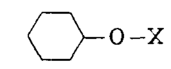
in which either a hydrogen atom, or methyl group, or phenyl group may be bonded to the oxygen atom. This multiple structure is stored in the library with all three substituents attached to the oxygen atom, as shown in Figure 15b. To find an exact match with any one of the three compounds represented, the subgraph algorithm is employed.

(a)



X may be H, CH_3, or C_6H_5

(b)

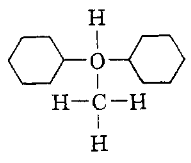


Figure 15.—(a) A Markush compound and (b) its representation.

The algorithm has been programmed for the IBM 7090. The program itself requires about 7000 words of storage and can accomodate graphs with up to 200 nodes. In a limited test of the system, 50 compounds were chosen from Beilstein to form the library. Each compound had exactly 50 atoms and contained only the elements carbon, hydrogen, oxygen, and nitrogen; only six different molecular formulas were represented. The compounds were selected in this manner to utilize the full power of the algorithm frequently. All the compounds were reused as queries so that the algorithm was employed 1275 times. The time to determine that two compounds were not isomorphic was, in 85% of the cases, less than 0.5 msec.; the longest time was about 100 msec. The time to compute a complete isomorphism ranged from 3 to 13 sec., with an average of about 7 sec. When all hydrogen atoms were removed from the compounds (so that each compound had approximately 30 atoms), the time was reduced to the range of 0.8 to 4 sec., with an average of 2 sec. No backtracking was ever required.[6]

ACKNOWLEDGMENT

The author wishes to thank Professor Gerard Salton, Mr. Joseph Rocchio, and Dr. Robert Wall of the Computation Laboratory of Harvard University for their many valuable discussions and suggestions.

(6) A revised and more efficient program for the IBM 7090 is available from the author.

A Chemical Structure Storage and Search System Developed at Du Pont*

D. J. GLUCK
E. I. du Pont de Nemours and Co., Wilmington, Delaware
Received March 23, 1964

INTRODUCTION

As early as 1961, we in the Engineering Department of Du Pont recognized the need for a better system for recording chemical structure information for storage and subsequent retrieval. We believed that current methods and the then current development of notation systems would not completely serve our chemists' long range chemical identification needs.

Accordingly, we studied and then developed a chemical structure storage and search system. Huber[1] gave a good review of the various approaches and applications. To use his terminology, our system is topological coding.

Our initial investigation led to singling out the following needs for such a system (Figure 1).

Of primary concern is that a system provide low cost processing of large files of compounds. To serve future needs for any centralized chemical registry, a system would have to be able to process efficiently upward of 500,000 compounds.

Associated with these large files would have to be low cost input, preferably clerical, which could at sometime be converted to a mechanical operation if such were economically justified.

For long-term utility the system must permit unlimited substructure or generic searches, not restricted by preestablished groupings.

At the same time the system should be highly efficient for simple searches, particularly searches for specific compounds.

Low Cost Processing of Large Files
Low Cost (Clerical) Input
Unlimited Sustructure Searches
Simple Searches—Complete Compounds
One Form/Structure
No Ambiguities

Figure 1.—Chemical structure system needs.

* Presented before the Division of Chemical Literature, 147th National Meeting of the American Chemical Society, Philadelphia, Pa., April 6, 1964.
(1) M. L. Huber, *J. Chem. Doc.*, 5, 4 (1965).

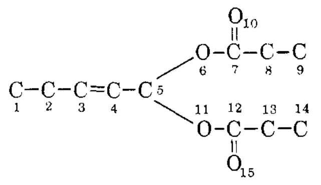
Regardless of how the structures are prepared for *input*, the system should produce one and only one form for each.

Finally, there should be no ambiguities (multiple compounds with the same form in the file).

During the last two years we have attempted to develop a system which fulfills these needs. The result of our effort is the topological system which was described briefly (*Chem. Eng. News*, Dec. 9, 1963) and which will be described in more detail in this paper.

INPUT PROCEDURES

Input Procedures.—Input to this system is entirely clerical (Figure 2). Starting with a drawn chemical compound structure, the clerk is asked to number each atom of the structure in any convenient manner. There are no rules for numbering the atoms of the structure. The only requirement is that each atom receives a different number at input. The clerk then lists the atom numbers and indicates next to each atom number the appropriate element symbol. Next, a simple code representation for each bond is entered on the same line as one of the two atoms involved in that bond along with the atom number for the other. *Atom No. 1* being bonded by a single bond to *Atom No. 2* is indicated by a *Bond Type 1* to *Atom No. 2* on the first line. *Atom No. 7* being single bonded to *No. 8* and double bonded to *No. 10* is similarly indicated. To simplify input each bond is entered only once. The computer system will supply the mirror image of the input

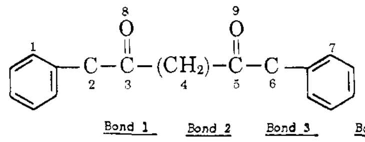


Retrieval Number	Atom No.	Atom Code	Group	Bond 1 Type	Bond 1 Atom No.	Bond 2 Type	Bond 2 Atom No.	Bond 3 Type	Bond 3 Atom No.	Bond 4 Type	Bond 4 Atom No.	Bond 5 Type	Bond 5 Atom No.
1234567	1	C		1	2								
	2	C		1	3								
	3	C		2	4								
	4	C		1	5								
	5	C		1	6	1	11						
	6	O		1	7								
	7	C		1	8	2	10						
	8	C		1	9								
	9	C											
	10	O											
	11	O		1	12								
	12	C		1	13	2	15						
	13	C		1	14								
	14	C											
	15	O											
	11	C	9										
	4	O	9										
	18	H	9										

Figure 2.

entry. Hydrogen atoms are not included in the input unless they are involved in some type of special bonding which will be discussed later. Normal hydrogen counts will be generated by the computer program. To complete the input sheet the clerk adds the molecular formula including hydrogens and labels it *Group 9*. This provides the computer with an adequate check on the accuracy of the input coding. The completed input work sheets are keypunched and verified.

Two input short cuts have been developed which greatly reduce the clerical effort. Both are illustrated on Figure 3.



Retrieval Number	Atom No.	Atom Code	Group	Bond 1 Type	Bond 1 Atom No.	Bond 2 Type	Bond 2 Atom No.	Bond 3 Type	Bond 3 Atom No.	Bond 4 Type	Bond 4 Atom No.	Bond 5 Type	Bond 5 Atom No.
1234568	1	Q	1	1	2								
	2	C		1	3								
	3	C		1	4	2	8						
	4	C	2	1	5								
	5	C		1	6	2	9						
	6	C		1	7								
	7	Q	3										
	8	O											
	9	O											
	8	RP	2										
	24	C	9										
	2	O	9										
	30	H	9										

Figure 3.

If there is a benzene ring with a nonhydrogen attachment in only one position, this can be given a single atom number, an *Element Code Q,* and a group number. The computer will generate the necessary six carbon atoms and the appropriate bonds. If the compound contains some substructural part, like a CH_2, that repeats in a sequence a specified number of times, the group can be coded for input only once and marked with a group number. Below opposite the same group number would be indicated that this was a repeat, *RP*, and the number of times that the structural piece repeated. It should be emphasized that these are only coding short cuts; the computer will generate the same representation whether or not they are used.

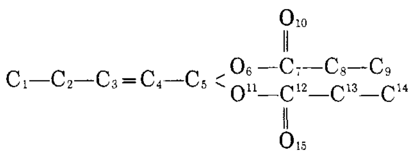
We attempted to determine how well this input procedure could be handled clerically. After one-half hour of explanation, clerks with no previous chemistry background, from one of the companies which supplies temporary clerical assistance, were able to prepare 2800 compounds for input in 258 hr., including the recopying of the structure diagram. This represents an out-of-pocket input cost of 19 cents per compound. Error rates were extremely low.

Input Computer Programs.—From this point the input procedure is handled entirely by the computer. The objectives of the input computer programs are to (Figure 4): (1) verify the accuracy of input and reject all compounds that contain errors; (2) develop a canonical form of the compound, that is, one unique representation regardless of how it was originally prepared for *input* (this canonical form, we believe, is unique to our system, and provides maximum efficiency in file maintenance and searching procedures); (3) reduce the canonical form to a compact list for efficient tape processing every time the file is used in updating or search; (4) eliminate duplicates of compounds already in the file by matching on the list equivalent of the canonical forms.

1. Error Checks
2. Canonical Form
3. Compact List for Storage
4. No Duplicates in File

Figure 4.—Input programs.

After the input cards are read by the computer, the mirror images of the bonds are created and all repeated groups and benzene rings are expanded so that the compound representation inside the machine looks like the contents of Figure 5. This is simply the input data from the work sheet and the expansion. Extensive error checking of the compound takes place at this point. Of



		Bond 1		Bond 2		Bond 3	
Atom No.	Code	Type	No.	Type	No.	Type	No.
1	C	1	2				
2	C	1	3	1	1		
3	C	2	4	1	2		
4	C	1	5	2	3		
5	C	1	6	1	11	1	4
6	O	1	7	1	5		
7	C	1	8	2	10	1	6
8	C	1	9	1	7		
9	C	1	8				
10	O	2	7				
11	O	1	12	1	5		
12	C	1	13	2	15	1	11
13	C	1	14	1	12		
14	C	1	13				
15	O	2	14				

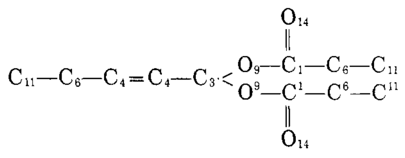
Figure 5.

the error types listed (Figure 6) all but the last two are strictly procedural. *Valence exceeded on an atom* and *molecular formula error* require referral to a machine-stored table of valences. It includes the reasonably common valence forms; however, exceptions can be introduced by indicating a bypass of these two tests.

Bad Atom Code
Bad Bond Code
Duplicate Structure Entry
Duplicate Atom Number
Bond to Unspecified Atom
Atom Bonded to Itself
Unconnected Atom
More than One Piece
Group Error
Valence Exceeded on an Atom
Molecular Formula Error

Figure 6.—Error checks.

After the compound has been checked out for errors, it is ready for the generation of the canonical form. Stated briefly, this program uniquely positions each atom of a compound within the compound representation according to its number of bonds, element code, type of bonds, and the positions in the structure of the atoms to which it is bonded. Perhaps we had best follow a simple example (Figure 7).

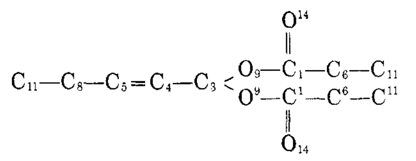


		Elem.	Bond 1		Bond 2		Bond 3	
Atom no.	Orig.	code	Type	No.	Type	No.	Type	No.
1	7	C	2	14	1	6	1	9
1	12	C	2	14	1	6	1	9
3	5	C	1	4	1	9	1	9
4	3	C	2	4	1	6		
4	4	C	2	4	1	3		
6	2	C	1	4	1	11		
6	8	C	1	1	1	11		
6	13	C	1	1	1	11		
9	6	O	1	1	1	3		
9	11	O	1	1	1	3		
11	1	C	1	6				
11	9	C	1	6				
11	14	C	1	6				
14	10	O	2	1				
14	15	O	2	1				

Figure 7.

The atoms of the input representation with mirror images are reordered by the number of bonds indicated for each. Atoms of the same number of bonds are sorted by element code according to a pre-established hierarchy of elements, resulting in carbon sorting ahead of oxygen in the example. Atoms of the same number of bonds and element are ordered by the magnitudes of their bond codes, *i.e.*, single *vs.* double *vs.* triple, etc. These sort sequences are entirely arbitrary and could be changed around in any fashion. The important consideration is that they must remain constant once a system is started if the output is to be canonical.

If you look at the example, you will see that some of the atoms gained unique identification from this initial ordering—*Atom No. 3* for example. Others are tied and given the same ranking number. For example, there are three *No. 11* atoms and the next number is *14.* The atom numbers in the bond columns are the newly assigned rank positions. The two *Atoms No. 4* have different atom ranks associated with their single bonds. The iterative procedure which follows the initial ordering breaks ties according to the magnitudes of the atoms to which the tied atoms are bonded. In Figure 8, one of the *No. 4 Atoms* has been relabeled *No. 5,* and all of its entries in the bond columns have been similarly changed.

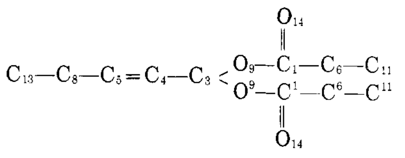


			Bond 1		Bond 2		Bond 3	
Atom no.	Orig.	Elem. code	Type	No.	Type	No.	Type	No.
1	7	C	2	14	1	6	1	9
1	12	C	2	14	1	6	1	9
3	5	C	1	4	1	9	1	9
4	4	C	2	5	1	3		
5	3	C	2	4	1	8		
6	8	C	1	1	1	11		
6	13	C	1	1	1	11		
8	2	C	1	5	1	11		
9	6	O	1	1	1	3		
9	11	O	1	1	1	3		
11	1	C	1	8				
11	9	C	1	6				
11	14	C	1	6				
14	10	O	2	1				
14	15	O	2	1				

Figure 8.

This iterative process of reordering according to the new rank of the atoms in the bond columns continues until all atoms are uniquely ranked, in which case the compound is in canonical form, or until no further reordering is possible and ties still remain. The latter case indicates that a symmetry was encountered. In this example (Figure 9) there is symmetry around the carbon atom now labeled *No. 3.* Since the tied parts must be truly symmetrical, or else their differences would cause reordering, we will always get the same canonical form regardless of which of the tied parts is placed first. Therefore, the computer program arbitrarily breaks the first tie (Figure 10), in this case *Atoms 1 and 2,* relabels all entries of the tied members in the bond list, and begins the iterative reordering process again. If another symmetry is encountered, the arbitrary tie-breaking procedure is repeated. Eventually all of the atoms are uniquely positioned in the list. Figure 11 is the canonical form of the compound, based entirely on the compound structure and independent of any input conventions.

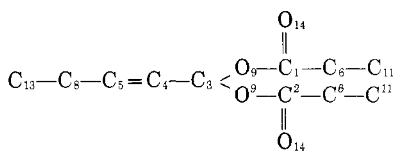
Our study effort was directed toward designing the most efficient possible system. Examination of the canonical



			Bond 1		Bond 2		Bond 3	
Atom no.	Orig.	Elem. code	Type	No.	Type	No.	Type	No.
1	7	C	2	14	1	6	1	9
1	12	C	2	14	1	6	1	9
3	5	C	1	4	1	9	1	9
4	4	C	2	5	1	3		
5	3	C	2	4	1	8		
6	8	C	1	1	1	11		
6	13	C	1	1	1	11		
8	2	C	1	5	1	13		
9	6	O	1	1	1	3		
9	11	O	1	1	1	3		
11	9	C	1	6				
11	14	C	1	6				
13	1	C	1	8				
14	10	O	2	1				
14	15	O	2	1				

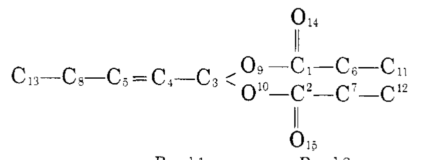
Figure 9.

form indicates that it is not an efficient representation of the compound. Atoms are bonded to other atoms scattered throughout the representation. All bond entries are recorded twice. There is no convenient way to put the canonical form out on tape in an array without dedicating a lot of space for nonexistent data. To correct these inefficiencies, an algorithm was devised for reordering the compound into a compact list. Because this reordering is entirely algorithmic, a given canonical form will always produce the same compact list; thus, it becomes a canonical list. Figure 12 contains the compact list for the example compound.



			Bond 1		Bond 2		Bond 3	
Atom no.	Orig.	Elem. code	Type	No.	Type	No.	Type	No.
1	7	C	2	14	1	6	1	9
2	12	C	2	14	1	6	1	9
3	5	C	1	4	1	9	1	9
4	4	C	2	5	1	3		
5	3	C	2	4	1	8		
6	8	C	1	1	1	11		
6	13	C	1	2	1	11		
8	2	C	1	5	1	13		
9	6	O	1	1	1	3		
9	11	O	1	2	1	3		
11	9	C	1	6				
11	14	C	1	6				
13	1	C	1	8				
14	10	O	2	1				
14	15	O	2	2				

Figure 10.



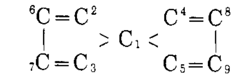
Atom no.	Orig.	Elem. code	Bond 1		Bond 2		Bond 3	
			Type	No.	Type	No.	Type	No.
1	7	C	2	14	1	6	1	9
2	12	C	2	15	1	7	1	10
3	5	C	1	4	1	9	1	10
4	4	C	2	5	1	3		
5	3	C	2	4	1	8		
6	8	C	1	1	1	11		
7	13	C	1	2	1	12		
8	2	C	1	5	1	13		
9	6	O	1	1	1	3		
10	11	O	1	2	1	3		
11	9	C	1	6				
12	14	C	1	7				
13	1	C	1	8				
14	10	O	2	1				
15	15	O	2	2				

Figure 11.

The compact list algorithm works like this. *Atom No. 1* of the canonical form always becomes *Atom No. 1* of the list. All of the atoms connected to *No. 1* follow next in sequence according to their rank in the canonical form. *Atom No. 2* was *6, Atom No. 3* was *9,* and *Atom No. 4* was *14.* The bonds shown are the bonds to *No. 1.* After the atoms connected to *No. 1* are listed, the atoms connected to *No. 2* are listed. In this case there is a single atom connected to *No. 2* which is listed as *No. 5.* Its bond is the bond to *No. 2.* Next the atoms connected to *No. 3* are listed (for the example, a single atom which becomes *No. 6*). *Atoms 4 and 5* introduce no new atoms to the list. Therefore, the next atoms on the list are those connected to *Atom No. 6.* They become *No. 7 and 8* according to their rank in the canonical form. This process is continued through the compound resulting in three columns of information, besides atom number, which is superfluous, for the complete recording of a compound structure.

As you can see, the compact list has one less bond than the number of atoms. This is sufficient for complete compound representation only when the structure contains no rings. Each ring in a structure introduces an additional bond which must be considered. A supplemental ring list is introduced. In the example (Figure 13), the position of *Atom No. 7* in the list comes from *Atom No. 3.* When the algorithm gets to the point of making the list from *Atom No. 6,* it encounters *No. 7* which has already been numbered. This indicates a ring. It is shown in the supplemental ring list as a ring *from 6 to 7* with a single bond. The same applies to the other ring *from 8 to 9* with a single bond.

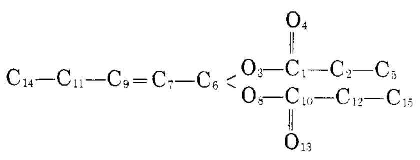
The compact list is just that—compact. The list representation can be stored with only three fields of data for each atom and each ring. The atom data are element code, starting atom number for its list, and bond from its



Atom no.	Elem. code	Atom to list	Bond from orig.
1	C	2	0
2	C	6	1
3	C	7	1
4	C	8	1
5	C	9	1
6	C	0	2
7	C	0	2
8	C	0	2
9	C	0	2

Ring no.	Ring from	Ring to	Ring bond
1	6	7	1
2	8	9	1

Figure 13.—Compact list.



Atom no.	Canonical form						
	Elem. code	Bond 1		Bond 2		Bond 3	
		Type	No.	Type	No.	Type	No.
1	C	2	14	1	6	1	9
2	C	2	15	1	7	1	10
3	C	1	4	1	9	1	10
4	C	2	5	1	3		
5	C	2	4	1	8		
6	C	1	1	1	11		
7	C	1	2	1	12		
8	C	1	5	1	13		
9	O	1	1	1	3		
10	O	1	2	1	3		
11	C	1	6				
12	C	1	7				
13	C	1	8				
14	O	2	1				
15	O	2	2				

Atom No.	Compact list		
	Elem. code	Atom to list	Bond from orig.
1	C	2	0
2	C	5	1
3	O	6	1
4	O	0	2
5	C	0	1
6	C	7	1
7	C	9	1
8	O	10	1
9	C	11	2
10	C	12	1
11	C	14	1
12	C	15	1
13	O	0	2
14	C	0	1
15	C	0	1

Figure 12.

originator. Ring data are atom number from, atom number to, and bond. Our test work indicates that with high-density tape drives, we will get approximately 70,000 compounds on a standard 2400-ft. magnetic tape. When searching is described, the reader will see how valuable this compound squeezing is. Much of the computer search cost is in tape reading associated with unwanted compounds. The tighter the packing, the lower the cost of tape passing. In large files the advantages here are significant.

All of these input programs run a lot faster on the computer than the time it takes to explain them. They have been completely programmed and tested on the IBM 7040 computer. A sample population of 2800 compounds, which represent a cross section of Du Pont's interests, were used in the test work. The average number of iterations of the ordering program for the test file was nine. The complete input program required 63 min. of computer time for the 2800 compounds, or figuring at second shift computer rates, less than $0.04 per compound.

System Exceptions.—Thus far in the description of the system examples have been used with simple single, double, and triple covalent bonds and elements from the periodic table. It was apparent early in our investigation that for a system to serve Du Pont's needs or the needs of the chemical industry, it had to be able to handle the full range of chemistry which cannot be described that simply. Considerable effort in the development of our system has gone into defining structural specialties, and devising coding conventions which will permit simple recognition and description at input and adequate handling by the machine system. A partial list of these specialties and their treatment is given (Figure 14).

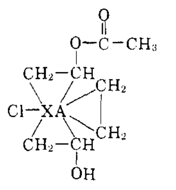
Aromatic Resonance	Special Bond
Ionic Bonds	Special Bonds
Polar Covalent Bonds	Special Bonds
Coordinate Bonds	Special Bond
Delocalized Bonds	Special Atoms
Olefin Complexes	Special Bond
Free Radical	Special Atom
Pi Complexes	Special Atom
Isotopes	Special Atoms
Stereochemistry	Special Bonds
Bridged Atoms	Special Bond

Figure 14.—Coding conventions.

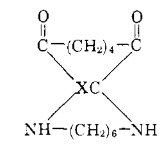
A separate paper would be required to describe each of these system specialties in detail and to develop the reasoning that justifies our choice of coding conventions. The problem can be illustrated with one simple example—aromatic resonance. If a normal aromatic ring with two substitutions on adjacent atoms is pictured, it is realized that with normal structure drawing conventions these atoms could be connected by either a single or double bond indicator. If we labeled the bonds of the ring *Bond Types 1 and 2,* we could get two canonical forms for the one structure, which is a direct contradiction to one of our objectives. A special bond code, arbitrarily numbered 6, with *1½ valence* was invented for the connections around the rings.

Similar conventions have been developed as shown. For example, the clerk does not code normal hydrogens. Even if she does, there would be no harm done because the computer program will drop them from the canonical form. If, however, a hydrogen is involved in any of the special bonding, such as a bridged hydrogen, it must be coded for input.

There is one large group of system specialties which require special mention, because of the complexity of the problems it presents and the significantly large role it plays in modern chemistry. This group is polymers. The problem comes from the chemists' inability, in many cases, to describe accurately the structural representation of a specific polymer chain, and the undesirability from a system's standpoint of handling something as long and redundant as a polymer chain even if the chemist could. This system's solution to polymers, which we believe is a novel solution, is to input each monomer in the form that it would exist in the polymer chain all linked to a dummy central atom. End groups, if any are known to exist, can also be linked to this dummy atom (Figure 15).



Unlimited Attachments



Limited Attachments

Figure 15.—Polymers.

The system has two dummy atoms for polymers. The first, corresponding to addition polymers, permits unlimited attachment across the dummy atom to any other connected atom of the polymer. Thus, in a search through the example polymer, the sequence of connected atoms could be from the CH_2 of the vinyl alcohol to the *XA* and back out to any of seven atoms including the same CH_2. The other polymer atom, *XC*, is used for condensation type polymers and other cases where alternating sequences of unlike units are required. With *XC*, the exit from the dummy atom must be to an element different from that of the entry to the dummy. In the example, if the sequence of connection is from one of the carbons to *XC*, it must then go to one of the nitrogens. In this particular case, because both monomeric units are symmetrical, a specific sequence is defined.

As stated before, we believe this treatment of polymers is novel and we welcome comments about it. It offers us many advantages which other methods do not. The canonical representation can be calculated just as with any other compound. Storage representation is compact. Most important, substructure searching for any sequence of atoms within a polymer chain is no more difficult than a normal substructure search.

Search.—This discussion of polymer searching leads us very nicely into the part of the system concerned with searching the structure file. Searches can be broadly categorized into two types: searching for complete compounds and substructure searching.

For this discussion, searches for complete compounds can be disposed of quickly. The canonical form and compact list algorithms are applied to the question compounds in the same way they were to the compounds in the file. Therefore, the same canonical list forms are generated for the questions as for their file counterparts.

The search consists of a simple line-by-line matching of lists. This is the simplest search any of the published full structure systems offer.

Substructure searching is not that simple. Because the position of an atom in the canonical list is dependent upon the entire compound, it is possible that something in the compound, irrelevant to the substructure, will cause the relative positions of the atoms of the substructure to be different in the full compound and in the substructure alone. This necessitates iterative searching.

Iterative searching proceeds as shown in Figure 16. The substructure of the question is put in the same list form, using the same numbering outward rules as the file compounds. The algorithm tests using *Question Atom No. 1* until it finds a satisfactory match or until it hits the last file atom in which case the file compound is not an answer. If the match is found for *Question Atom No. 1*, the algorithm tries to find a match for *Question Atom No. 2*. This is accomplished by testing the file atoms connected to the file atom previously matched with *No. 1*. These connections can be the backward connection or the *from atom*, the atoms of the *to list*, or atoms in the *ring list*. If a match is found for *No. 2*, the question atom number is increased by one to *No. 3* and the testing is repeated. Always we are testing file atoms connected to the file atom which is a match for the question atom from which the present question atom derived its position in the list. Testing includes comparison of element symbols, valence state, and bonds of attachment. This operation is repeated for the entire question list until a satisfactory match has been found for the last question atom or no match can be found for the atom currently being tested. In the former case, a match for the last question atom, we have found a successful answer to the question. If we can find no match for a question atom, it is because the assigned matches for the previous question atoms do not lead to a satisfactory answer. Some of these earlier matches might be the first of several file atoms connected to the same atom. This means the alternative branches must be tested. The algorithm backs up the question atom one position and looks for another candidate. From this point the search might again move forward on the question list or might back up again. If the search substructure is not contained in the compound, eventually backing up will bring us back to the *Question Atom No. 1*, and we will reach the bottom of the file atom list without a satisfactory match.

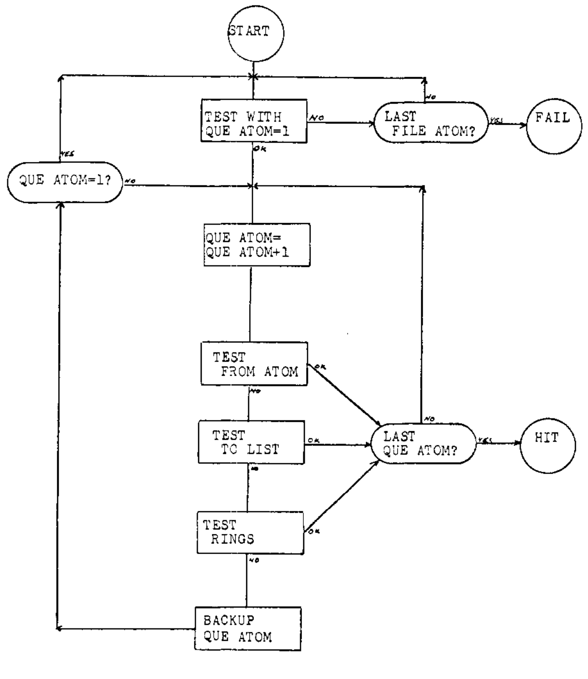


Figure 16.—Search algorithm.

To the noncomputer-oriented person, this search algorithm might look complicated, but it really is not. It is simple, straightforward, and, as our test results indicate, fast. Unlike the search for the complete compound, the atoms in the substructure list can be in *any* order provided that the rules for numbering within the list are followed. This means that an experienced questioner can put the atom or atoms and bonds which will make it most difficult to find the substructure at the top of the question list. In this way, most of the comparing through the file is done with *Question Atom No. 1* and a minimum of backing up.

This search algorithm is fast. In our test work we searched 2800 compounds for 50 searches in one pass in less than 30 min. of IBM 7040 computer time. This means that the computer was performing full iterative searches at the rate of about 4000 per minute. Fast as that is, the cost of performing full iterative searches on files of several hundred thousand compounds would still be considerable. Generally, there is some limiting characteristics in each question, which if tested for separately could eliminate the necessity for iterative searching large portions of the file. This can be accomplished two ways: by segmenting the file and only searching the part of the file that could possibly contain the question substructure, or by first testing each compound in the file for the limiting characteristic and then searching iteratively only those compounds which successfully passed this test. These two methods should not be confused. The former is file subdivision; the latter is screening. It is anticipated that a large file in full operation will employ both techniques.

Screens.—We are approaching the problem of screens in a slightly different manner from that used by others in the past. We have no preconceived notions as to what will comprise our system of screens. Screen selection will be on a strictly economic basis; *i.e.*, a screen must pay for itself in reducing the amount of iterative searching more than enough to offset the added cost of testing each file item for the screen. Before we look further at the economics of screens, let us consider what possible screening characteristics might be (Figure 17).

Screen(S) = Element
Bond
Element–Bond
Element–(Any Bond)–Element
Bond–(Any Element)–Bond
Element–Bond–Element
Bond–Element–Bond

1. Is S^Q present?
2. Is S_1^Q or $S_2^Q \ldots S_N^Q$ present?
3. Is $S^Q| \leqq | S^F$?
4. Is $(S_1^Q + S_2^Q \ldots S_N^Q) | \leqq | (S_1^F + S_2^F \ldots S_N^F)$?

Figure 17.—Screens.

Screen testing might examine for the presence or absence of a particular element or a particular bond, especially in the case of the dummy bonds which were set up for the exception chemistry, or a screen characteristic might be an element bond pair. More complex but also more limiting screens might be based on a pair of elements connected by any bond or any atom with two specified bonds. More complex still would be an element-bond-element triplet, or a bond-element-bond triplet. You can see that as we go down this list we get closer and closer to a full iterative search and an associated high testing cost.

The screen test itself can take several forms. The first and most simple is to test for the presence or absence of the question characteristic in the file item. Closely related is to test for the presence or absence of a member of a family of characteristics when the family is represented in the question. This has the advantage of taking less space and computer time than a group of individual tests, but the disadvantage that the family members, question and file, may be different. Another way screens may be applied is to test for at least as many of the screen characteristics as are required in total by the question. This is equivalent to testing atom count, for example. Testing by count can be accomplished either by individual counts, or again by family counts with the advantages and disadvantages cited before. It is apparent from this simple description of screens that each of the different types of screen characteristics in conjunction with each of the different screening methods has its own machine cost and probability of screening out unwanted filed items from the iterative search. These determine the economics of a candidate screen.

We are presently evaluating candidate screens using this mathematical model (Figure 18). P_i is the probability that a compound will be screened out of further processing by screen i. The probability that a compound would be screened out by a sequence of N independent screens is given by the expression

$$P = 1 - \pi_{i=1}^{N} (1 - P_i)$$

If C is the cost of an iteration search and L_i is the ratio of screen i cost to iteration cost, the total cost of a system with N screens and a file of F compounds is given by eq. 2 (Figure 18). Our problem is to evaluate an individual screen which can be thought of as the $N + 1$ screen. The change in total cost associated with the $N + 1$ screen is given in eq. 3, and simplified in eq. 4. If you look at eq. 4, you will see that the second half is the simple probability that the file item has successfully passed the first N screens. From this equation we, therefore, conclude that whether a screen is going to save money or cost money is determined by whether the expression $(L - P)$ for that screen is positive or negative. We are currently evaluating candidate screens on this basis. Our evaluation is not sufficiently far along to be able to present a set of selected screens.

P_i = Probability screen i screens out
P = Probability a compound is screened out
N = Number of screens

$$P = 1 - \pi_{i=1}^{N} (1 - P_i) \qquad (1)$$

C = Iteration Cost
L_i = Screen i cost/iteration cost
F = Number of compounds in file
TC = Total cost (screen + iteration)

$$TC_N = \left[\sum_{i=1}^{N} L_i \pi_{j=1}^{i} (1 - P_j) + \pi_{i=1}^{N} (1 - P_i) \right] CF \qquad (2)$$

$$\Delta TC_{N+1} = \left[L_{N+1} \pi_{i=1}^{N} (1 - P_i) + \pi_{i=1}^{N+1} (1 - P_i) + \pi_{i=1}^{N} (1 - P_i) \right] CF \qquad (3)$$

$$\Delta TC_{N+1} = (L_{N+1} - P_{N+1}) CF \pi_{i=1}^{N} (1 - \mathrm{P}_i) \qquad (4)$$

Figure 18.—Screens.

Before leaving the subject of screens, let us consider another point. When should screen information be generated, at input or at search time? The first reaction is to say at input time so that the data are calculated only once for all future question runs. Our analysis has indicated that this conclusion is erroneous. The computer can recalculate the data just about as fast as it can read them. Furthermore, it only has to calculate the screen data needed for a particular batch of questions, which are, far less than would have to be put out on tape initially for all future uses.

Now let us look at the alternative to screens—subdividing the file by categories. To be effective, file categories must be highly user oriented, and the scarcest commodity at this time in systems design is real information on how the user will ask his questions. The saving feature is that subfiles are inefficient with small files because of the extra handling involved, and only begin to pay off with files of upward of two tapes or 140,000 compounds. Since we expect our files to grow concurrently with use, we can wait until we have some user experience before we tackle the problem of file categories.

Economics of System.—While we have not yet made our selection of screen and file categories for an operating system, test searches indicate that combined screening and file subdivisions should reduce the number of compounds to be searched iteratively by about 95%. Using this figure and our actual computer search costs with our test files of 2800 compounds, we have projected the following search economics (Figure 19).

Four simultaneous searches of a file of 100,000 compounds would cost about \$64 in total, or \$16 per search. Five simultaneous searches of a file of 200,000 compounds would cost approximately \$26 each. Searching a file of a half-million compounds with ten simultaneous searches would cost in the neighborhood of \$48 per search. Finally, this last figure should be of interest to any one concerned with the processing of very large files. Fifty simultaneous substructure searches of a file of one million compounds can be run for less than \$30 each. We presently see ways

Searches	Thousands of compounds	Cost/search, \$
4	100	16
5	200	26
10	500	48
50	1000	29

Figure 19.—Search economics.

to reduce these search costs even further by computer hardware modification. Faster tape drives and two channel inputs to the computer should give additional processing speed advantages.

To give you a complete picture of the economics of this system, the input costs are summarized in Figure 20. Input costs are of three types: clerical costs, card preparation including keypunching and verification, and computer costs. An out-of-pocket clerical cost of $0.19 per compound was stated earlier. Based on a keypunch and verification rate of $5.00 per hour, including machine, keypunching our test file ran $0.15 per compound and verification another $0.12 per compound. Finally, the lowest of the input costs, the computer itself, ran less than $0.04 per compound giving a total input cost of approximately $0.50 per compound.

	Cost/compound, $
Clerical costs	0.19
Card preparation:	
Keypunch	0.15
Verification	0.12
Computer input	0.04
Total	0.50

Figure 20.—Input costs.

SUMMARY

Our primary objective was an efficient system. These costs for search and input appear attainable. Let me reiterate that these are real costs which we have experienced with our pilot system, not estimates from a hypothetical system. Much of our system is similar to other work which we have read about in the literature or heard about here today. However, we believe our system gains real cost advantage from its unique features. In summary, the significant features of the system are (Figure 21):

First, the system is fed with clerical input. The clerical approach has been tested and found to be satisfactory.

The system develops a canonical form for each compound, that is, one unique representation without regard for any input.

Clerical Input
Canonical Form
Error Checks
Compact Lists
No Duplicates on File
Full Range of Chemistry
Polymers
Simple Searches—Complete Compounds
Any Substructure Search

Figure 21.—Features of the system.

Input programs, which have been thoroughly tested, include extensive error checking.

The canonical representation of the compound is transformed into a compact list which is the basis for all subsequent computer processing efficiencies.

The compact list is checked against the compounds already on file to prevent the insertion of duplicates.

The full range of exception chemistry of interest to Du Pont has been considered and coding conventions for input and computer storage have been adopted to make the system universally useful.

Of special interest is the treatment of polymers which provides full flexibility and at the same time is efficient and compact.

On the search side of the system, the compact canonical list provides the simplest possible searching for complete structures—simple matching of lists.

The iterative substructure search algorithm has been programmed and tested. By utilizing questions in the same list form as the file items, it is conceptually simple and fast.

Finally, screens are being incorporated into the system, not because they appear useful, but because the ones selected will be economically justified.

Economic justification and operating efficiency are the keynotes of any system. A few microseconds per compound can quickly add up when processing million-compound files. We are disclosing the details here because we believe that Du Pont's interests and the interests of the entire chemical community can best be served by considering the economical features of this system, and at the same time the concept of efficiency, in the design of a major widely used system for storage and retrieval of chemical structures such as that underway at Chemical Abstracts Service. We have waited this long to disclose in order to present a *developed and tested*, efficient system. We look to the American Chemical Society for leadership in this area and have offered it the results of our work for the benefit of the chemical community.