MOLECULAR STRUCTURE ELUCIDATION III

by

Harold Brown

STAN-CS-74-469
DECEMBER 1974

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

MOLECULAR STRUCTURE ELUCIDATION III

by

Harold Brown

Abstract. A computer implemented algorithm to solve the following graph theoretical problem is presented: given the empirical formula for a molecule and one or more non-overlapping substructural fragments of the molecule, determine all the distinct molecular structures based on the formula and containing the fragments. That is, given a degree sequence of labeled nodes and one or more connected multigraphs, determine a representative set of the isomorphism classes of the connected multigraphs based on the degree sequence and containing the given multigraphs as non-overlapping subgraphs.

This work was supported in part by NIH Grant RR612-05A1 and NSF Grant DCR74-12646.

MOLECULAR STRUCTURE ELUCIDATION III

1. Introduction. This paper is the third in a sequence of papers on the derivation of combinatorial algorithms necessary for the development of a package of computer programs designed to assist the analytic chemist in determining the topological structure of organic molecules.[1) The first paper [1] described an algorithm for labeling the nodes or edges of a graph, and the second paper [2] described an algorithm for determinfng all the distinct graphs based on a given degree sequence of nodes. The relevance of these algorithms to structure elucidation problems in analytic chemistry is discussed in [3].[2)

The present paper addresses itself to a frequently encountered probiem in analytic chemistry. Namely, by applying spectroscopic measuring devices and various laboratory techniques to an unknown organic compound, the chemist can often determine the molecular formula of the compound as well as the topological structure of several fragments of this molecule, at least up to some unspecified bonds.[3) What is desired then is a complete and irredundant,

1) Throughout, we view a chemical molecule as a connected graph whose (labeled) nodes represent the atoms in the molecule and whose edges represent bonds, i.e., as the Kekule diagram of the molecule.

2) [3] also contains numerous references to articles describing specific chemical applications of these algorithms.

3) These laboratory techniques usually yield much more information about the unknown molecule, e.g., excluded fragments and multiple bonding patterns. Integration of this other information into our program package will be described in later papers.

i.e., nonisomorphic, set of the topological structures based *on* the molecular formula and containing the known fragments. In some cases these known fragments may overlap, i.e., have atoms in common. However, we will consider here only the case in which the fragments are assumed to be disjoint.

2. Problem Formulation. In order to formulate this molecular structure problem in precise, graph theoretical terms, we make the following definitions.

2.1. Let $N = \{n_1, \ldots, n_k\}$ be a collection of k, not necessarily distinct, ordered pairs of the form $n_i = (l_i, v_i)$ where l_i is an alphanumeric symbol, (the label of n_i) and v_i is a positive integer (the valence value of n_i). We call such a collection N an atom set. By a graph based on the atom set N we mean a loop-free, connected multigraph G = (N, D) with node collection N and edge collection D such that the degree of each n_i in G is equal to v_i[4].

We say that two graphs based on N, say G = (N, D) and H = (N, E) *are* isomorphic if and only if there is a graph isomorphism Ψ from G to H which preserves labels, i.e., Ψ is a permutation of N such that the multiplicity of each (n_i, n_j) in D is equal to the multiplicity of $(\Psi(n_i), \Psi(n_j))$ in E and $\Psi(n_i) = n_j$ implies $l_i = l_j$. Since G and H are graphs based on N, such a Ψ must, necessarily, also preserve valence values.

If G = (N, D) and H = (M, E) are graphs where the node collections N and M are both atom sets, then H is said to be a subgraph of G if there is an injection η from M into N which preserves connectivity, labels and valence values, i.e., the multiplicity of each (m_i, m_j) in E does not exceed the multiplicity of $(\eta(m_i), \eta(m_j))$ in D and $\eta(m_i) = n_j$ implies m_i and n_j

[4] Note that we distinguish here between the valence value of an atom and its degree as a node in a graph.

have the same label and valence value.

In terms of these definitions, our molecular structure problem can now be stated as follows:

Given: An atom set N and a list of q loop-free, connected multigraphs $H_1 = (M_1, E_1), \ldots, H_q = (M_q, E_q)$ with each Mi an atom set and satisfying:

i) The disjoint union of the M_i is a subcollection of N.

ii) The degree of a node in any H_i does not exceed its valence value.

iii) In each H_i at least one node has degree less than its valence value.

Determine: A representative set of the isomorphism classes of those (loop-free, connected, multi-) graphs based on N which contain $H_1, H_2, \ldots$, and H_q as pairwise disjoint subgraphs.

Using our current techniques, a direct, effective, computer implementable solution to the above problem does not seem possible. Our solution strategy, therefore, consists of reducing this problem to an iterative sequence of simpler problems which, when solved, yields a collection of graphs containing the desired representative set but possibly with redundancies, These redundancies, as produced, are pruned from the collection. We now describe our problem reduction technique.

2.2. Let $k_i = \sum_{m_j \in M_i}$ (valence m_j - degree in H_i of m_j). k_i is called the free valence of H_i. It corresponds to the number of unassigned valences in the fragment whose known structure is represented by H_i. By assumption, k_i is positive.

Since we consider only connected graphs, in any solution graph at least one of the free valences of H_i must be used for an edge going from a node in M_i to a node not in M_i. Moreover, those free valences used for edges going between nodes in M_i must occur in pairs. Accordingly, we let $B = \{(b_1, \ldots, b_q) \mid 0 < b_{i1} \leq k_i , b_i \equiv k_i \pmod 2\}$, where each difference $k_i - b_i$ indicates the number of free valences to be used by edges going between nodes in M_i.

Let $y_i, \ldots, y_q$ be q distinct atom labels different than any of the labels of the atoms in N. For each $b = (b_1, \ldots, b_q)$ in B, let N^b denote the atom set obtained from N by deleting from N all the atoms in the disjoint union of the M_i and adding to the remaining atoms the set of q new atoms $\{x_1 = (y_i, b_i) \mid i = 1, \ldots, q\}$. For each so modified atom set N^b, consider the following sequence of constructions:

1. Construct a representative set of the isomorphism classes of the graphs based on N^b.
2. For each graph G^o constructed in step 1, for i = 1, . . . , q , iteratively construct a representative set of the isomorphism classes of all the graphs G^i obtained from all the graphs G^{i-1} as follows:
   a) Add $(k_i - b_i)/2$ edges to H_i in such a way that the resulting degree of each node m_j in H_i does not exceed the valence value of m_j.
   b) Delete the atom x_i from G^{i-1} and replace each edge in G^{i-1} of of the form (n_s , xi) with an edge of the form (n_s , m_3) in such a manner that the resulting degree of each m_3 is equal to the valence value of m_j.

Each graph produced by this sequence of constructions will be a graph satisfying the conditions of our molecular structure problem. Moreover, if for each b in B we perform these constructions, the resulting collection of graphs will contain, up to isomorphism, all solution graphs of our original problem but possibly with redundancy.

We have previously developed and implemented an algorithm which, given a degree sequence of nodes representing the atoms of an organic molecule, determines a representative set of the isomorphism classes of all loop free, connected multigraphs based on that degree sequence [2]. This algorithm yields an effective solution to step 1 of the above construction. Thus, up to redundancy elimination which is discussed in Section 3.6, our molecular structure problem is reduced to the problem of deriving an effective algorithm for step 2 of the construction. We call this latter problem the fragment embedding problem.

3. Fragment Embedding. In this section we will give an independent, more precise formulation of the fragment embedding problem, and we will show that this problem can be represented, at least partially, as a special double coset representative problem,

3.1. Let G = (M, D) and H = (N, E) be connected, loop-free, multigraphs with disjoint node sets $M = \{m_1, \ldots, m_k\}$ and $N = \{n_1, \ldots, n_q\}$ and edge sets D and E, respectively. Here, the edge sets are considered as unordered pairs of nodes with multiple edges appearing multiply. For nodes m_i in M and n_j in N, we now formally define an embedding of H at n_j in G at m_i where degree (n_j) - degree (m_i) is non-negative and even. To simplify the notation we assume, without loss of generality, that i = j = 1.

An embedding of H at n_1 *in G at* m_1 is a multigraph A = (B, C) where

i) The node set B consists of $M \cup N \setminus \{m_1 , n_1\}$, i.e., all the nodes of both G and H except m_1 and n_1.

ii) The edge set C consists of

$$\{(m_i, m_j) \in D \mid i \neq 1, j \neq 1\} \cup \{(n_i, n_j) \in E \mid i \neq 1, j \neq 1\} \cup F \cup K ,$$

where F satisfies:

a) Every element in F is an edge of the form (m_i, n_j) where $(m_1 , m_i) \in D$ and $(n_1, n_j) \in E$.

b) For each n_3 in N, the number of edges in F having n_j as an endpoint does not exceed the multiplicity of the edge (n_1, n_j) in H.

c) For each m_i in M, the number of edges in F having m_i as an endpoint is equal to the multiplicity of the edge (m_1 , m_i) in G.

and K satisfies:

a! Every element K is an edge of the form (n_i, n_j), $i \neq j$, where both (n_1, n_i) and (n_1, n_3) are in E.

b) For each n_i in N, the sum of the number of edges in F having n_i as an endpoint and the number of edges in K having n_i as an endpoint is equal to the multiplicity of (n_1, n_i) in H.

That is, C consists of all edges in D except those with endpoint m_1, all edges in E except those with endpoint n_1, the connecting edge set F and the internal edge set K. Note that by definition, an embedding is a connected, loop-free multigraph, and it is completely determined by the edge sets F and K.[5)]

[5)] This formulation of the embedding problem corresponds to the formulation in the previous section as follows:
a) G corresponds to a graph G^{i-1} and the [illegible] :he
b) H c 'responds to the [illegible]

Our objective is to develop a reasonably efficient, computer implementable algorithm which accepts as input the graphs G and H and which outputs a representative set for the topological isomorphism classes of the embeddings of H at n_1 in G at m_1.

We consider first the special case where degree (m_1) = degree (n_1), i.e. the case where the internal edge set K is empty.

3.2. Let w = degree (m_1) = degree (n_1), and let S_w denote the full permutation group on $\{1, 2, \ldots, w\}$. We index from 1 to w all edges in D of the form (m_1, m_i), say index $(m_1, m_{i(t)}) = t$, $t = 1, \ldots, w$, where for definitiveness, we require that $t_1 < t_2$ implies $i(t_1) \leq i(t_2)$. Similarly, we index from 1 to w all edges in E of the form (n_1, n_j), say index $(n_1, n_{j(t)}) = t$, where $t_1 < t_2$ implies $j(t_1) \leq j(t_2)$. For any ψ in S_w, we define $F(\psi)$ as the set of (multiple) edges $\{(m_{i(t)}, n_{j(\psi(t))}) \mid t=1, \ldots, w\}$. $F(\psi)$ is a connecting edge set of an embedding of H at n_1 in G at m_1. Conversely, if F is a connecting edge set for an embedding of H at n_1 in G at m_1, we define the map $\pi(F)$ iteratively as follows:

For $t = 1, \ldots, w$, $\pi(F)(t) = t_1$ where t_1 is the least unassigned index such that $(m_{i(t)}, n_{j(t_1)})$ is in F. $\pi(F)$ is a well-defined permutation in S_w. Moreover, for any connecting edge set X, $F(\pi(X)) = X$. Hence we have:

Lemma 1. Let degree (m_1) = degree (n_1) = w. Relative to an indexing of the edges of G with endpoint m_1 and the edges of H with endpoint n_1, there is a surjective correspondence from the elements of S_w onto the embeddings of H at n_1 in G at m_1.

We will now show that there is a surjective correspondence between a certain set of double coset, representatives in S_w and the topologically distinct, i.e., nonisomorphic, embeddings of H at n_1 in G at m_1.

Let Grp(G) be the topological symmetry group of G considered as acting on the nodes of G, and let Stab (G) be the stabilizer in Grp(G) of ml, i.e., Stab (G) = $\{\alpha \in \mathrm{Grp}(G) \mid \alpha(m_1)=m_1\}$. If, as above, we index those edges of G with endpoint ml, then each node map α in Stab (G) naturally induces a well-defined permutation $\tau(\alpha)$ in S_w follows: For any index t, $(m_1, m_{i(t)})$ is the edge in D with index t and $(m_1, \alpha(m_{i(t)}))$ must also be an edge in D. Moreover, both $(m_1, m_{i(t)})$ and $(m_1, \alpha(m_{i(t)}))$ have the same multiplicity in G, say k. Let x and y be the least indices of the k edges $(m_1, m_{i(t)})$ and the k edges $(m_1, \alpha(m_{i(t)}))$, respectively. Since multiple edges were indexed in sequence, t = x + b for some $0 \leq b < k$, and we define $\tau(\alpha)(t) = y + b$. Since Stab (G) is a subgroup of Grp(G), the set I(G) = $\{\tau(\alpha) \mid \alpha \in \mathrm{Stab}(G)\}$ is a subgraup of S_w.

For each i such that (ml, m_i) is in D, say the multiple edges (ml, m_i) are indexed by $x_i, x_i + 1, \ldots, x_i + k - 1$ where k is the multiplicity of (m_1, m_i) in G, let $S^{(i)}$ denote the full permutation group on $\{x_i, \ldots, x_i + k - 1\}$ considered as a subgroup of S_w. Let M(G) denote the internal direct product of all the $S^{(i)}$ such that (ml, m_i) is in D. Then, I(G) $\cdot$ M(G) = M(G) .I(G) and I(G) and M(G) have only the identity in common. Hence, the set product U(G) = I(G) .M(G) is a subgroup of S_w with order (U(G)) = order (I(G)) .order (M(G)).

In a completely analogous manner, we define the subgroups I(H), M(H) and U(H) of S_w corresponding to H at n_1.

<u>Lemma 2.</u> Let γ and δ be two elements in S_w lying in the same double coset of U(H) and U(G) *in* S_w, i.e., $U(H)\,\gamma\, U(G) = U(H)\,\&J(G)$. Then, the embeddings C_γ and C_δ of H at n_1 in G at m_1 determined by γ and δ , respectively, are topologically isomorphic graphs.

<u>Proof.</u> Since δ is an element of $U(H)\,\gamma\, U(G)$, $\delta\tau_2\tau_1 = \eta_2\eta_1\gamma$ for some $\tau_1 \in I(G)$, $\tau_2 \in M(G)$, $\eta_1 \in I(H)$ and $\eta_2 \in M(H)$. Let $\tau \in$ Stab(G) and $\eta \in$ Stab(H) be elements inducing τ_1 and η_1, respectively. By definition, both C_γ and C_δ have the same node set $L = D\cup E \diagup \{m_1, n_1\}$. We define a map ψ on L by $\psi(m_i) = \tau(m_i) \ldots \psi(n_i) = \eta(n_i)$. Since $\tau(m_1) = m_1$ and $\eta(n_1) = n_1$, ψ is a well-defined permutation of L. Yoreover, since $\tau \in$ Grp(G), ψ restricted to the subgraph of C_γ consisting the edges of C_γ of the form (m_i, m_j) is an isomorphism from this subgraph to the corresponding subgraph of C_δ . Similarly, ψ determines an isomorphism from the subgraph of the edges of the form (n_i, n_j) in C_γ to the corresponding subgraph of C_δ . Thus to show that ψ is an isomorphism from C_γ tc C_δ , we need only consider the action of ψ on $F(\gamma)$. We claim that (m_x, n_y) is in $F(\gamma)$ if and only if $(\psi(m_x), \psi(n_y))$ is in $F(\delta)$.

For any pair (m_x, n_y), let $\psi(m_x) = m_u$ and $\psi(n_y) = n_v$. Then, by definition of $F(\gamma)$, (m_x, n_y) is in $F(\gamma)$ iff

i) $x = i(t)$ and $y = j(\gamma(t))$ for some index t. By definition of ψ, (i) is true iff

ii) $u = i(\tau_1(t))$ and $v = j(\eta_1(\gamma(t)))$.

Since τ_2 only moves the index of an edge with endpoint m_1 to the index of one of its multiples and similarly for η_2, (ii) is true iff

iii) $u = i(\tau_2\tau_1(t))$ and $v = j(\eta_2\eta_1\gamma(t))$. By assumption, (iii) is true, iff

iv) $u = i(\tau_2 \tau_1(t))$ and $v = j(\gamma \tau_2 \tau_1(t))$. By d finition of $F(\delta)$, (iv) is true iff$(\psi(m_x), \psi(n_Y))$ is in $F(\delta)$.

Hence, ψ is an isomorphism from C_γ to C_δ.

☜⊠♋○□•♏✍ Let H and G be the graphs in figures la and lb, respectively.

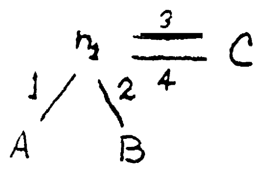


Fig. 1a

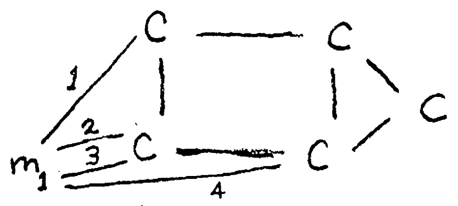


Fig. 1b

Here A, B and C are the node (atom) labels. Both Grp (G) and Grp(H) consist of only the identity map and, using the image vector notation for elements of S_4,

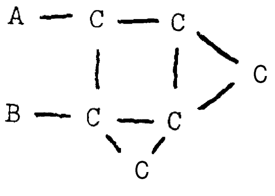
$U(G) = \{(1,2,3,4), (1,3,2,4)\}$ and

$U(H) = \{(1,2,3,4), (1,2,4,3)\}$.

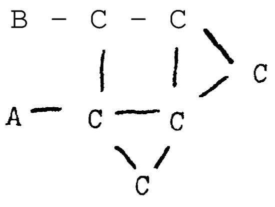
There are seven double cosets of U(H) and U(G) in S_4. A set of double coset representatives is:

a. (1,2,3,4)
b. (2,1,3,4)
c. (3,2,1,4)
d. (4,2,3,1)
e. (3,1,4,2)
f. (1,4,3,2)
g. (2,3,4,1)

The corresponding embedding are given in Figure 2.



(a)



(b)

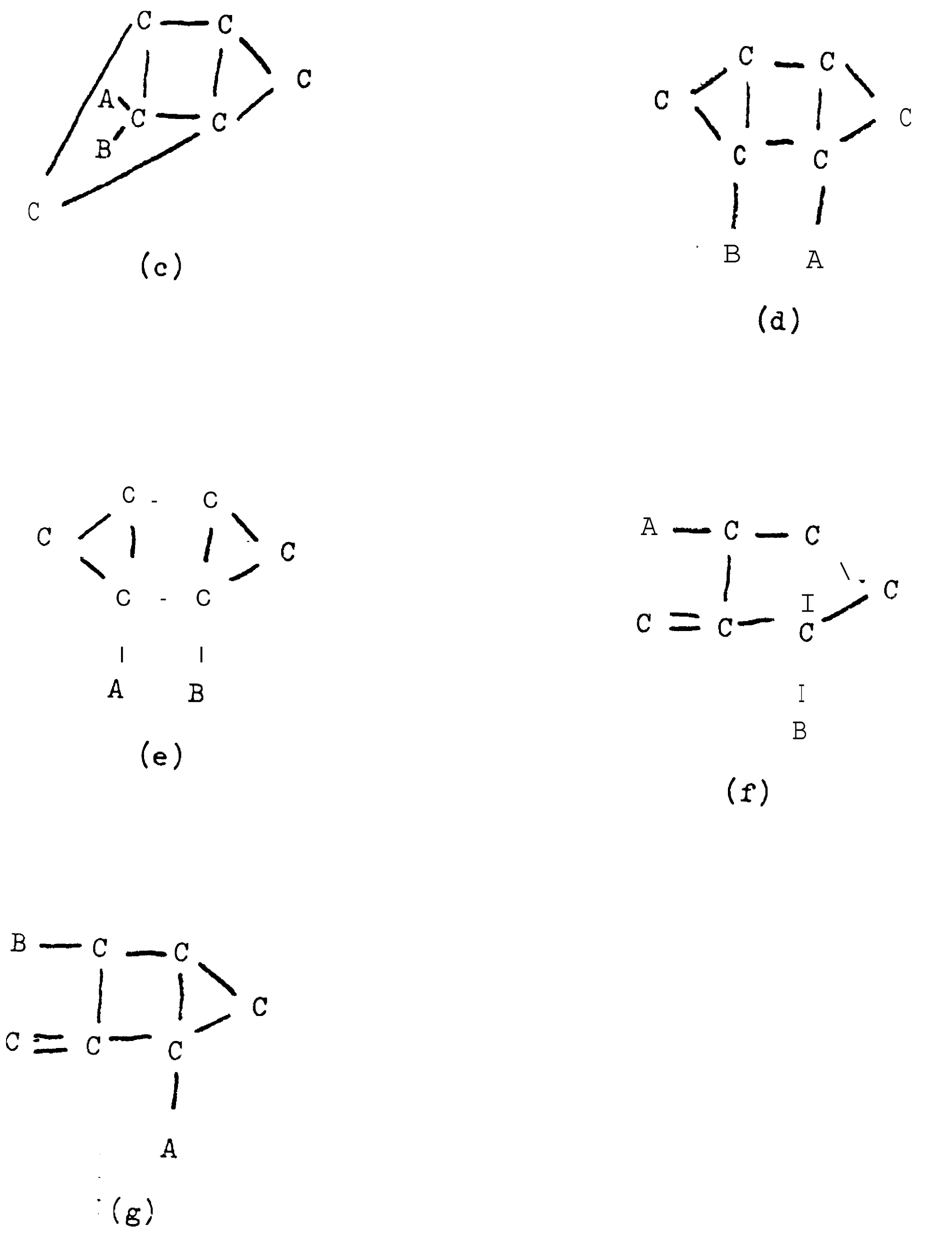


Figure 2.

Note that the embeddings (d) and (e) are topologically isomorphic.

As shown in the above example, the converse of Lemma 2 is not true. The difficulty here is that an embedding may have symmetries not induced by symmetries of its components. We do have, however, the following result:

Lemma 3. Let γ and δ be elements of S_w with associated embeddings of H at n_1 in G at m_1, C_γ and C_δ, respectively. If there exists an (node) isomorphism ψ from C_γ to C_δ such that $\psi(D \setminus \{m_1\}) = D \setminus \{m_1\}$, i.e., ψ permutes the nodes of G, and hence. also those of H, among themselves, then γ and δ are in the same double coset of U(H) and U(G) in S_w.

Proof. By the definition of an embedding, there is a τ in Stab(G) such τ and ψ agree on the subgraph of C_γ consisting of all edges of the form (m_i, m_j). Similarly, there is an $\eta \in$ Stab(H) such that η and ψ agree on the subgraph of C_γ consisting of all edges of C_γ of the form (n_i, n_j). Also, (m_i, n_j) is in $F(\gamma)$ iff$(\psi(m_i), \psi(n_j))$ is in $F(\delta)$. Thus we have that (m_i, n_j) is in $F(\gamma)$ iff$(\tau(m_i), \eta(n_j))$ is in F(6). Let τ_1 and η_1 be elements of U(G) and U(H) induced by τ and η, respectively. Then, up to a permutation of indices on multiple edges, we have that for any index t:

$(m_{i(t)}, n_{j(\gamma(t))})$ in $F(\gamma)$ implies

$(m_{i(\tau_1(t))}, n_{j(\eta_1 \gamma(t))})$ is in $F(\delta)$ implies

$\delta\tau_1(t) = \eta_1\gamma(t)$.

Hence, $\eta_1^{-1}\delta\tau_1 = \gamma$ up to permutations of the multiple edge indices, and's $\in$ U(H) γ U(G).

The above results yield a method for determining all embeddings of H at n_1 in G at m_1 where degree (ml) = degree $(n_1) = w$. Namely,

1. Construct the subgroups U(G) and U(H) of S_w.
2. Construct a set of double coset representatives for U(H) and U(G) in S_w.

3. Construct the set of graphs determined by these double coset representatives.

4. Eliminate any isomorphic duplicates from this set of graphs.

Although this method does produce initially a set of graphs with possible redundancies, a great deal of empirical evidence leads us to believe that at least in the case of graphs of organic molecules, the number of duplicates is statistically relatively small, e.g., less than 10%. Moreover, since the resulting graphs are needed later in a canonical form, the additional effort needed to prune out duplicates is not excessive.

3.3. We will consider first the problem of determining the groups U(G) and U(H). The essential problem is given a graph G = (M, D), a node m_1 in M, say degree (ml) = w, and an indexing from 1 to w of the edges in G with endpoint m_1 where multiple edges are indexed in sequence, effectively determine the subgroup I(G) of S_w induced on the edge indices by Stab(G), the subgroup of the topological symmetry group of G which fixes m_1. The derivation of U(G) from I(G) is a straight forward process.

Most graph symmetry group'algorithms are based on the following technique:

1. Partition the node set M of the graph G such that each member of the partition is a union of orbits of nodes with respect to the topological isomorphism group of G.
2. Via a recursive backtrack generation scheme, systematically generate those node permutations which preserve the partition, i.e., which carry-a node m_i to an element in the member of the partition containing mi. Here, the ith level of generation is choose the

image of m_i subject to the condition that the partially determined permutation preserves the adjacency structure of G.

Often the partitioning of M is done via a sequence of partitions P_1, ... P_k, where associated with each partition P_t is an isomorphism invarient node weight function W_t. Here two nodes m_i and m_j are in the same member of the partition P_t if and only if m_i and m_j are in the same member of P_{t-1} and $W_t(m_i) = W_t(m_j)$. A simple example at such a node weight function is the degree function. The finest partition of the nodes would be the orbit partition. However, since one wants a partition which is relatively cheap to compute, a compromise which yields a partition coarser than orbits is usually used, for example, the Morgan partition [4].

For the problem of determining I(G), empirical evidence indicates that the following sequence of node weight functions yields an effective partition:

$W_1(m_i)$ = label of node m_i.

$W_2(m_i)$ = 1 if m_i is adjacent to m_1 else 0.

$W_3(m_i)$ = degree (m_i)

$W_k(m_i) = \sum W_{k-1}(m_j)$ where the sum is over all m_j adjacent to m_i counted with multiplicity, $k>3$.

The partitioning is done iteratively until either all members of the current partition are singleton sets or two, not necessarily successive, iterations do not yield finer partitions.

Since we need only the permutations induced on the edge indices by Stab(G), the following economies are made in our algorithm:

1. The singleton set $\{m_1\}$ is made a member of the first node partition, and the node m_1 is not considered further in the partitioning process.
2. For the backtrack permutation generation routine, the nodes are ordered so that:
   a) Those nodes which occur as singleton sets in the final partition come first, say $m_{x_1}, \ldots . m_{x_k}$.
   b) Of the remaining nodes, those which are adjacent to ml come next, say $m_{x_{k+1}}, \ldots, m_{x_t}$.
   c) The nodes in each member of the final partition are in sequence.

   Then, the backtrack generation starts at level x_{k+1} and, whenever an allowable permutation is generated, the algorithm backtracks immediately to level x_t.

An algorithmfor generating I(G) which implements the above ideas is given in Appendix III.

3.4. We will now consider the double coset problem. The problem of effectively determining a set of double coset representatives for two subgroups A and B in S_w is very difficult. In fact, at least to the author's knowledge, no generally effective, computer implementable algorithm to perform this task is known. However, in the case of fragment embedding in graphs of organic molecules, both w and the number of double cosets are usually sufficiently small that a fairly weak algorithm suffices.

The double coset representative algorithm which we present here is based partially on ideas due to Charles Sims [5].

The group Sw admits a natural-total ordering " $<<$ ". Namely, we associate with each $\pi \in S_w$ the vector $(\pi(1), \pi(2), \ldots, \pi(w))$, and for α and β in S_w, we define $\alpha << \beta$ if and only if the associated *vector* of α is lexicographically less than or equal to the associated vector of β. If X is a subset of S_w, we write $\alpha << X$ if and only if $\alpha << x$ for every $x \in X$.

We select as the canonical representative γ of a double coset $A\pi B$ of A and B in S_w the least element in $A\pi B$, i.e., that γ in $A\pi B$ satisfying $\gamma << A\pi B$. Since a double coset is determined by any of its members, i.e., $A\pi B = A\gamma B$ if and only if $\gamma \in A\pi B$, we have that for α in $A\pi B$, $\alpha << A\pi B$ if and only if $\alpha << A\alpha B$.

Clearly if γ is the least element in $A\gamma B$, then $\gamma << A\gamma$ and $\gamma << \gamma B$. The converse, unfortunately, is not true. Even so, if we can determine all γ in S_w satisfying $\gamma << A\gamma$ and $\gamma << \gamma B$, we have effected a considerable reduction of the double coset problem.

Lemma 4. (Sims [5]). Let B be asubgroup of S_w. Let 0_i, i=1, ... w-1, be the orbit of i with respect to the elementwise stabilizer in B of $\{1,2,\ldots,i-1\}$, i.e., $0_i = \{\pi(i) \mid \pi \in B \text{ and } \pi(j) = j,\ 1 \leq j < i\}$. Then for $\eta \in S_w$, $\eta << \eta B$. if and only if $\eta(i) \leq \eta(x)$ *for every* x in 0_i, i=1, ... w-1.

The above lemma yields a very powerful method for generating those γ in Sw satisfying $\gamma << \gamma B$. However, relative to our functional notation for permutations, it is only applicable to left cosets. The technique that we use for determining those γ satisfying $\gamma << A\gamma$, is much more direct, and it is based on the following elementary ol

Let A be a subgroup of S_w. Then $\gamma \in S_w$ satisfies $\gamma << A\gamma$ if and only if $\gamma(j) = \text{MIN}\{\tau(\gamma(j))\}$ where τ ranges over $\text{STAB}_A(\gamma(1), \ldots, \gamma(j-1)) = \{\alpha \in A \mid \alpha(\gamma(x)) = \gamma(x), x=1, \ldots, j-1\}$. In particular, $\gamma(1)$ must be the least element in its A orbit.

Our method of generation of representative permutations is based on a backtrack scheme where the j-th level of the backtrack tree corresponds to selecting the image of j under γ. Since we seek only those $\gamma \in S_w$ which satisfy $\gamma << A\gamma B$ and hence, $\gamma << \gamma B$ and $\gamma << A\gamma$ we note that:

1. If a potential image k for j under γ is rejected because it violates the condition of Lemma 4, then by the above observation, we can also eliminate from consideration the values $\{\tau(k) \mid \tau \in \text{STAB}_A(\gamma(1), \ldots, \gamma(j-1))\}$ as potential values for $\gamma(j)$.

2. Let $A_1, \ldots, A_k$ be the orbits of A, i.e., the A_i form the partition of $\{1, \ldots, w\}$ induced by the equivalence relation $x \sim y$ iff $y = \alpha(x)$ for some $\alpha \in A$. Here, we canonically index the A orbits via $\text{MIN}\{x \in A_i\} 4 \text{ MIN}\{x \in A_{i+1}\}$. Let B_1 be the B orbit of 1. Then, if $\gamma(1) \in A_i$, $i > 1$ and $j \in B_1$, we can eliminate from consideration as values for $\gamma(j)$ the elements of $A_1 \cup A_2 \cup \ldots \cup A_{i\ 1}$.

Based on the above ideas, we now present *an* algorithm in an ALGOL type format, which given w and two subgroups A and B of S_w first generates those γ in S_w satisfying $\gamma << A\gamma$, $\gamma << \gamma B$ and $\gamma(1) \leq \alpha\gamma\beta(1)$ for every $\alpha \in A$ and $\beta \in B$. We note that the set of all $\gamma \in S_w$ satisfying the above conditions does contain the canonical double coset representatives for A and B in S_W. Moreover, such a γ satisfies $\gamma << A\gamma B$ if and only if $\gamma << \alpha\gamma\beta$ for all pairs $\alpha \in A$ and $\beta \in B$ satisfying $\gamma^{-1}\alpha^{-1}\gamma(1) = \beta(1)$. This latter condition is used directly by the algorithm to test each permutation as it is generated. A proof that the algorithm does produce the desired output is given in Appendix II

Algorithm I.

Input: An integer $W > 0$ and two subgroups A and B of S_W.

output: The canonical set of double coset represetnatives for A and B in S_W.

```
BEGIN
Determine the $O_i$ of Lemma 4, i=1,2,...,W-1
Determine $R_i = \{j \mid i \in O_j,\ j < i\}$, i=2,3,...,W
Determine the orbits of A; $A_1, A_2, \ldots,$ AT where $\mathrm{MIN}\{x \in A_I\}$
        $\leq \mathrm{MIN}\{x \in A_{I+1}\}$
Initialize: $k \leftarrow 1,\ j \leftarrow 2,\ P_1 \leftarrow 1,\ SB_2 \leftarrow$ IF $1 \in R_2$ THEN 1 ELSE 0,
        $IM_2 \leftarrow \{2, \ldots, W\}$
WHILE $k \leq T$ (T = number of A orbits) DO
    BEGIN k-loop
    WHILE $IM_j \neq \emptyset$ DO
        BEGIN IM-loop
        $P_j \leftarrow \mathrm{MIN}\{x \in IM_j\}$
        IF $P_j < SB_j$ THEN
            Determine $S_j = \mathrm{STAB}_A(P_1, \ldots, P_{j-1})$
            $IM_j \leftarrow IM_j \setminus \{\tau(P_j) \mid \tau \in S_j\}$
```

```
ELSE
    $j \leftarrow j + 1$
    $SB_j \leftarrow MAX \{P_x \mid x \in R_3\}$
    IF $j \leq W-1$ THEN
        $IM_j \leftarrow \{1,2,\ldots,W\} \setminus \{P_1, \ldots, P_{3-1}\}$
            $\setminus \{$IF $i=1$ OR $j \notin O_1$ THEN $\emptyset$
              ELSE $A_1 \cup . \bullet$ [illegible] $i-1$ 3
   ELSE
   $P_w \leftarrow$ ELEMENT$(\{1, \ldots, w\} \setminus \{P_1, \ldots, P_{w-1}\})$.
   IF $P_w < SB_w$ THEN
        IF $\gamma = (P_1, \ldots, P_w)$ satisfies
              $\gamma << \alpha \gamma \beta$ for all pairs
              $\alpha \in A$, $\beta \in B$ satisfying
             $\gamma^{-1} \alpha^{-1} \gamma(1) = \beta(1)$ THEN
           output $\gamma$
      $j \leftarrow j - 1$
      Determine $S_j = STAB_A (P_1, \ldots, P_{j-1})$
      $IM_j \leftarrow IM_j \setminus \{\tau(P_j) \mid \tau \in S_j\}$
  END IM-loop
IF $j > 2$ THEN
   $j \leftarrow j - 1$
   Determine $S_j = STAB_A (P_1, \ldots P_{j-1})$
   $IM_j \leftarrow IM_j \setminus \{\tau(P_j) \mid \tau \in S_j\}$
ELSE
   $k \leftarrow k + 1$
   IF $k \leq T$ THEN
      $P_1$ [illegible]
```

```
            $SB_2 \leftarrow MAX\{P_x \mid x \in R_2\}$
            $IM_2 \leftarrow \{1,2,$ [illegible] $\setminus \{P_1\}$
        END k-loop
    END.
```

3.5. The above algorithm coupled with algorithms to determine the relevant groups, to perform the necessary edge indexing and to output the embedded structures yields an effective, computer implementable scheme for solving the embedding problem in the case where the degree of the replaced atom in the molecule is equal to the number of unassigned valences in the fragment, i.e., in the case that degree (ml) = degree(n_1).

The case where degree (n_1) > degree (m_1), degree (n_1) = degree (ml) (mod 2) can be handled by a simple extension of the above techniques. This is the case where some of the bonds to be allocated are internal to the fragment.

Let k = (degree (n_1) - degree (m_1))/2. We construct a new graph G' obtained from G by adding k new bivalent nodes all with a label different than any label occurring previously in G, say NIL, where each NIL labeled node is double bonded to m_1. The embedding problem for H in G' is of the above considered type and thus cun be handled by our algorithms. In the resulting embedded structures, the NIL labeled nodes are then simply erased leaving bonds internal to the fragment. Since we want to generate *only* loop-free structures, the following constraint is inserted in the backtrack double coset representative generation algorithm:

No pair of (successive) edge indices which correspond to a double bond to a node with label NIL in G' can map to a pair of edge indices which correspond to two multiple edges from a node n_i to n_1 in H.

A simple flagging of the relevant edge indices yields an economical method for testing this additional constraint.[6)]

3.6. As shown by the example in section 3.2, the double coset algorithm, as well as the overall technique, can produce isomorphic structures. These duplicate structures are eliminated by a direct pruning based on a canonical node indexing scheme.

There are numerous canonical node indexing algorithms in use, i.e., algorithms which index the nodes of a graph in such a manner that two graphs are isomorphic if and only if relative to the indexings, they have identical adjacency (or indidence) matrices. The indexing algorithm most used for chemical structures is due to Hm Morgan [4]. Morgan's indexing scheme is based on a node weight classification similar to that used by our group determination algorithm. This scheme is the basis of the Chemical Abstract% Serial Index of Organic Compounds. - an index containing, at present, about three million structures.

Since it was desired that the structures produced by our programs be compatible with Chemical Abstract's Serial Index, all output from our structure elucidation package is in a Morgan-type canonical form. In particular, as each structure is produced by the embedder, it is checked against possible additional constraints given by the user, and, if it satisfies these constraints, it is put into canonical form. This

6) Several types of constraints on the embedded structures can be introduced at the backtrack generation level by using edge flags.

canonicalized structure is then directly compared with the previously generated structures for possible duplication. In the process of canonicalizing a structure, a simple many-to-one one word isomorphism invarient structure key is determined: This key is used to economize structure comparison.

4. Implementation. The above described fragment embedding algorithm has been coded in SAIL, an ALGOL like language. This computer implementation makes use of numerous devices to speed up the computation, for example, the cases $w \leq 3$ are handled directly - bypassing the double coset algorithm entirely, and the stabilizers $STAB_A(P_1,\ldots,P_{j-1})$ are recomputed only when necessary.

The fragment embedding program has been extensively tested using the Stanford University Medical Experimental Computing Facility (SUMEX). This facility is based on a PDP-10 computer running under the TENEX operating system. The average execution time is about .3 seconds per completed structure.

The embedder program is incorporated in the general molecular structure elucidation package under development at Stanford as part of the DENDRAL project [3]. This package has a chemist-oriented I/O interface which permits the user to input only the emperical formula for a molecule, the desired fragments in graphical form and several other types of informationabout the unknown compountm The package driver itself then calls the *necessary* structure generation routines, thus freeing the user from this task.

ACKNOWLEDGEMENTS

This work was supported in part by NIH Grant RR612-05A1 and NSF Grant DCR74-12646.

REFERENCES

[1] H. Brown, L. Hjelmeland and L. Masinter, Constructive Graph Labeling Using Double Cosets, Discrete Math., 7 (1974) 1-30.

[2] _____ and L. Masinter, An Algorithm for the Construction of the Graphs of Organic Molecules, Discrete Math., 8 (1974) 227-244.

[3] D. Smith, L. Masinter and N. Sridharan, Heuristic DENDRAL: Analysis of Molecular Structure, in Computer Representation and Manipulation of Chemical Information (J. Wiley and Sons, New York, 1974)287-315.

[4] H. Morgan, The Generation of a Unique Description for Chemical Structures - A Technique Developed at Chemical Abstracts Service, J. Chem. Doc. 5(2) (1965) 107-113.

[5] C. Sims, Computations with Permutation Groups, in Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation (ACM, New York, 1971) 23-28.

Appendix I. Example

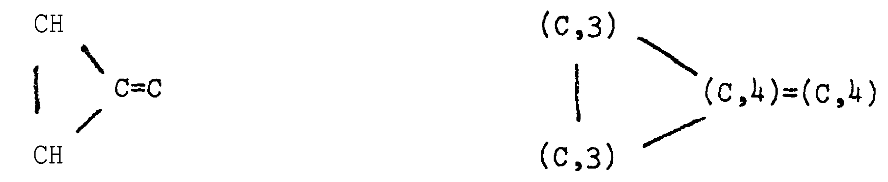
The following example was chosen' for its convenience in illustrating the operation of the fragment embedder rather than for its chemical relevance.

<u>Atom Set</u>: $\{(C,4),(C,4),(C,4),(C,3),(C,3),(C,3),C,2),(C,2),(O,2),(R,1)\}$
This atom set represents the emperical formula C_8OH_7R where R is a monovalent radical and the bonding of the hydrogen atoms to the carbon atoms is known, namely there are three CH's and two CH_2's.

<u>Fragments</u>:

Atomic Form — Label - Valence Form

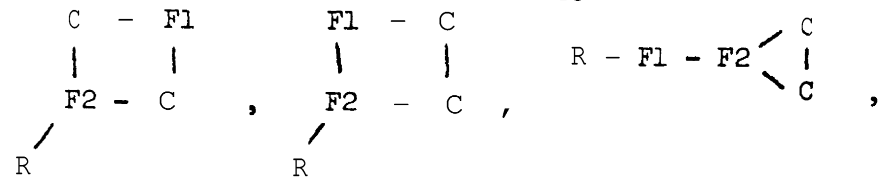
F1:

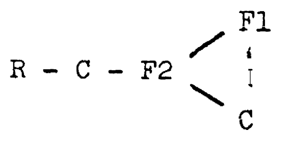


F2: O = C - CH — (O,2)=(C,4)- (C,3)

Fragment F1 has free valence 4 and fragment F2 has free valence 3. We assume that it is not known how the free valences are distributed between bonds internal and external to the fragment F1; and that the free valences on fragment F2 are all used for external bonds. Hence there are two cases. Namely, F1 has *one* additional internal bond and F1 has no additional internal bonds.

<u>Case 1.</u> F1 has one additional internal bond. Reduced atom set:
$\{(F1,2),(F2,3),(C,2),(C,2),(R,1)\}$.
There are seven non-isomorphic graphs based on this atom set.

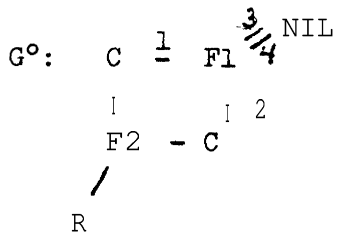


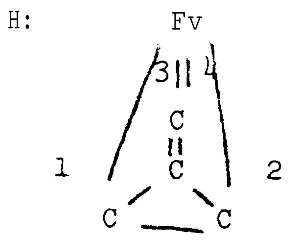


, C = F2 − F1 − C − R

C = F2 − C − F1 − R , F1 = F2 − C − C − R

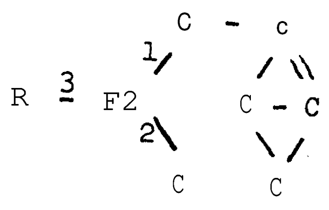
We will consider only the first graph. Then:



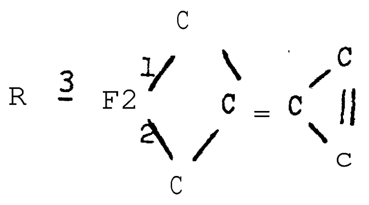


We want first the embeddings of H at FV in G at F1. If we index the relevant edges as indicated, then U(H)=U(G)= {(1,2,3,4), (2,1,3,4), (1,2,4,3), (2,1,4,3)}. There are three double cosets of U(H) and U(G) in S_4. The canonical set of double coset representatives is {(1,2,3,4),(1,3,2,4), (3,4,1,2)}. The first representative violates the bonds to NIL atom condition and is discarded. The second and third representatives give, respectively, the following structures G^1:

G'_1

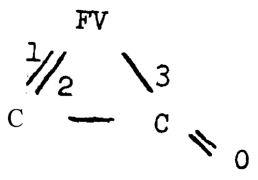


G'_2



The graph H corresponding to F2 is:

H:



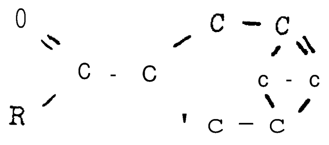
The associated groups are:

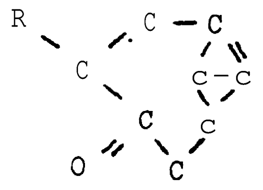
$U(H) = \{(1,2,3),(2,1,3)\}$,

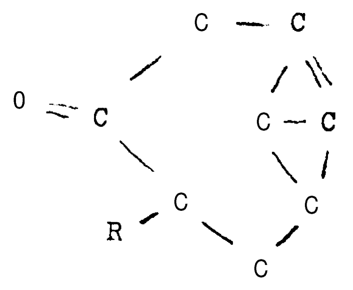
$U(G'_1) = \{(1,2,3)\}$,

$U(G'_2) = \{(1,2,3),(2,1,3)\}$.

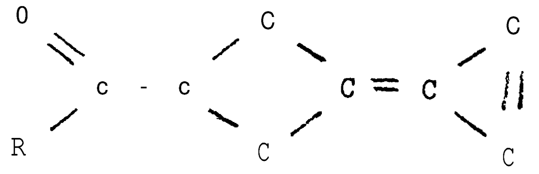
The canonical set of double coset representatives for U(H) and $U(G'_1)$ in S_3 is $\{(1,2,3), (1,3,2), (3,1,2)\}$. These representatives correspond, respectively, to the following structures:

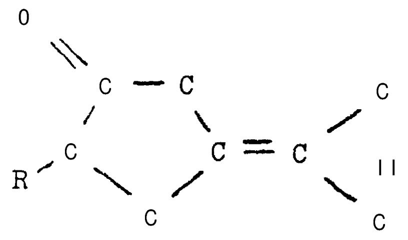






The canonical set of double coset representatives for U(H) and $U(G'_2)$ in S_3 is $\{(1,2,3), (1,3,2)\}$. These correspond to the structures:

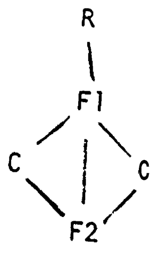


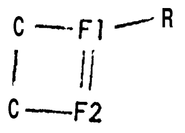


All of the final structures are distinct. Thus there are five embedded structures based on the first graph and with one allocated bond internal to the fragment F1

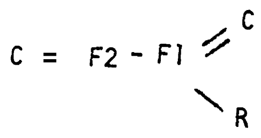
Case 2. No additional internal fragment bonds. Reduced atom set: {(F1,4), (F2,3), (C,2), (C,2), (R,1)}. There are eight distinct graphs based on this atom set.

F2 = Fl - C C - R , C = Fl = F2 = C - R,

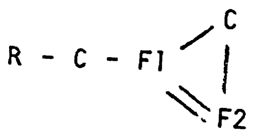




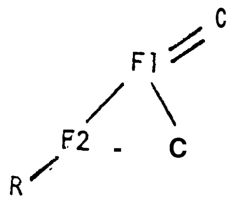
,



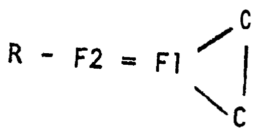
,



,



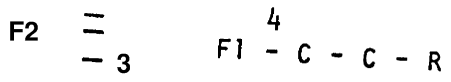
,



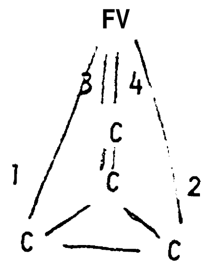
.

We will consider only the first graph. Then:

G^{O}:



H :



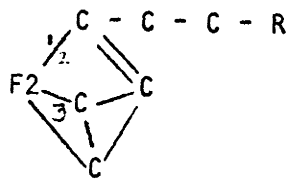
Relative to the above indexings,

U(H) = $\{(1,2,3,4),(2,1,3,4),(1,2,4,3),(2,1,4,3)\}$ and

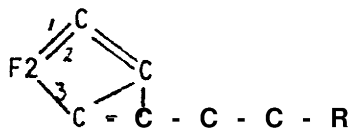
U(G) = $\{(1,2,3,4),(2,1,3,4),(3,2,1,4),(1,3,2,4),(2,3,1,4),(3,1,2,4)\}$.

The canonical set of double coset representatives for U(H) and U(G) in S_4 is $\{(1,2,3,4),(1,3,4,2)\}$. These representatives correspond, respectively, to the following structures G':

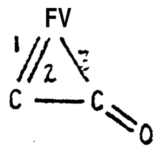
G_1' :



G_2' :



The graph H corresponding to F2 is again:



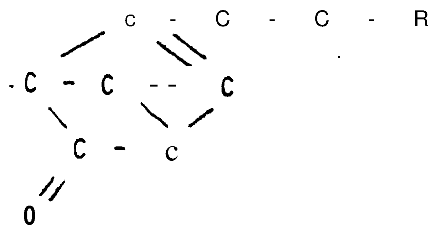
The associated groups are:

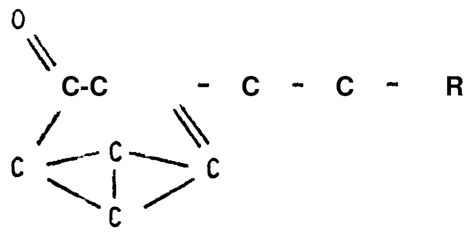
U(H) = $\{(1,2,3),(2,1,3)\}$,

U(G_1') = $\{(1,2,3),(1,3,2)\}$,

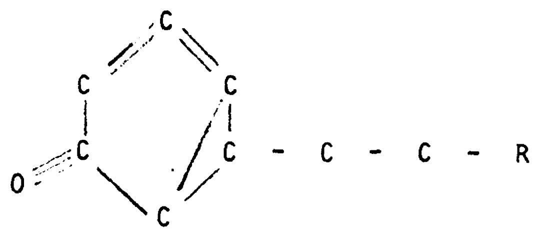
U(G_2') = $\{(1,2,3),(2,1,3)\}$.

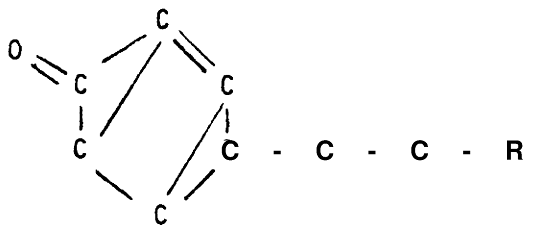
The canonical set of double coset representatives for U(H) and U(G_1') in S_3 is $\{(1,2,3),(3,1,2)\}$. The corresponding structures are:





The canonical set of double coset representatives for U(H) and $U(G_2')$ in S_3 is $\{(1,2,3),(1,3,2)\}$. Thses correspond to the structures:





Using our programs, we have determined all embeddings of the fragments F1 and F2. There are 46 distinct structures based on the given atom set and containing the non-overlapping fragments F1 and F2 where one F1 .has one additional internal bond and 68 distinct structures where F1 has no additional internal bonds.

Appendix II. Proof of the correctness of the double coset algorithms.

Lemma 1. Let $\pi \in S_W$ be generated by the algorithm of Section 3.4. Then:

a) $\pi \ll A\,T$.

b) $\pi \ll \pi B$.

c) $\pi(1) \leq \alpha\pi\beta(1)$ for every $\alpha \in A$, $\beta \in B$.

Proof. a) Choose any $\alpha \in A$. Then $\alpha(\pi(1))$ is in the A orbit of $\pi(1)$. By choice of $\pi(1)$, $\pi(1)$ was least in this orbit. Hence $\pi(1) \leq \alpha\pi(1)$. Assume $\pi(t) \leq \alpha\pi(t)$ for $1 \leq t \leq j-1 < W-1$. If for any t in this range, $\pi(t) < \alpha\pi(t)$, then $\pi \ll \alpha\pi$. Thus, we may assume that $\pi(t) = \alpha\pi(t)$, $1 \leq t \leq j-1$, and, hence, $\alpha \in S = \mathrm{STAB}_A(\pi(1), \ldots, \pi(j-1))$. Since S is a subgroup of A, every orbit of A is a (disjoint) union of orbits of S. Let IM_j be as in the algorithm, i.e., IM_j is the set from which $\pi(j)$ was chosen. By the design of the algorithm and the above remark, IM_j is of the form

$$\{1, \ldots, W\} \setminus \{\pi(1), \ldots, \pi(j-1)\} \quad \bigcup_{x \in T} (\text{S-orbit of } x),$$

for some subset T of $\{1, \ldots, W\}$. Assume $\pi(j) > \alpha\pi(j)$. Since $\pi(j)$ was chosen as least in IM_j, $\alpha\pi(j)$ is not in IM_j. Hence either $\alpha\pi(j)$ is in $\{\pi(1), \ldots, \pi(j-1)\}$ or $\alpha\pi(j)$ is in S-orbit of x for some $x \in T$. However, both of these alternatives contradict the assumption that α is in S. Thus $\pi(j) \leq \alpha\pi(j)$ and, by induction, $\pi(i) \leq \alpha\pi(i)$ for $1 \leq i \leq W-1$. Hence $\pi \ll \alpha\pi$.

b) By the design of the algorithm, $\pi(j) \geq \mathrm{MAX}\{\pi(x) \mid j \in O_x,\ x < j\}$. Hence, by Sim's Lemma (Lemma 4, Section 3.4), $\pi \ll \pi B$.

c) By choice, $\pi(1)$ is the least element in some A_k = A orbit of (1). For $\alpha \in A$ and $\beta \in B$, let A_t be that A orbit containing $\alpha\pi\beta(1)$. Then $\pi\beta(1) \in A_t$. Also $\beta(1) 0_1$ = B -orbit of 1. Now, if $k = 1$, $\pi(1) = 1$ and $\pi(1) \leq \alpha\pi\beta(1)$. If $k > 1$, then $\pi\beta(1) \notin A_1 \cup \ldots \cup A_{k-1}$. Hence $t \geq k$ and $\pi(1) = \mathrm{MIN}\{x \in A_k\} \leq \mathrm{MIN}\{x \in A_t\} \leq \alpha\pi\beta(1)$.

Lemma 2. Let $\pi \in S_W$ satisfy:

a) $\pi << A\pi$.

b) $\pi << \pi B$.

c) $\pi(1) \leq \alpha\pi\beta(1)$ for every $\alpha \in A$, $\beta \in B$.

Then π is generated by the algorithm of Section 3.4.

Proof. Since $\pi << \pi B$, by Sim's Lemma $\pi(j) \leq \pi(t)$ for any j and t such that $t \in 0_j$, $t \neq j$. Also, $t \in 0_j$, $t \neq j$ implies $t > j$. Hence $\pi(j) > SB_j$, $2 \leq j \leq W$.

Since $\pi << A\pi$, $\pi(1)$ must be the least element in A_k = A - orbit of $\pi(1)$. Hence $\pi(1) = P_1$ for some pass of the algorithm. Assume $\pi(s) = P_s$, $1 \leq s < j < W$ for some pass of the algorithm. Now $\pi << A\pi$ implies π (j) is the least element in U = $\mathrm{STAB}_A(P_1, \ldots, P_{j-1})$ - orbit of $\pi(j)$. Also, (c) implies that $\pi(j) \in A_k \cup \ldots \cup A_T$ if j is in B-orbit of 1. Thus, on the initial pass to select P_j, $\pi(j) \in IM_j$. Since IM_j is only decreased by some orbit of $\mathrm{STAB}_A(P_1, \ldots, P_{j-1})$ on any pass, W(j) is eliminated from IM_j only when all of U is. Now IM_j is used as the selection set for P_j until $IM_j = 0$. Thus an element from U must be selected on some pass; and, since $\pi(j)$ is least in U, it must be the first element of U so selected. Moreover, since $\pi(j) > SB_j$, $\pi(j)$ is not rejected. Hence, by induction, at some pass $P_j = \pi(j)$, $1 \leq j \leq W$ unless $P_1 = \pi(1), \ldots$ $P_{W-2} = \pi(W-2)$, P_{W-1}, $P_W \neq \pi(W)$, occurred earlier and was rejected because $P_W < SB_W$. In this case we would have $P_{W-1} = \pi(W)$, $P_W = \pi(W-1)$ and

$\pi(W) < \pi(W - 1)$ since the algorithm generates the permutations in lexicographical order. But then $\pi(W) < \pi(W - 1) < SB_W$ which is impossible.

Appendix III. Symmetry group generation.

Let G be a node indexed graph, x a node of G and $Stab_x$ the subgroup of the topological symmetry group of G stabilizing x. The following algorithm is used by our program to generate, as node index permutations, a subset S of $Stab_x$ maximal with respect to the property that distinct elements of S induce distinct permuations of the edges with endpoint x.

<u>ALGORITHM.</u>

1) Place the node x in a singleton class and classify the remaining nodes by weight according to the scheme given in Section 3.3.
2) If all the node classes are singleton classes, then output the identity permutation and exit.
3) Order the node classes so that
   a) The singleton classes come first.
   b) Following the singleton classes are the non-singleton classes which contain nodes neighboring node x listed in non-decreasing size.[1)]
   c) Following the neighbor classes of (b) are the remaining non-singleton classes listed in non-decreasing size.
4) Reindex the nodes so that node <u>a</u> preceeds node <u>b</u> if and only if the class containing node b does not preceed the class containing node a.

[1)] Our weighting scheme is such that either all the nodes in a given class neighbor x or no node in the class neighbors x.

5) Execute (for effect) with initial parameter value START the recursive procedure given by the SAIL program listed below where the external variables are initially defined as follows:

a) NUMBEROFNOOES = number of nodes in G.

b) START = number of singleton classes + 1.

c) STOP = total number of nodes in singleton classes or neighbor classes

d) ADJACENCYMATRIX [I,J] is the $(i,j)^{th}$ entry in the adjacency matrix for G relative to the node indexing formed in step (4).

e) LOWBOUND [J] is the least index of the nodes in the J-th class (relative to the class ordering of step 3 and the node indexing of step 4) and UPBOUND [J] is the greatest index of the nodes in the J-th class. Here J $\geq$ START.

f) IMAGE[I] = I for 1 $<$ I $<$ START

g) MAPPED[I] = FALSE for START $\leq$ I $\leq$ NUMBER OF NODES

h) CLASS[I] = class index of the class containing the node with index I.

PROGRAM.

```
RECURSIVE BOOLEAN PROCEDURE PERMUTATION(INTEGER I);
BEGIN "PERMUTATION"
INTEGER J,K;

IF I LEQ NUMBEROFNODES THEN
    BEGIN
    FOR J <--LOWBOUND[CLASS[I]]  STEP 1 UNTIL UPBOUND [CLASS [I]] DO
```

```
                BEGIN "J LOOP"
                IF MAPPED [J] THEN
                     CONTINUE "J LOOP"'
                FOR K <-- 1 STEP 1 UNTIL I-1 DO
                     IF ADJACENCYMATRIX [I ,K] NEQ
                          ADJACENCYMATRIX [J,IMAGE[K]]
                          THEN CONTINUE "J LOOP";
                IMAGE [I] <-- J;
                MAPPED [J] <-- TRUE;
                IF PERMUTATION (I+1) AND I > STOP THEN
                     BEGIN
                     MAPPED [J] <-- FALSE;
                     RETURN (TRUE) ;
                     END
                ELSE
                     MAPPED [J] <-- FALSE;
                END "J LOOP";
          RETURN (FALSE) ;
          END
     ELSE
        . BEGIN
          INDUCED PERMUTATION; COMMENT:   INDUCED PERMUTATION IS AN EXTERNAL
               PROCEDURE WHICH COMPUTES AND STORES THE PERMUTATION OF THE EDGES
               OF G WITH ENDPOINT X INDUCED BY THE NODE INDEX PERMUTATION
               I --> IMAGE [I-J;
          RETURN (TRUE) ;
          END;
     END "PERMUTATION";
```