most plausible. Considerable work must be done before this

goal can be realized. One problem is the determination of which composition was represented by the original substance. Another problem is "cleaning up" a real spectrum. A third problem is obtaining a prediction of the spectrum of a given structure for purposes of comparing with real data. The present spectrum predictor is only a crude, first-order attempt at generating the spectrum of a structure.

The process of generating the most plausible structure from a given composition can be improved by making use of a chemist's knowledge about commonly occurring substructures. Soon the program will contain a GOODLIST which can keep track of likely combinations of atoms. Then, during the initial consideration of the composition, the program can remove certain groups of atoms and replace them by a symbol representing the substructure. Then, using the arbitrary atom X in the function CHNOPSXVQW, the program can treat this substructure as an atomic unit and insure that all generated structures will contain the desired substructure.

Other proposed modifications to the program include the following items:

1) Revising the method of remembering past work so that the "dictionary of structures" requires less core storage space.

2) Putting the plausibility parameters within the reach of the user so that different schemes for rating plausible partitions may be compared.

3) Introducing more efficient "tree pruning" methods so that the search space for plausible structures will be made smaller.

4) Introducing "mood items" for BADLIST so that the program can interrogate the user as to what general class of compounds he expects. Certain structures will be added to or removed from BADLIST as a result of the user's "mood".

It should be evident that the Dendral program is constantly expanding and becoming more sophisticated. In evaluating the efforts performed up until now, the importance of the basic Dendral notation for providing unique, non-ambiguous, and algorithmic representation of chemical structures cannot be overstated. Not only has the Dendral notation allowed all the isomers of a chemical formula to be quickly, accurately, and completely generated; but also it has provided a base for studying a certain "inductive and enquiring" system.

# APPENDIX 1

## The Sussenguth Method of Graph Matching as Implemented in the Dendral Program

### 1. GRAPH MATCHING IN DENDRAL

The Dendral program may be made to generate all topologically possible structures from a chemical formula, restricted only by the valence limitations on the atoms. Many of the structures generated by the Dendral algorithm are not chemically meaningful because they contain certain impossible substructures. The forbidden substructures were few enough in number that they could be listed; and a graph matching algorithm was introduced to check each generated structure against the list of forbidden substructures. If a structure generated by Dendral is found to contain even one of the forbidden substructures, it is not acceptable output and the Dendral algorithm attempts to find the next higher structure which does not contain a forbidden substructure.

The graph matching algorithm that has been implemented to compare Dendral-generated structures against the BADLIST is essentially that presented in the PhD thesis of E. H. Sussenguth, Jr. at Harvard, 1964. This algorithm considers two graphs, made up of nodes (atoms) and connections (bonds), and compares sets of nodes with equal properties. Using set operations such as union and intersection, and making assignments of node correspondences where sufficient information is not

present for a unique determination, the algorithm avoids a time-consuming direct node-by-node comparison of the two graphs. This algorithm is especially efficient in determining when no match exists. The process of actually finding an isomorphism takes somewhat longer.

## 2. NOTATION FOR GRAPH MATCHING

The form of a graph G is a list of elements, each element representing a node of the graph.

Each node-element has three parts:

1) a node number- an integer. Node numbers must be unique. And each node number is equal to the position of that node-element in the list that forms the graph.

2) a node type- the name of the atom at that node (usually C, N, O, P, or S -- but sometimes an abbreviation for a whole structure that is to be treated as a unit - such as OH, NHR, COOH, etc.)

3) A list of connections- This list has as many elements as the node has bonds connected to it. Thus the length of this list cannot (theoretically) be greater than the valence of the node. (In practice this restriction is irrelevant.) Each connection is a two element list of the form ( CN CB ). CN is the node number of the connected node and CB denotes the type of connection:

CB=T (triple bond), CB=D (double bond), CB=S (single bond)

For example, the Dendral notation . C = O N.C. = C C would be represented by ((1 C(2 D)(3 S))(2 O(1 S))(3 N(1 S)(4 S))(4 C(3 S)(5 S)(6 D))(5 C(4 S))(6 C(4 D))) for purposes of graph matching.

Operations comprising the graph matching algorithm depend heavily on the use of node numbers, both alone and in list (sets). An isomorphism (answer) is a pair of lists of node numbers, one from each of the two structures being considered, and where nodes in corresponding positions are isomorphic.

A property for a graph is a list of node numbers headed by an atom denoting the value common to all the nodes. Various property values are used in matching the graphs. They are described briefly below:

1) Node value. All nodes with value C are grouped together and headed by the atom "C". (etc. for N, O, P, ..)

2) Branch value. All nodes adjacent to single bonds are grouped together and headed by "S". (similarly for D and T)

3) Gamma degree. The gamma set of a node n is the list of nodes reachable from n by a path of a specified length (i.e., by traversing a specified number of bonds). Thus, the gamma=1 set for a node n is a list of all nodes immediately adjacent (connected) to node n. The gamma=2 set for node n is the list of all nodes connected to the nodes in the gamma=1 set of node n. The gamma=m degree of node n is the number of nodes (length) of the gamma=m set for node n.

4) Quasi-order. The quasi-order of node n is the number of bonds (connections) that must be traversed before getting back to the given node n. Direct backtracking is forbidden.

Thus, the quasi-order for any node in a non-ringed or tree structure must be zero. (Quasi-order is not included in the current version of the Dendral program since ring structures are not included.)

5) Connectivity. Connectivity of a set of nodes is another set of nodes representing all nodes which can be reached by a path of given length from any one of the original nodes. Thus, connectivity of a set of nodes is the union of the gamma sets of all its nodes. Connectivity is usually found for paths of length 1 or 2, but it could be calculated for longer paths until redundant answers begin to be generated.

## 3. DESCRIPTION OF THE GRAPH MATCHING ALGORITHM

The computer program for the graph matching algorithm is written in the language LISP. The operation of the algorithm is based on comparing corresponding properties from each of two graphs in an attempt to find pairs of nodes with identical properties. During this pairing process, careful checks are made to determine whether otherwise similar nodes have contradictory properties. Such a situation indicates "no match", and the algorithm terminates immediately.

The algorithm makes heavy use of variables which are global to all LISP functions. These variables are certain lists which are set, changed, and translated by the major functions. Eight of these lists deserve special mention:

1) G -- one of the two graphs to be matched. This

variable is set as input to the algorithm and never changed.

2) GS -- the other graph to be matched. GS must be the larger of the two (in the sense of having more nodes).

3-4) -- L and LS are lists whose elements are sets of nodes with corresponding properties from G and GS respectively.

5-6) -- V and VS are edited versions of L and LS, being lists of sets of nodes of G and GS with corresponding properties. Non-informative sets have been removed, and pairs of nodes that are known to correspond are removed from the sets in which they appear.

7-8) -- K and KS are lists of corresponding (known) nodes for G and GS. The first element of K is the node corresponding to the first element of KS. If the length of these sets equals the number of nodes in graph G, then an isomorphism is determined. There may be more than one isomorphism, but the current version of the algorithm is satisfied with a single one.

There are two classes of functions making up the Dendral graph matching algorithm. The first class contains all functions that are an integral part of the pure subgraph matching. The second class of functions contains all those needed to use the graph matching from within Dendral. This class depends on the presence of both pure

subgraph matching and pure Dendral.

The important functions in the first class are ISOLISP (the supervisory function), SETFORM (which extracts properties of graphs), FF (a function which obtains families of corresponding nodes), PAR (the function which obtains correspondents of the nodes of the smaller graph), and NEWABLK and NEXTASSIGN (which provide for systematic assignment of correspondences in cases where previous work has failed to find a unique correspondent for some node).

The function CHECKMATCH is the link between Dendral and the graph matching algorithm. It converts a structure to notation suitable for graph matching and calls ISOLISP for each element of BADLIST which may be contained in the test structure.

BADLIST is a global variable which contains information about all of the "forbidden" chemical structures. Each element of BADLIST contains the following pieces of information about a forbidden structure:

1) A predicate which indicates whether (T) or not (NIL) the structure to be matched with this subgraph should be checked for terminal OH and NHR before matching.

2) A number indicating the number of atoms (nodes) in the subgraph.

3) The composition list for the subgraph.

4) The subgraph itself in the notation required by the graph matching algorithm.

The function ISOLISP is called only if the number of atoms in the subgraph is less than or equal to the number of atoms in the

test structure and if the composition of the subgraph is contained in the composition of the test structure. The value of CHECKMATCH is T if any forbidden subgraph is found in the test structure. Otherwise the value of CHECKMATCH is NIL.

4. PRIMARY FUNCTIONS FOR GRAPH MATCHING

ISOLISP - 2 arguments

G - a graph

GS - a graph

ISOLISP is the control function for the algorithm. It calls other functions which construct and use the lists L, LS, V, VS, K and KS. ISOLISP recognizes when an isomorphism has been found or denied and calls for assignments to be made if necessary.

The first action of ISOLISP is to set up lists L and LS from the properties of graphs G and GS by calling the function SETFORM. Next, the elements of L and LS are examined and non-redundant sets of nodes are placed on V and VS by the function FF which checks for possible contradictions in properties of G and GS. The function CONNEC places new sets on L and LS. These sets are obtained by applying the property of connectivity to all sets on V and VS. The function PAR obtains the set of correspondents for each node of G which is not on list K. These sets are added to V and VS by the function PAR.

If the sets V, VS, K and KS are unchanged after the functions FF and PAR have both been executed, then an isomorphism cannot be determined without making an assignment. The function NEWABLK causes a node assignment to be made and added to the lists K and KS. If no

assignment is possible, then a previous assignment must be contradicted. (If no previous assignment was made, then no isomorphism is possible). The variable called ASTACK contains a record of current assignments. NEXTASSIGN revises the most recent assignment if possible. Otherwise, the most recent assignment is discarded and the state of the system prior to that assignment is retrieved in order to revise the previous assignment.

FF-0 arguments

FF uses the lists L and LS to form families of corresponding nodes. Each pair of elements ($L_i$ , $LS_i$) is examined for useful information. Known corresponding nodes are removed (by the function REMK) from $L_i$ and $LS_i$. If exactly one node remains in each of $L_i$ and $LS_i$ , then this becomes a new known correspondence, the nodes are placed on K and KS and removed from the elements of V and VS (by the function REMNEWK). Otherwise, the elements $L_i$ and $LS_i$ are places on lists V and VS (by the function MERGESET), provided they do not contradict (no isomorphism) or duplicate any pair of elements ($V_j$ , $VS_j$).

After FF has considered all elements on L and LS, the length of lists K and KS determines the future action of the algorithm. If the number of correspondences (length of K) is equal to the number of nodes of graph G, an isomorphism is found and FF terminates with a value of 1. If the length of K is greater than the number of nodes of G then a contradiction must exist and no isomorphism will be possible. In such a case FF exits with a value of 0. Otherwise FF terminates with a value of 2 indicating that more work has to be done.

PAR - 0 arguments

PAR examines elements of V and VS and combines sets (using operations similar to union, intersection, and complement) to find single node correspondences. PAR returns 0 if an isomorphism is impossible, 1 if an isomorphism is found, and 2 if more work needs to be done.

In doing this, PAR takes each node of G and constructs the list of its possible correspondents by considering each pair of sets $V_j$ and $VS_j$. Initially the known correspondents of node n are all the non-known nodes of graph GS. If node n is in $V_j$ then $VS_j$ is intersected with the set of possible correspondents for node n. If n is not in $V_j$ then the complement of $VS_j$ is intersected with the set of possible correspondents for node n. If the set of correspondents has length 1 then node n becomes a known node and is added to list K, otherwise n is added to V and its set of correspondents is added to VS.

SETFORM - 0 arguments

SETFORM uses set generating (property generating) functions to set L and LS to lists of corresponding nodes of G and GS. The properties are: gamma degree = 1, gamma degree = 2, node value and branch value.

NEWABLK - 0 arguments

NEWABLK adds a new assignment block to ASTACK. An assignment block is a list of the form (XS X SS V VS K KS) where the correspondence (assignment) X:XS was made from set SS which is part of LIST VS. Values of V, VS, K, and KS are prior to assignment so they can be restored if the assignment fails. X is added to ASLIST, and

lists K, KS, V, and VS are all updated using the new assignment.

ASSIGNI - 1 argument

A - a number or NIL

ASSIGNI finds a possible correspondent of node X in set SS. (Both X and SS are global to ASSIGNI, being set within NEWABLK and NEXTASSIGN which are the functions which can call ASSIGNI.) If A is not NIL, then it is the last correspondent used for X, and the search for a new correspondent starts with the successor of A in list SS. The new correspondence X:XS is checked for validity against the sets of V and VS.

NEXTASSIGN - 0 arguments

NEXTASSIGN makes the next assignment in the current block. The current block is the first element of ASTACK, which has kept tract of all assignments made in the current attempt to locate an isomorphism.