AN ALGORITHM FOR THE CONSTRUCTION OF THE GRAPHS OF ORGANIC MOLECULES

by

HAROLD BROWN
LARRY MASINTER

STAN-CS-73-361
May 1973

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

An algorithm for the construction of the graphs of organic molecules

By Harold Brown and Larry Masinter

ABSTRACT:

A description and a formal proof of an efficient computer implemented algorithm for the construction of graphs is presented. This algorithm, which is part of a program for the automated analysis of organic compounds, constructs all of the non-isomorphic,connected multi-graphs based on a given degree sequence of nodes and which arise from a relatively small "catolog" of certain canonical graphs. For the graphs of the more common organic molecules, a catolog of most of the canonical graphs is known, and the algorithm can produce all of the distinct valence isomers of these organic molecules.

This work was supported in part by ARPA Contract SD-183 and NSF Grant GP-16793

An Algorithm for the Construction of the Graphs of Organic Molecules

By Harold Brown and Larry Masinter

I. Introduction. The usual problem in analytical organic chemistry is to determine the molecular structure of an unknown organic compound. The heuristic DENDRAL program for explaining empirical data [1,2] is a machine implemented program which applies artificial intelligence techniques to this problem of molecular structure determination. The primary input to this program is the mass spectrum of the unknown compound. Secondary inputs, if available, include the nuclear magnetic resonance spectrum and the elementary formula of the compound. The output of the program is a list consisting of one or more molecular graphs, each of which represents a heuristically plausible explanation of the given input. These graphs are rank ordered with respect to their relative plausibility scores. Central to this DENDRAL program are algorithms which generate all of the distinct valence isomers of a given set of atoms. It is these algorithms that we will describe in this paper.

In graph theoretical terms, the problem that we consider is the following: Given a degree sequence of nodes which corresponds to the valences of the atoms of a given organic compound, algorithmically construct a representative set of the distinct isomorphism classes of the connected, loop-free multi-graphs based on that degree sequence. (For brevity, such graphs will be referred to as elf-graphs.) Moreover, for pragmatic reasons, the algorithm should:

a) Be machine implementable in a reasonably efficient manner.

b) Be able to accept as additional input certain constraints arising from the chemical heuristics of the situation. For example, generate only those graphs which contain a certain substructure.

c) Generate no isomorphic graphs in the intermediate stages of the algorithm.

This last restriction is necessary because of the time and the storage limitations when using a machine.

In section II of the paper we define the concept of a superatom, and we briefly describe the subalgorithms used by the main algorithm. In section III, first a conceptual description and then a formal description of the main algorithm is given. Section IV contains the graph theoretical results on which the main algorithm is based, and in section V a formal proof of the completeness and the correctness of the main algorithm is nresented.

II. Terminology and Subalgorithms. The concept of a superatom is used throughout the paper. A superatom is a collection of atoms bonded together into a cyclic structure which possesses at least one unassigned valence, i.e., at least one free valence. Graph theoretically, a superatom is a connected, loop-free multi-graph with no isthmuses, i.e., with no edge whose deletion disconnects the graph, and with at least one unassigned valence. If a superatom A has k free valences, then in forming molecular structures which include A, A behaves almost like an atom of valence k. The difference in forming structures including A and in forming structures including an atom of valence k is the following: The k free valences on an atom of valence k are, as edge endpoints in a graph, indistinguishable, i.e., barring optical. isomerism, the free valences on

the atom implicitly admit as symmetry group the group S_k, the full permutation group on k objects. However, the k free valences on the superatom A usually are distinguishable from a symmetry view-point, and the free valences on A will, in general, admit only a subgroup of S_k. For example, the superatom in Figure 1 has three free valences, but the group of symmetries of these free valences is only $\{\binom{1\ 2\ 3}{1\ 2\ 3}, \binom{1\ 2\ 3}{3\ 2\ 1}\}$. Thus, associated with each superatom with k free valences is a subgroup of S_k, namely, the group of symmetries on its free valences.

By the term tree we will always mean a graph whose nodes can represent either atoms or superatoms and whose edges are all isthmuses. Also, throughout the paper, a collection of atoms given by their valences, i.e., a degree sequence, will be represented as an integral n-vector $V = (a_1, a_2, \ldots, a_n)$, where a_i is the number of atoms of valence i in the collection.

For a degree sequence $V = (a_1, a_2, \ldots, a_n)$, the unsaturation of V, U(V), is defined as $U(V) = 1/2(2 + \sum_{1}^{n} (i-2)a_i)$. If there exists at least one elf-graph based on V, i.e., a elf-graph with a_i nodes of valence i, then U(V) is precisely the minimal number of edges of any elf-graph based V which must be deleted in order to make the graph into a tree of atoms.

The description of the main algorithm makes use of the following subalgorithms:

a) A tree generator subalgorithm, TG. TG accepts as input

a list of valences corresponding to atoms and superatoms, each superatom with its associated free valence group, and outputs all non-isomorphic trees based on these atoms and superatoms. Such a tree generator algorithm is described in [3].

b) A labeler subalgorithm, LAB. LAB accepts as input an ordered list of m distinct symbols, a group H of permutations on these symbols, and a list of m not necessarily distinct labels. It outputs all non-equivalent, with respect to H, labelings of the given symbols with the given labels subject to a set, possibly empty, of constraints of the form that certain types of symbols must be labeled with certain types of labels. Such a labeling algorithm is described in [4].[1]

c) A "catolog", CAT, containing certain connected, loop-free, isthmus-free multi-graphs, for brevity, clfif-graphs, underlying, in a sense to be made explicit later, the graphs of organic molecules. A list of these underlying graphs is known for most of the common organic molecules, and consists of a relatively small number of graphs.

III. Main Algorithm. We give now a brief conceptual description of the main algorithm. The input to the algorithm is a degree sequence, i.e., an integral n-vector V.

1. Determine all distinct allowable partitions of V into atoms and superatom sets with assigned free valences. These partitions are based on the unsaturation of V.
2. For each superatom set, determine all the distinct allowable

allocations of the free valences to the atoms of the set.

3. For each such free valence allocation, determine recursively the allowable sets of atoms remaining after the deletion of the bivalent atoms and the "pruning" of any resulting loops. This recursion is done until: a) the remaining bivalent atoms in any clfif-graph based on the set must all be on edges, or b) one of two special cases is encountered,

4. For each such set of atoms, if condition (a) terminates the recursion, "look up" in the "catolog" all the clfif-graphs based on the non-bivalent atoms in the set and for each such graph label the edges with the bivalent atoms. If condition (b) terminates the recursion, directly 'write down" the allowable graphs.

5. For each such graph, recursively label the atoms with loops and the loops and edges with bivalent atoms.

6. For each graph so obtained, label the atoms with the free valences.

7. For each set of atoms and superatoms obtained as above, use the tree generator to construct all the non-isomorphic elf-graphs based on these atoms and superatoms.

The following formal description of the main algorithm is presented as a sequence of subalgorithms:

Definitions. Given $V = (a_1, a_2, \ldots, a_n)$, define:

a) $U(V) = (2 + \sum_{1}^{n}(i-2)a_i)/2.$

b) $TD(V) = \sum_{1}^{n} i a_i.$

c) $N(V) = \sum_{1}^{n} a_i$.

d) $P(V) = a_1$.

e) $M(V) = \max\{\, 1 \le i \le n \mid a_i \neq 0 \,\}$.

Note that $U(V) = (2 + TD(V) - 2N(V))/2$.

A. Superatom Partitioner (SAP).

Input: $V = (a_1, a_2, \ldots, a_n)$.

1. (Test) If U(V) is a non-negative integer and $2M(V) \le TD(V)$, continue; else STOP.
2. If $U(V) = 0$, go to TG.
3. Do 12 for each non-trivial summand $\overline{V}$ of V such that $P(V) = 0$ and $N(\overline{V}) \ge 2$.
4. Do 11 for $1 \le k \le \lfloor N(\overline{V})/2 \rfloor$.
5. Do 10 for each distinct k-term ordered partition of U(V), say $U(V) = u_1 + u_2 + \ldots + u_k$ where $1 \le u_1 \le u_2 \le \ldots \le u_k$.
6. Do 9 for each distinct set $\{ (V_i, u_i) \mid 1 \le i \le k \}$ where $\sum_{1}^{k} V_i = \overline{V}$ and $N(V_i) \ge 2$, $1 \le i \le k$.
7. Compute $FV_i = 2(U(V_i) - u_i)$, $1 \le i \le k$.
8. If $FV_i < \max\{1, 2M(V_i) - TD(V_i)\}$ for any $1 \le i \le k$, go to 6.
9. Put $\{ \overline{V}; (V_1, FV_1), \ldots, (V_k, FV_k) \}$ on the list GLSA(V).
10. Continue.
11. Continue.
12. Continue.
13. If $P(V) = 0$, put $\{ V; (V, 0) \}$ on the list GLSA(V).

B. Free Valence Partitioner (FVP).

(Input: $(0,v_2,\ldots,v_n)$, FV_i) where $N(V_i) \geq 2$,

$2(U(V_i) - u_i) = FV_i$, $1 \leq u_i \leq 1$, and

$FV_i \geq \max\{ 1, 2M(V_i) - TD(V_i) \}$.

1. Determine all sets of non-negative integers

$\{ f_{jt} \mid c_j = \max\{0, j + (FV_i - TD(V_i))/2\} \leq t \leq j-2,$

$j = 2,\ldots,n \}$ satisfying:

i) $\sum_{t=c_j}^{j-2} f_{jt} = v_j$, $j = 2,\ldots,n$.

ii) $\sum_{j=2}^{n} \sum_{t=c_j}^{j-2} t f_{jt} = FV_i$.

(Note that $f_{20} = v_2$).

2. For each such set, form

$W = W(V_i, FV_i, \{f_{jt}\}) = (0,w_2,\ldots,w_n)$ where

$w_s = \sum_{\substack{s \leq h \leq n \\ h \geq c_h + s}} f_{h(h-s)}$, and put W on the list $GLFV(V_i, FV_i)$.

C. Loop-Bivalent Partitioner (LBP).

(LBP is applied recursively until either $m_1(W) = m_0(W) = 0$

or $m_1(W) < m_0(W)$).

(Input: $W = (0,w_2,\ldots,w_n)$ where $N(W) \geq 2$, $U(W)$ is a

positive integer, and $2M(W) \leq TD(W)$.

1. Compute $m_1(W) = \min\{ w_2, \sum_{j=4}^{n} \lfloor (j-2)/2 \rfloor w_j \}$ and

$m_0(W) = \max\{ 0, w_2 + (2M(W) - TD(W))/2 \}$.

2. If $m_0(W) > m_1(W)$, go to Special Case.

3. If $m_0(W) = m_1(W) = 0$, go to Catolog.

4. Do 9 for $p = m_0(W),\ldots,m_1(W)$.

5. Determine all sets of non-negative integers

$\{ p_{jt} \mid 3 \leq j \leq n,\ 0 \leq t \leq \lfloor (j-2)/2 \rfloor \}$ satisfying:

i) $\sum_{t=0}^{\lfloor (j-2)/2 \rfloor} p_{jt} = w_j,\ 3 \leq j \leq n.$

(Note that $p_{30} = w_3$).

ii) $\sum_{j=3}^{n} \sum_{t=0}^{\lfloor (j-2)/2 \rfloor} t p_{jt} = p.$

6. Do 8 for each such set $\{p_{jt}\}$.

7. Form $Y = Y(W, p, \{p_{jt}\}) = (0, y_2, \ldots, y_n)$ where

$$y_s = \sum_{\substack{m=s \\ \text{by } 2}}^{n} p_{m,(m-s)/2}.$$ (Note that $p_{20} = 0$).

8. If $2M(Y) \leq TD(Y)$, put Y on the list $\tilde{Y}GLL_B(W,p)$.

9. Continue.

D. Special Case (SC).

(Here $m_1(W) < m_0(W)$).

1. If $m_1(W) = 0$, put the graph consisting of the ring with N(W) bivalent nodes and its associated node symmetry group on the list $\tilde{Y}GLGPH(W)$ and go to Loop Labeler; else go to 2.

2. If $m_1(W) = s > 0$, put the graph consisting of one node of degree 2(s+1) with s+1 loops attached and its associated edge-loop symmetry group on the list $\tilde{Y}GLGPH(W)$ and go to Bivalent Labeler.

E. Catolog (CAT).

Input: $Y = (0, y_2, \ldots, y_n)$ where $m_1(Y) = m_0(Y) = 0$.

1. If $y_2 \neq 0$, "look-up" all non-isomorphic clfif- trivalent graphs G with y_3 nodes and their associated edge symmetry groups, EGRP(G); else go to 6.
2. Do 4 for each such graph G.
3. Determine all distinct ordered partitions of y_2 into TD(G)/2 non-negative integral terms.
4. For each such partition, say $b_1 + \ldots + b_{TD(G)/2} = y_2$, use the general labeling subalgorithm to determine all distinct, with respect to EGRP(G), labelings of the edges of G with the labels $b_1, \ldots, b_{TD(G)/2}$, replace the label b_i on the labeled graphs by b_i bivalent nodes, and place the resultant graphs and their node symmetry groups on the list GLGPH(Y).
5. Go to Loop-Bivalent Labeler.
6. If $y_2 = 0$, "look-up" all non-isomorphic clfif- graphs based on Y, and put these graphs and their associated node symmetry groups, on the list GLGPH(Y).
7. Go to Loop-Bivalent Labeler.

F. Loop-Bivalent Labeler (LBL).

(LBL is applied recursively to each graph G on GLGPH(Y) up the path of vectors from which G arose until reaching the $W = W(V_i, FV_i, f_{jt})$ on GLFV(V_i, FV_i) heading the path).

Loop Labeler (LL).

Input: A elf-graph G , its node group NGRP(G) and the vector Y = Y(W,p,{p_{jt}}) on GLL(W,p) from which G arose.

1. Compute $q_t = \sum_{j=3}^{n} p_{jt}$, t = 0,...,$\lfloor (n-2)/2 \rfloor$.
2. Determine all distinct, with respect to NGRP(G), labelings of the nodes of G with q_t-t's subject to the constraint that p_{jt}-t's must go on nodes of valence j-2t.
3. For each such labeling, replace the label t by t loops.

Bivalent Labeler (BL).

Input: A connected graph H with p loops, the edge-loop group of H, ELGRP(H), and w_2 the number of bivalent nodes in the vector W from which H arose.

1. Compute h = p + number of (non-loop) edges in H.
2. Do 4 for each ordered partition of w_2 into h non-negative terms, say $w_2 = b_1 + \ldots + b_h$, $b1 \leq \ldots \leq b_h$, such that at least p of the b_i's are non-zero.
3. Determine all distinct, with respect to ELGRP(H), labelings of the edges ahd loops of H with the labels $\{ b_i \mid 1 < i \leq h \}$ subject to the constraint that each loop gets a positive label.
4. For each such labeling, replace the label b_i by b_i bivalent nodes, $1 \leq i \leq h$, and place the resultant graph and its node group on the list VGLGPH(W).

G. Free Valence Labeler (FVL).

Input: A graph G , its node group NGRP(G), and the vector $w = W(V_i, FV_i, \{f_{jt}\})$ from which G arose.

1. Compute $f_t = \sum_{j=2}^{n} f_{jt}$, $t = 0,\ldots,n + (FV_i - TD(V_i))/2$.
2. Determine all distinct, with respect to NGRP(G), labelings of the nodes of G with f_t-t's subject to the constraint that f_{jt}-t's must go on nodes of valence j-t.
3. For each such labeling, replace each label t by t free valences and put the resultant graph and its free valence symmetry group on the list GLGPH(V_i,FV_i).

H. Input for Tree Generator.

For $\{ \overline{V}: (V_1, FV_1), \ldots, (V_k, FV_k) \}$ on the list GLSA(V), input to TG:

a) V-V. (The atoms).

b) A k-tuple of graphs $(G_1, \ldots G_k)$ and their associated free valence groups where G_i is on the list GLGPH(V_i, FV_i), $1 \leq i < k$. (The superatoms).

IV. Theoretical Results. We now present the graph-theoretical results on which the main algorithm is based. Even though some of these results are known, we give proofs for the sake of clarity and completeness. Throughout this section, $V = (a_1,\ldots,a_n)$ denotes an integral n-vector.

Lemma 1. If there is at least one elf-graph G based on $V = (a_1,\ldots, a_n)$, then U(V) is a non-negative integer.

Proof. A spanning tree of G has N(V)-1 edges, and G has TD(V)/2 edges. Hence U(V) = TD(V)/2 - (N(V)-1) is a non-negative integer.

Lemma 2. If $2M(V) \leq TD(V)$ and U(V) is a non-negative integer, then there is at least one elf-graph based on V.

Proof. Let m = M(V), the maximum valence of V. Given non-negative integers, $b_2, \ldots, b_m$, $b_m \neq 0$, let $bl = L(b_2,\ldots,b_m)$ be the least non-negative integer such that there is at least one elf-graph based on $B = (b_1, b_2,\ldots,b_m)$. Since for sufficiently large b_1, e.g., $b_1 = \sum_{2}^{m} ib_i - 2(\sum_{2}^{m} b_i - 1)$, such a graph exists, $L(b_2,\ldots,b_m)$ is well-defined.

Assume $bl = L(b_2,\ldots,b_m) > 2$, and let G be a elf-graph based on B. If there where two distinct nodes Nl and N_2 of G both of which had adjacent nodes of valence 1, we could delete these valence 1 nodes and add the edge (N_1,N_2) to G obtaining a elf-graph with a lesser number of nodes of valence 1. Thus all valence 1 nodes of G must be adjacent to the same node, say N. If there was a node N_1 adjacent to N and a node $N_2 \neq N$ adjacent to N_1, we could delete the edge (N_1,N_2) and two of the valence 1 nodes and add the edges (N_1,N) and (N,N_2) obtaining a elf-graph with a lesser number of valence 1 nodes. (See Figure 2). Thus all nodes adjacent to N are adjacent only to N, and since G is connected, all nodes must be adjacent to N. (See Figure 3). Hence, N must be the unique node of valence m, i.e., $b_m = 1$, and $b_1 = m - \sum_{i=2}^{m-1} ib_i$.

Now assume the lemma is false, i.e., $2m \leq TD(V)$, U(V) is a non-negative integer and there is no elf-graph based on $V = (a_1, \ldots, a_n)$. Then $a_1 < L(a_2,\ldots,a_m)$. If $L(a_2,\ldots,a_m) \geq 2$, then $a_1 < m - \sum_{i=2}^{m-1} ia_i$, and $TD(V) < 2m$, a contradiction. If $L(a_2,\ldots,a_m) = 0$, then $a_1 < 0$, a contradiction. If $L(a_2,\ldots,a_m) = 1$, then $a_1 = 0$. Since a

elf-graph based on $(1,a_2,\ldots,a_m)$ does exist, by Lemma 1, $c = 1 + \sum_{i=2}^{m} (i-2)a_i$ is even. Hence $U(V) = (c+1)/2$ is not integral, a contradiction.

Theorem 1. There exists at least one elf-graph based on V if and only if $2M(V) \leq TD(V)$ and U(V) is a non-negative integer.

Proof. The sufficiency is just Lemma 2. For the necessity, let $m = M(V)$. By Lemma 1, U(V) is a non-negative integer. Assume that $2m > TD(V)$. Then $m > a_1 + 2a_2 + \ldots + (m-1)a_{m-1}$. Hence $a_m = 1$ and $m > \sum_{i=1}^{m-1} ia_i$. This contradicts the loop-free assumption.

Corollary 1. V determines a loop-free tree if and only if $U(V) = 0$. Moreover, in this case every graph based on V is a loop-free tree.

Theorem 2. There is a clfif-graph based on V if and only if U(V) is a positive integer and $2M(V) \leq TD(V)$.

Proof. If there is a clfif-graph based on V, then by Theorem 1 and Corollary 1, U(V) is a positive integer and $2M(V) \leq TD(V)$.

Conversely, by Theorem 1 there exist elf-graphs based on V. Assume that every such graph has at least one isthmus. Let G be a graph based on V with a minimal number of isthmuses, i.e., a "least criminal." We will show that G can be modified to a elf-graph based on V with a lesser number of isthmuses, a contradiction to our assumption.

Let the edge (A,B) be an isthmus of G, and let X and Y be the connected components of $G\setminus(A,B)$. Since $U(V) > 0$, not both X and Y are trees. Say X is not a tree and A is in X. Let C be an elementary circuit in X, and let C be a node on C of minimal path length from A, where we consider here a multiple edge as a circuit. (Note that

we may have A = C). By the choice of C, there is a path from A to C which is edge-disjoint from C. Let D be a node on C adjacent to C. (See Figure 4).

Case 1. Y is a tree.

Since G is loop-free, there must be a valence 1 node N in Y. Say N is attached to the node E of Y. Then, delete (C,D), add (D,E), and move N from E to C. (See Figure 5).

Case 2. Y is not a tree.

Let E be a elementary circuit in Y, E a node of E of minimal path length from B and F a node of E adjacent to E. Then, delete (C,D) and (E,F), and add (D,F) and (C,E). (See Figure 6).

In both cases, the resulting graph is a elf-graph based on V and has at least one less isthmus than G. Hence our assumption was false, and there is at least one clfif-graph based on V.

Lemma 3. Let $\bar{V}$ and FV_i, i = 1, ... k, be as in Superatom Partitioner. Then $\sum_{i=1}^{k} FV_i = 2(k - U(V-\bar{V}))$.

Lemma 4. In a loop-free graph based on V, there are $TD(V)/2 - a_1$ edges between non-univalent nodes.

Theorem 3. Let $V = (0,a_2,\ldots,a_n)$, $N(V) \geq 2$, U(V) a positive integer, and $2M(V) \leq TD(V)$. Let $m_0(V) = \max\{0, (a_2 + (2M(V)-TD(V))/2)\}$ and $m_1(V) = \min\{ a_2, \sum_{i=4}^{n} \lfloor (i-2)/2 \rfloor a_i \}$. Then, $m_0(V) \leq m_1(V)$ except in precisely the following two cases:

a) $m_1(V) = 0$ and $m_0(V) = 2$.

b) $m_1(V) = s > 0$ and $m_0(V) = s + 1$.

iioreover, case (a) holds if and only if $a_j = 0$, $j > 3$, and case (b) holds if and only if $a_{2(s+1)} = 1$ and $a_j = 0$, $j \geq 3$, $j \neq 2(s+1)$.

Proof. Let $m_0(V) > ml(V)$. If $ml(V) = a_2$, then $2M(V) - TD(V) > 0$, a contradiction. Hence $ml(V) = \sum_{i=4}^{n} \lfloor (i-2)/2 \rfloor a_i$. Assume $ml(v) = 0$. Then $a_j = 0$, $j > 3$. By Theorem 1, a elf-graph based on V exists. Hence a_3 is even. From $0 < m_0(V) = (2M(V) - 3a_3)/2$, it follows that a3 = 0 and $m_0(V) = 2$. The converse is immediate. Assume that $m_1(V) = s > 0$. Then, $a_2 + (2M(V) - TD(V))/2 > s > 0$, and $M(V) > 2s + 3a_3 + \ldots t\ (M(V)-1)a_{M(V)-1} \geq 2s$. Hence $a_{M(V)} = 1$. Also, $s = ml(V) = \sum_{i=4}^{M(V)} \lfloor (i-2)/2 \rfloor a_i$ implies that $2s \geq M(V) - 3$. Thus, $2s + 3 \geq M(V) > 2s$. This implies that $a_i = 0$, $3 \leq i < M(V)$. Since a elf-graph based on V exists, M(V) must be even. Thus $M(V) = 2(s+1)$. A direct computation gives that $m_0(V) = s + 1$. The converse follows directly from $2M(V) \leq TD(V)$.

Lemma 5. Let V, $m_0(V)$, and $m_1(V)$ be as in Theorem 3. Assume that $m_0(V) = m_1(V) = 0$ and $a_2 \neq 0$. Then $a_3 \neq 0$ and is even 'and $a_1 = 0$ for $i > 3$.

Proof. $a_2 \neq 0$ implies that $m_1(V) = \sum_{i=4}^{n} \lfloor (i-2)/2 \rfloor a_{.1} = 0$. Thus $a_{.1} = 0$, $i > 3$. Now $m,(V) = 0$ implies that $2M(V) - 3a_3 \leq 0$. Hence $a_3 \neq 0$. Since a elf-graph based on V exists, a_3 must be even.

V. Proof of Main Algorithm.

A. Every elf-graph based on V occurs. Let G be a elf-graph based on V. We will show that the algorithm produces a graph isomorphic to G.

Since G exists, by Theorem 1, U(V) is a non-negative integer and $2M(V) \leq TD(V)$. If u(v) = 0, the tree generator produces a graph isomorphic to G. Assume U(V) > 0. Let B be the set of isthmuses in G, and let $G_1, \ldots G_k$ be those connected components

of $G\backslash B$ having at least two nodes. Since U(V) > 0, G is not a tree and $k \geq 1$. Also, no G_i has univalent nodes. Let $V_i = (0, a_2^i, \ldots, a_n^i)$, where a_j^i is the number of valence j nodes of G_i, and let FV_i be the number of elements of B connected to G_i in G, $1 \leq i \leq k$. Set $\overline{V} = \sum_{i=1}^{k} V_i$. Note that k, V_i, FV_i and $\overline{V}$ depend only on the isomorphism class of G. Now $P(\overline{V}) = 0$ and $N(\overline{V}) \geq 2$. Thus $\overline{V}$ is an allowable summand of of V in SAP. Also, $N(V_i) \geq 2$, $1 \leq i \leq k$, and $1 \leq k \leq \lfloor N(\overline{V})/2 \rfloor$. set $u_i = (2U(V_i) - FV_i)/2$. Let $\hat{G}_i$ be the graph obtained from G_i by replacing in G each of the FV_i connected components of G/B connected to G_i by a univalent node. Then $\hat{G}_i$ is a elf-graph which is not a tree. Let $\hat{V}_i$ be the n-vector associated with $\hat{G}_i$. Then $U(\hat{V}_i) = u_i$, and u_i is a positive integer. Moreover, by Lemma 3, $\sum_{A=1}^{k} u_i = U(V)$. Since $\hat{G}_i$ is a elf-graph, by Theorem 1, $2M(\hat{V}_i) = 2m(V_i) \leq TD(\hat{V}_i) = TD(V_i) + FV_i$. Thus $FV_i \geq 2M(V_i) - TD(V_i)$, and, after suitable reindexing, $\{ \overline{V};\ (V_1, FV_1), \ldots (V_k, FV_k) \}$ is on the list GLSA(V).

For a fixed i, $1 \leq i \leq k$, let f_{jt} be the number of valence j nodes of $\hat{G}_i$ with t univalent nodes attached, $2 \leq j \leq n$, $0 \leq t \leq j$. Since $\hat{G}_i$ is a elf-graph and has isthmuses only to univalent nodes, $f_{jt} = 0$ for $t > j-2$. Also, by Lemma 4, $\hat{G}_i$ has $(TD(V_i) - FV_i)/2$ edges between non-uivalent nodes. Hence, any node of valence j in $\hat{G}_i$ has at least $c_j = \max\{ 0, j + (FV_i - TD(V_i))/2 \}$ univalent nodes attached, i.e., $f_{jt} = 0$ for $t < c_j$. The set $\{ f_{jt} \mid 2 \leq j \leq n, c_1 \leq t \leq j-2 \}$ satisfies the conditions of FVP. Hence, $W(V_i, FV_i, \{f_{jt}\})$ is on the list GLFV(V_i, FV_i) and depends only on the isomorphism class of G_i.

Let $\tilde{G}_i$ be the graph obtained from $\hat{G}_i$ by deleting the FV_i univalent nodes and their connecting edges. Then $W = W(V_i, FV_i, \{f_{jt}\}) = (0, w_2, \ldots, w_n)$ is the n-vector associated with $\tilde{G}_1$, and $\tilde{G}_i$ is a cifif-graph. Assume that $m_1(W) \leq m_1(W)$ where m_0 and m_1 are as in LP. By Theorem 3, $\tilde{G}_i$ is not one of the special case graphs, and hence deleting the bivalent nodes of $\tilde{G}_i$ leaves a connected, isthmus-fret graph $\bar{G}_i$ with at least two nodes and say t_i loops. Since $\bar{G}_i$ is isthmus-free, each node of $\bar{G}_i$ must have at least two non-loop edges attached. Hence $t_i \leq \min\{ w_2, \sum_{i=4}^{n} \lfloor (i-2)/2 \rfloor w_i \} = m_1(W)$. Let G_i^* be the graph obtained from $\bar{G}_i$ by deleting the t_i loops. G_i^* is a clfif-graph. Let Y be the n-vector associated with G_i^*. Then $M(W) \leq M(Y) + 2t_i$. Hence, by Theorem 1, $2M(W) - 4t_i \leq 2M(Y) \leq TD(Y) = TD(W) - 2w_2 - 2t_i$, and $w_2 + (2M(W) - TD(W))/2 \leq t_i$. Thus $t_i \geq m_0(W)$. Let p_{jt} be the number of valence j nodes of $\bar{G}_i$ with t loops, $4 \leq j \leq n$, $0 \leq t \leq \lfloor (j-2)/2 \rfloor$. Then $\{p_{jt}\}$ satisfies the conditions of LP and Y is precisely the n-vector $Y(W, t_i, \{p_{jt}\})$. Since G_i^* is a elf-graph, $2M(Y) \leq TD(Y)$ and Y is on the list $GLL(W, t_i)$. Moreover, Y depends only on the isoxorphism class of $\tilde{G}_1$.

Recursive application of the above process yields a sequence of graphs $G_{i0}^* = \tilde{G}_i, G_{i1}^* = G_i^*, G_{i2}^*, \ldots, G_{1s_1}^*$, and associated n-vectors $Y_0^i = W(V_i, FV_i, \{f_{jt}\}), Y_1^i = Y(W, t_i, \{p_{jt}\}), Y_2^i, \ldots, Y_{s_i}^i$, where Y_j^i is on $GLL(Y_{j-1}^i, p_{j-1})$ for some allowable p_{j-1} and either (i) $m_0(Y_{s_i}^i) = m_1(Y_{s_i}^i) = 0$, or (ii) $m_0(Y_{s_i}^i) > m_1(Y_{s_i}^i)$. Moreover, Y_j^i depends only on the isomorphism class of $G_{i,j-1}^*$. Assume (i) holds

for $Y = Y^i_{s_i} = (0, y_2, \ldots, y_n)$. Now by Lemma 5, Y has only bivalent and trivalent nodes or no bivalent nodes. In the first case deleting the bivalent nodes of $G^*_{is_i}$ leaves a proper clfif, trivalent graph. In the second case $G^*_{is_i}$ is a graph with no bivalent nodes. In either case, a graph $G^{*'}_{is_1}$ isomorphic to $G^*_{is_i}$ is produced by CAT. If (ii) holds, by Theorem 3, $G^*_{is_i}$ is either the ring with y_2 bivalent nodes or $G^*_{is_i}$ with its bivalent nodes deleted is a single node of valence $2m_0(Y)$ with $m,(Y)$ loops attached, and a graph $G^{*'}_{is_i}$ isomorphic to $G^*_{is_i}$ is produced by BL. Hence we have that a graph $G^{*'}_{is_i}$ isomorphic to $G^*_{is_i}$ and the vector $Y^i_{s_i-1}$ are produced in the algorithm as input to LLBL. Since $G^*_{is_i-1}$ induces in the obvious manner a loop and/or bivalent node labeling of $G^{*'}_{is_i}$ arising from $Y^i_{s_i-1}$, $G^*_{is_i-1}$ is isomorphic to one of the graphs produced by LLBL with input $(G^{*'}_{is_i}, Y^i_{s_i-1})$. Recursively applying LLBL up the sequence $\{Y^i_j\}$, we obtain a graph G'_i isomorphic to G_i with associated n-vector $W_i = W(V_i, FV_i, \{f_{jt}\})$ in the output, i = 1, ... k. With the input (G'_i, W_i), FVL produces a graph G'_i isomorphic to G_i. Hence, $(V-\bar{V}, G'_1, \ldots, G'_k)$ is among the inputs to TG arising from V, and with this input TG produces a graph isomorphic to G.

B. <u>The algorithm produces no redundances</u>. Let G and Hbetwo graphs produced by the algorithm with input V. We will show that G and H are not isomorphic, i.e., $G \not\cong H$.

Assume $G \cong H$. TG is irredundant. Hence if G and H arise from the input to TG, $(V-v, G_1, \ldots, G_k)$ and $(V-\bar{V}', H_1, \ldots, H_{k'})$, respectively, where the G_i and H_j are superatoms with attached free valences, we must have that $\bar{V}' = \bar{V}$, k = k', and

the H_j can be so indexed such that $H_j \cong G_j$, $1 \leq j \leq k$. Hence, H_i and G_i both must have the same free valence FV_i, and G and H must *arise* from the same SAP of V, say $\{\overline{V}; (T_1, FV_1), \ldots (Tk, FV_k)\}$. Now since the labeler is irredundant, so is FVL. Hence from $H_i \cong G_i$, it follows that the graphs $\widetilde{G}_i$ and $\widetilde{H}_i$ defined as in (A) must be isomorphic. Also, $\widetilde{G}_i$ and $\widetilde{H}_i$ must arise from the same FVP of (T_i, FV_i), say $W_i = W(T_i, FV_i, \{f_{jt}\})$, $1 \leq i \leq k$. *Since* BL is irredundant, it follows that the graphs $\overline{G}_i$ and $\overline{H}_i$ defined as in (A) must be isomorphic. Similarly, since LL is irredundant, the graphs G_i^* and H_i^* must be isomorphic. Moreover, G_i^* and H_i^* must arise from the same LP of W_i, say $Y_i = Y(W_i, p_i, \{p_{jt}\})$, $1 \leq i \leq k$. Applying the above argument recur sively to G_i^* and H_i^*, we have that the CAT "look-up" (or SC determination) graphs, say G_i' and H_i', respectively, must be isomorphic and that the sequence of loop partitions leading from Y_i to the partitions determining G_i' and H_i' must be identical. Now $G_i' \cong H_i'$ if and only if G_i' and H_i' are equal. Thus $G_i' = H_i'$, $1 \leq i \leq k$.

Hence we have that if $G \cong H$, they arise from the identical sequence of vector inputs to the various labelers and TG, and that $G_i' = H_i'$, $1 \leq i \leq k$. Since for a given input, the labelers produce only non-isomorphic graphs, we must have, successively, that $G_i^* = H_i^*$, $\overline{G}_i = \overline{H}_i$, $G_i = H_i$, and $G_i = H_i$, $1 \leq i \leq k$. Hence if $G \cong H$, they both arise from the same input to TG. But for a given input, TG produces only non-isomorphic graphs, a contradiction,

C. The partitioners produce only allowable partitions of V. We will now show that every sequence of partitions of V produced by the algorithm yields at least one elf-graph based on V.

Let $Y = Y(W, p, \{p_{jt}\})$ be a member of the list $\tilde{Y}GLL(W, p)$. By direct computation, $U(Y) = U(W) - p$. Also, $p \leq \sum_{i=4}^{n} (i-2)w_i/2 < U(W)$. Thus $U(Y)$ is a positive integer. By construction, $2M(Y) \leq TD(Y)$. Since $m_1(W) \geq m_0(W)$, it follows from Theorem 3 that $N(Y) \geq 2$. Hence by Theorem 2, there is a clfif-graph based on Y. In the case $m_1(W) < m_0(W)$, Theorem 3 assures 'us that a clfif-graph based on W exists. Thus LP produces only allowable output.

Let $W = W(V_i, FV_i, \{f_{jt}\})$ be a member of the list $\tilde{Y}GLVF(V_1, FV_1)$. By direct computation, $U(W) = U(V_i) - FV_i/2 = u_i$, a positive integer. Also, $TD(W) = TD(V_i) - FV_i$. Now $M(W) = \max\{ j-t \mid 3 \leq j \leq n, c_j \leq t \leq j-2, f_{jt} \neq 0 \}$, and $j-t \leq (TD(V_i) - FV_i)/2$. Hence, $2M(W) \leq TD(V_i) - FV_i = TD(W)$. By Theorem 2 there is a clfif-graph based on W, and FVP produces only allowable output.

Let $\{\bar{V}; (V_1, FV_1), \ldots, (V_k, FV_k)\}$ be a member of the list $\tilde{Y}GLSA(V)$, say $V_i = (0, a_2^i, \ldots a_n^i)$, $1 \leq i \leq k$. Let $H_i = (FV_i, a_2^i, \ldots a_n^i)$. Now $2U(H_i) = 2U(V_i) - FV_i = 2u_i$, and $U(H_i)$ is a positive integer. Also, $P(H_i) = FV_i \geq 1$, and $TD(H_i) = TD(V_i) + FV_i \geq 2M(V_i) = 2M(H_i)$. Thus, by Theorem 2, there is a clfif-graph based on H_i, $1 \leq i \leq k$. Hence, at least one set, $G_1, \ldots, G_k$, of superatoms based on the given partition of V exists. If $\hat{V}$ is the n-vector corresponding to $v - \bar{V}$ and the free valences of the G_i, then a direct computation

using Lemma 3 gives that $U(V) = 0$. By Corollary 1, a loop-free tree based on $V - \overline{V}$ and the G_i exists. Thus the sequence of psrtions of V does yield at least one elf-graph based on V.

VI. Acknowledgements. The DENDRAL concept and its applications to organic chemistry were originally conceived by Professor Joshua Lederberg. The authors thank Professor Lederberg for his guidence and his critical suggestions which have made this work possible.

Computer Science Department
Stanford University
Stanford, California 94305

REFERENCES

1. E. A. Feigenbaum, B. G. Buchanan and J. Lederberg, *On* generality and problem solving: A case study using the DENDRAL program, in: Machine Intelligence 6 (Edinberg University Press,'Edinberg, 1971) 165-190.

2. B. G. Buchanan, G. L. Sutherland and E. A. Feigenbaum, A program for generating explanatory hypotheses in organic chemistry, in: Machine Intelligence 4 (Edinberg University Press, Edinberg, 1969) 209-254.

3. J. Lederburg, et. al., A tree generation algorithm, To appear.

4. H. Brown, L. Hjelmeland and L. Masinter, Constructive graph labeling using double cosets, To appear, Discrete Math.

FOOTNOTES.

1. More formally, if X is an m-element set and H is a group of permutations on X, then a labeling of X with n_1 labels a_1, n_2 labels a_2, ..., n_k labels a_k, $n_1+n_2+\ldots+n_k = m$, is a mapping $\psi: X \to \{a_1, a_2, \ldots, a_k\}$ such that $|\psi^{-1}(a_i)| = n_i$. Two such mappings ψ_1 and ψ_2 are equivalent with respect to H if there is an $\eta \in H$ such that $\psi_1 = \psi_2\eta$.

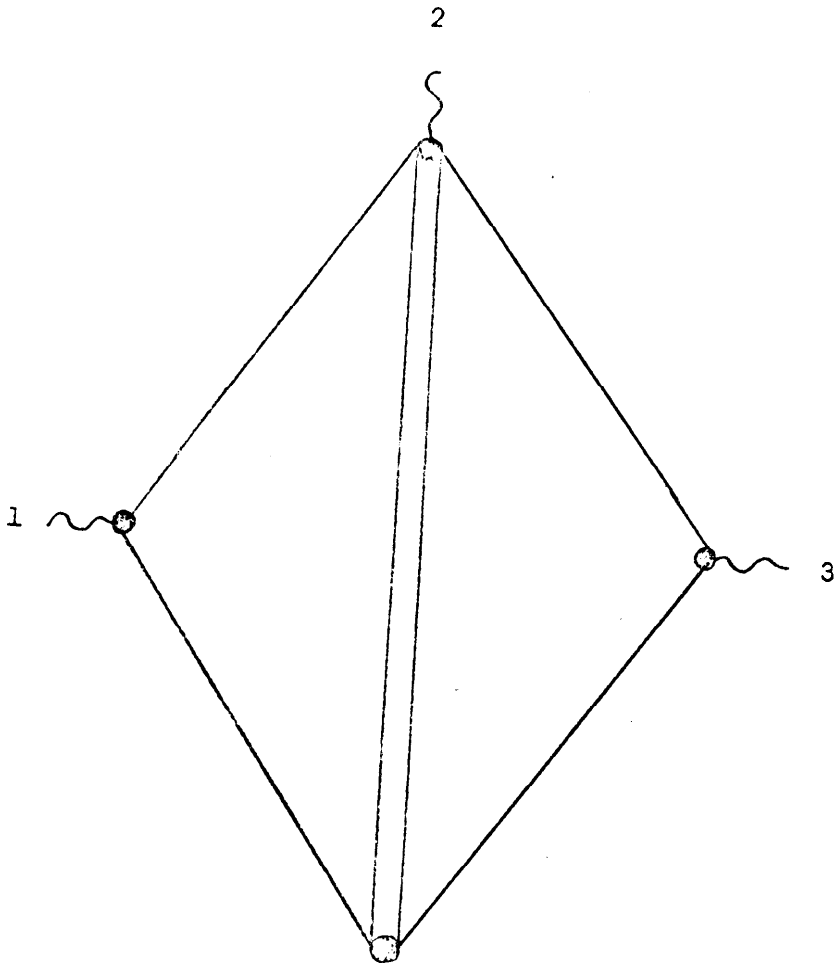

Figure 1.

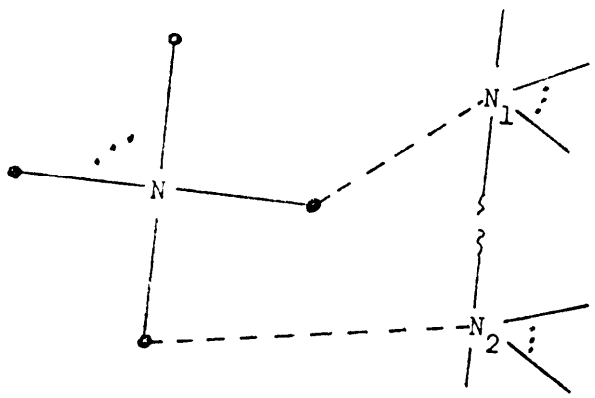

Figure 2.

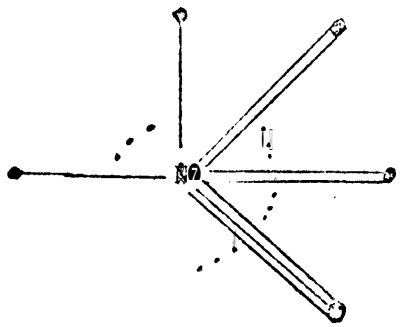

Figure 3.

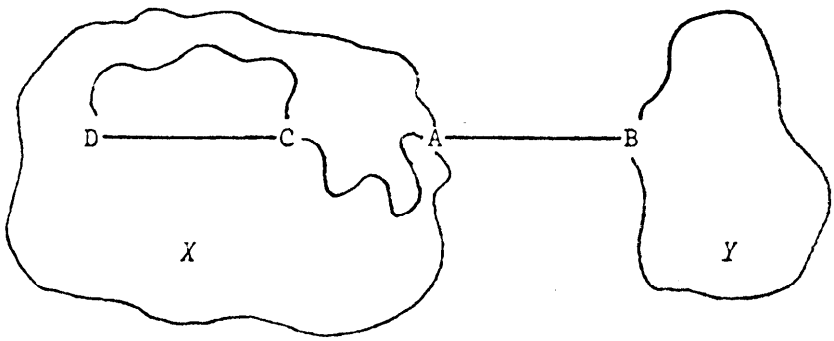

Figure 4.

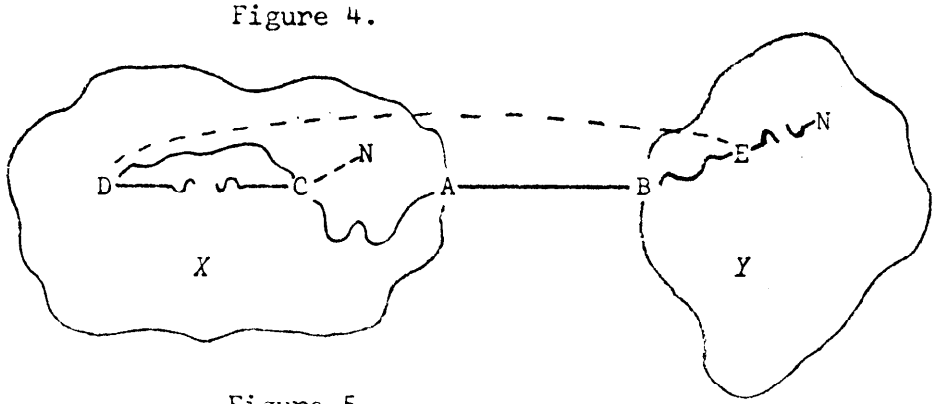

Figure 5.

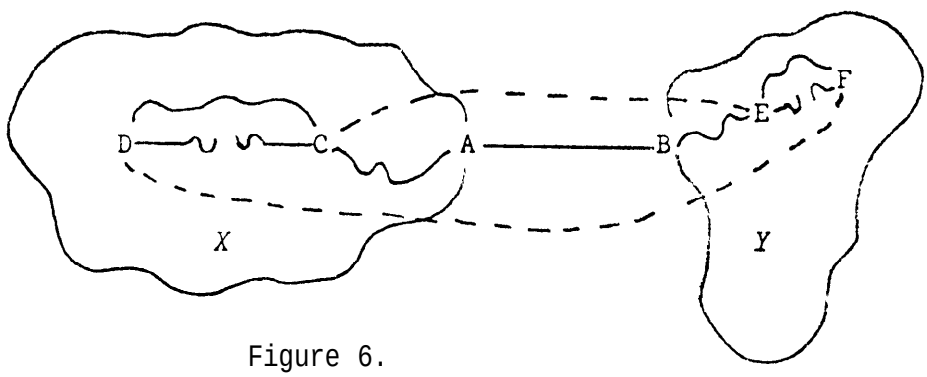

Figure 6.