Computer Program for the LINCO* System**

H. BOUMAN
Shell Internationale Research Maatschappij N. V., the Hague, Holland

Received December 16, 1963

INTRODUCTION

In the article,[1] "*L*inearly organized *c*hemical code for *u*se in computer *s*ystems," there was described a method of representing chemical structural formulas which makes it possible to feed this information to computers. The system is based on the idea of treating the formula as a network containing branching points (points where three or more series of bonded atoms other than hydrogen interlock), giving to these points an arbitrary number and writing down the series of atoms between two points of that type ("bridges") together with the series of atoms extending from such branching points but not meeting others ("chains"). For example, glycerol tristearate is treated as a network between four numbered branching points (italicized in Figure 1) connected by bridges (. . . . .) and from which chains (<) extend (Figure 2).

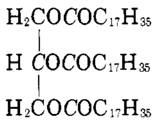


Figure 1.

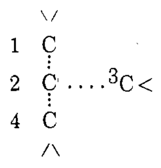


Figure 2.

By typing out (*e.g.*, on a conventional tape-making typewriter) the bridges and chains in the structure according to simple rules, one gets the LINCO notation. First the numbers given are typed out with their atom type (in this case 1234C). Then the bridges and chains come between oblique lines (given once only for each and in an arbitrary sequence), starting with a branching point number, followed by the atoms bonded thereto (neglecting hydrogen bonded to carbon). If another branching point is encountered its number is preceded by an exclamation mark. In this case the complete notation could be

1234C/1=*O*/1C 17/1*O*C!2/2*O*!3/2C*O*!4/3=*O*/3C ˆ17/4=*O*/4C ˆ17

where the indicates that two characters belong together. This information can be fed to a computer and processed by it to form a storage list containing all the information given in the structural formula. In the example the list would be as given in Figure 3 (repeating the branching point atoms each time). Each line in the list is called a term. The parts separated by horizontal lines

1	CC17	
1	C=O	
1	COCC	2
2	CCOC	1
2	COC	3
2	CCOC	4
3	CC17	
3	C=O	
3	COC	2
4	CC17	
4	C=O	
4	COCC	2

Figure 3.

are called sections of the list. The terms in the list, which refer to bridges are those which have branching point numbers (reference numbers) on both sides.

In the LINCO system, various reference numbers can be used to describe the same compound; this produces different storage lists, although the chemical parts of the terms do not change. It is better however, to have a unique unambiguous notation for the reference numbers as well, thus making it possible to represent chemical compounds in a standard way on a standardized storage list.

Thus, although the data in any of the storage lists which can be obtained are sufficient for searching purposes, more rapid searching without computation steps would be possible in certain cases if the list could be transformed into a list not influenced by the way of numbering the branching points (standardization).

This can be done as will be shown below *via* a matrix in which the reference numbers are used as column and row indication and the connections between the points are indicated at the intersections of the relevant rows and columns. In the above example this connection matrix would be as shown in Figure 4.

	1	2	3	4
1		1		
2	1		1	1
3		1		
4		1		

Figure 4.

*The previous name LOCUS has been changed to LINCO for foreign language reasons.

**Presented before the Division of Chemical Literature, 147th National Meeting of the American Chemical Society, Philadelphia, Pa., April 6, 1964.

Standardization.—In order to standardize the notation, the reference numbers which were allotted arbitrarily to the branching points have to be revised. For this purpose each of the branching points is given a relative weight based on the connections in the network. The final numbering depends on these weights.

The procedure is comparable to that used in arranging a large number of articles according to weight. Thus the articles would first be arranged in groups of the same number of tons, ignoring the hundredweights, stones, pounds, etc., in the weight figures; then the groups so obtained would be divided into subgroups of the same number of hundredweights only, and so on, until there was no need to go any further.

Instead of weights, special numbers, known as yardstick numbers, are used; they are calculated afresh for each branching point in each step. Each succeeding step takes into account a further, more detailed, aspect of the network. After each step the total series of groups obtained is renumbered in weight sequence.

Of course, groups containing only one branching point cannot be further subdivided (comparable to the case in which only one article comes in the 5-ton group; its position relative to a number of other articles in other ton groups is not changed by ascertaining that, in fact, it weighs 5.13592 tons). Therefore in many cases only a few steps are required to find the relative weights.

The process is complete when each branching point is alone in its group or when no further yardstick numbers can be calculated. In the latter case every detail of the network has been taken into account, and those branching points which still show no difference in weight are in symmetrical positions.

The machine program for carrying out the above process is outlined at the end of the Appendix, and the reader could pass now immediately to that point. It is, however, difficult to see from the outline what in fact occurs during the program.

To enable the reader to visualize the process, a description is given below in concepts relating to the network as seen by a human being. The program for the machine (exemplified in detail in the central part of the Appendix) does not of course use these visual concepts, but the steps taken in it in the connection matrix have virtually the same effect as those described here in visual terms.

The first four steps are: (1) counting the other branching points to which direct connections exist; (2) as in step 1, but now adding one to the result for all those branching points which show one or more direct connections to themselves (rings, not containing other branching points); (3) counting the direct connections to other branching points which exist in addition to the number of single connections to other branching points (multiple connections); (4) as in step 3, but now adding to this number the total number of rings not containing other branching points.

In order to understand the next steps, the network can be considered as a pyramid-like structure, with the branching point whose yardstick number is being worked out at the top (a detailed example is shown at the beginning of the Appendix). For each branching point a different pyramid has thus to be considered. Rings with only one branching point and also multiple connections are left out for the moment, as the information they can give in this part of the procedure has already been taken into account. From the top of each of the pyramids a number of connections will go to the first level below the top, where the branching points directly connected to the top are found; on the next lower level those branching points are found which are directly connected to the branching points of the previous level but not to the top, etc. Branching points on the same level together form a class.

To get the yardstick numbers for the first class, certain counts are made in succession, each count yielding a yardstick number for the top point of the pyramid in which the count is made. Thereafter similar counts are made for the next class, etc., until the bottom of the pyramid is reached and no further counts are possible.

The counts made are:

(a) The number of connections coming into each of the branching points on the level from above
(b) The number of connections going out from each branching point on the level, either to branching points on the same level or to the next lower level
(c) The number of connections going from each of the branching points on the level to the next lower level only

These counts in short are:

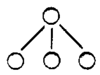
(a) All incoming connections
(b) All outgoing connections
(c) All connections going to the next level

Since each of these counts is made per branching point of the class involved, as many results are obtained as there are branching points in that class. These results are put in a certain order and used together as a yardstick number for the top branching point of the pyramid investigated, to be compared with the similarly obtained yardstick numbers for the tops of the other pyramids.

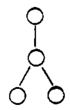
The order in which the results of a count are arranged is governed by the results themselves. As soon as a count gives different results for the branching points in a certain class, these branching points are arranged in subclasses according to increasing results, and the results are noted in subclass order to get the yardstick number. For later counts in the same class the arrangement remains in the order of the subclasses already given; if inside a certain subclass a difference is found between the results for the branching points therein, the subclass involved is subdivided into new subclasses. For counts in subsequent classes the branching points therein get the subclass indications of those branching points in the previous class to which they are connected, which subclass indications are arranged in increasing order per branching point of the subsequent class. In this way any difference in the counts in earlier levels of the pyramid is taken down to lower levels, so that points which are connected to differently connected higher points in the pyramid can be distinguished from one another. One can visualize the use of the subclasses as lines, starting at a certain level between points yielding different count results, which lines go down the pyramid and separate those points which are connected to such different points.

The general result of the process described above is that the top branching points are weighted and grouped in accordance with the "degree of centrality" of their position in the network (*i.e.*, the more new details, about the network, connections, and other branching points that are found in the fewest number of steps through the network—when going from a particular branching point stepwise in all directions—the greater the weight of that branching point is considered to be).

If we now refer to the structure in Figure 2, it will be clear that count 1 (directly connected other branching points) itself will give a preference for one of the branching points over the others: the result of the count for that branching point is 3, while for the others it is 1. The "pyramid" of that particular branching point would have the following appearance.



It is not used, however, as this point is already alone in its group. The pyramid for each of the other branching points has the following appearance.



As they are all alike, further investigation will not give other weights to their tops. In more complicated cases, as illustrated in the Appendix, a difference *will* be found.

Even if the positions in the network are symmetrical, they may not be so in the compound because the atoms in two bridges are not the same, or the chains extending from the branching points in the network are not identical. Therefore, after a first grouping based on the network only has been made, a second grouping, mainly on an alphabetical basis, is made to subdivide the groups. At the same time certain ambiguities which could arise from the neglecting of certain connections (multiple connections and rings) are corrected.

That this may be necessary in certain cases can be seen from an example. Imagine a compound which gives the pyramid (amongst others) shown in Figure 5, in which a

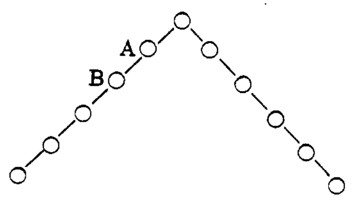
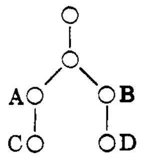


Figure 5.

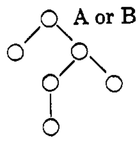
double connection between the points A and B has been left out. The points A and B have already fallen into a group, different from the group entered by the other points at the beginning of the arrangement (count 3), but for the arrangement of the points at a level below B this has no effect. Therefore, the left-hand and right-hand points on a level below B will be treated as equivalent, which is not correct, since the left-hand ones are closer to the "heavier" points A and B than the right-hand points. By using as a yardstick number the group numbers of directly connected other branching points, already given in earlier steps and arranged in ascending order, a correction can be made.

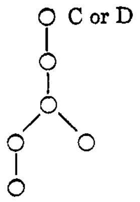
This method, expressed in items 22–23 of the program (Figure 6), of giving group numbers could be used as an alternative to the program given by prescribing, for example in test item 2 of the program, that as soon as more than one group has been formed, steps 22–23 should be applied. As this situation already occurs after the first count (number of connected other branching points) for many compounds, it would shorten considerably the time needed by the computer. The result, albeit unique, would, however, be less systematic, and the "degree of centrality" as a weighting means would be followed less stringently. We believe therefore that the program proposed will yield more satisfactory results for large files.

Thereafter there may still be some ambiguity present: two or more identical groups of atoms may each contain two or more branching points which, although in two different groups, are mutually dependent. For instance, in the structure



where the letters A, B, C, and D indicate an ascending series of reference numbers, the points A and B would come in one group and those marked C and D in another as their "pyramids" are different; *viz.*





If the nature of the bridges and attached chains gives no clue for further division (indicating symmetry), one could number them in two different ways which are not equivalent; *viz.*, $\begin{smallmatrix}A\,B\\C\,D\end{smallmatrix}$ as shown, or $\begin{smallmatrix}B\,A\\C\,D\end{smallmatrix}$. To streamline such cases a choice is made arbitrarily in one of the groups between the available numbers, and the numbers in the other group(s) are derived therefrom. Remaining mirror-symmetrical numbering sequences may look different on paper, but the difference is of no chemical significance and has no effect on the standardized storage list.

A program outline is shown in Figure 6 and an example illustrating how these steps are performed in a matrix is given in the Appendix, as is a more detailed outline.

Consequences.—The LINCO system is a simple method of documenting defined chemical structures, including "general formulas," which can be combined with nonstructural data as described in the original publication.†

†*Stereoisomerism*: The following convention was adopted after the publication of the original article for noting *cis–trans* isomerism, inosite-type isomerism of substituted rings, *d*-type and *l*-type isomerism of sugars, etc. Groups on the same side of the plane of a double bond or of a ring are indicated in the notation by the same figure between brackets and those on the other side by a different figure. Usually (1) and (2) are used indiscriminately for this purpose, but for sugars the groups conventionally written on the left are indicated by (1), and

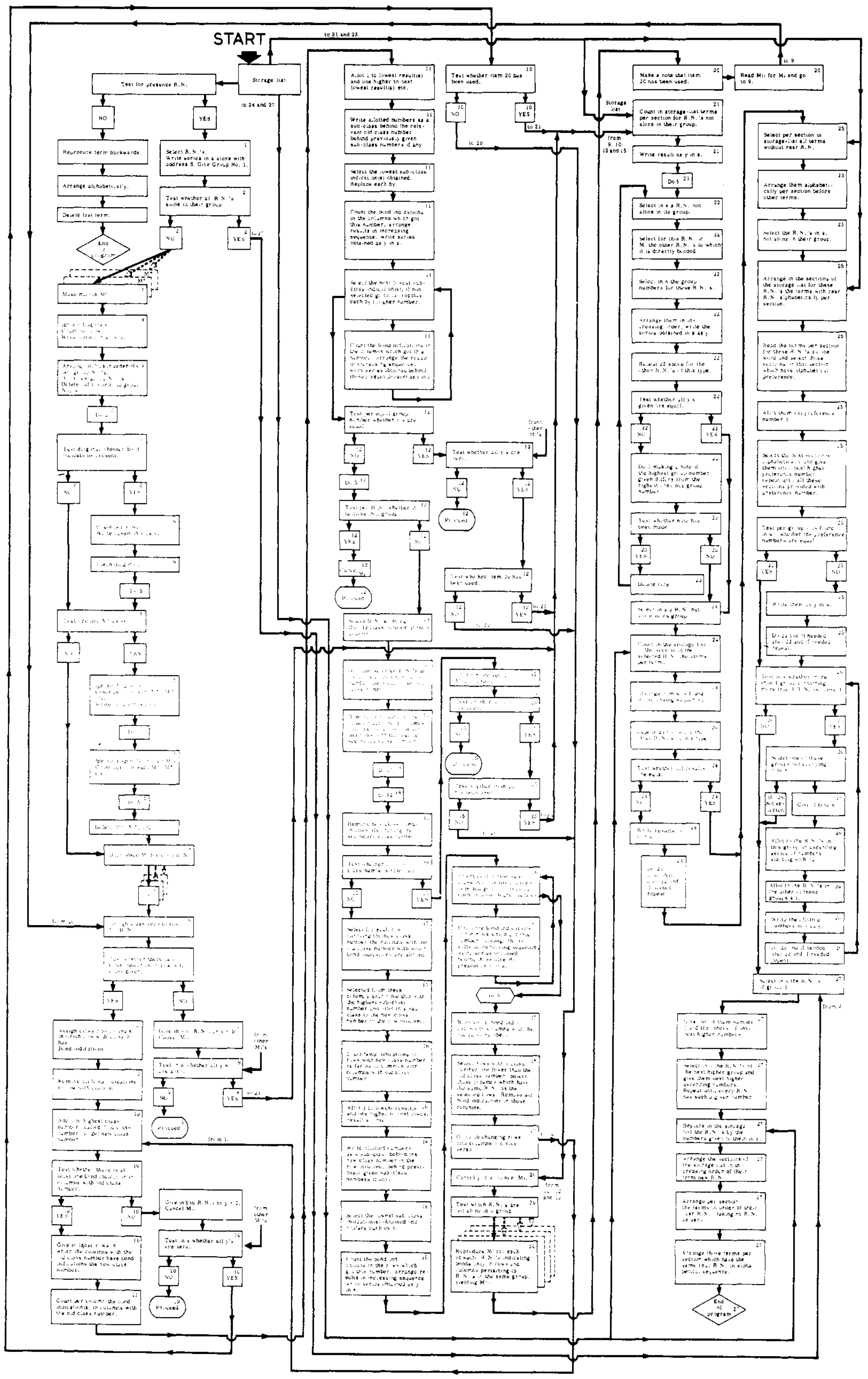


Figure 6.—Program outline for the LINCO system.

those on the right by (2). These position-numbers are placed directly adjacent to the carbon atom of the double bond or, if the latter is a branching point, between the reference number for that branching point and the group in question. If the group involved is a hydrogen atom bonded to carbon (which is not noted in LINCO), its position number is written only if no other groups determine its position already. Thus the notation for *cis*-butene-2 is /C(1)C = C(1)C, and that for *trans*-butene-2, /C(1)C = C(2)C.

In the machine program, terms with these special indications are stored in addition to the normal storage list (in which the special indications are not involved). After the standard reference numbers have been found, the numbers in the additional terms are also replaced by the standard numbers. During a search, compounds found *via* the storage list can be checked for the required isomeric form by means of the additional terms.

Isotopes: Special isotopes when present in chemical structures are indicated by the symbol ˆ ˆ followed by the atomic weight and the atomic indication. Thus ^{14}C is indicated as ˆ ˆ 14C and ^{60}Co is ˆ ˆ 60ˆ C*O*. In the machine the symbol ˆ ˆ means that the number which follows forms a whole with the atomic indication.

The simplicity of the system for the user is obtained at the cost of relatively complex, albeit straightforward, machine actions, since the machine has to substitute something for all those details (priority rules, etc.) which a human using other systems has to take care of.

It is of course desirable to investigate in cooperation with experts from different fields whether certain details could be made more convenient; the notation may be changed so as to facilitate writing or machine processing without deviating from the principles of the notation. Thereafter documentation by means of the LINCO system is intended to proceed along the following lines. After the chemist has selected from a document the formulas he considers worth including in the file, the rest of the work can usually be done by a nonspecialist clerical staff. For some formulas the chemist will have to provide additional notes (*e.g.*, for indicating an electric charge, etc.) before giving them to the clerk; once the chemist has become familiar with the system, he can see in which cases such indications are useful.

For searches directed toward a whole compound (*i.e.*, not a substructure), it is sufficient to search for identity between the notation for the compound in the file and the notation of the compound sought, after the latter has also been standardized (thus been fed to the computer preceded by the standardization program). No computing steps are necessary in such cases and very fast searching is possible.

If one prefers not to use the LINCO input system (which requires a conventional tape-making typewriter only, *e.g.*, a Flexowriter), but, say, the input method using the "chemical typewriter" (Walter Reed Comp.) or *via* a scanning machine as developed by E. Meyer (BASF), it should not be too difficult to transform the data stored *via* such inputs into LINCO-like data and to apply the program given here.

Substructure searches can be made in several ways:

(1) As described in the Appendix to the original article.[1] This is a very simple and straightforward method applicable in a large majority of cases.

(2) A special case of substructure searches is the search for rings. The rings present can easily be deduced from the matrix used (*e.g.*, as indicated by Dyson[2] by searching from point to point, but in this case for a much lower number of points), especially after classing the reference numbers in the network in the pyramid form described above (as the class numbers give certain clues to leave out a number of steps from the search). The parts of the rings found as sequences of reference numbers can very easily be translated into rings written in atomic notations from the storage list. It is not necessary—although possible—to ascertain the smallest rings present only (as is usual in the IUPAC–Dyson notation), as the others are found as well. If desired, every ring present in a compound can be filed, together with the storage list as a series of bonded atoms. If for complicated structures this would mean too many rings, it is easy to restrict those noted to medium size rings only. By adding the reference numbers encountered in the ring, the place of substituents is easily ascertainable.

(3) Comparison as proposed by Salton and Sussenguth.[3]

(4) The tree method mentioned by Dyson.[2]

Methods 3 and 4 will probably mean extending the matrix to form an atom-by-atom connection matrix, which is easily done. These methods can have advantages in cases where the structure searched for contains several atoms which may or may not be branching points in compounds related to the question in the storage.

An extended matrix of this type can also be used to translate the information stored *via* the LINCO system into the other codes (*e.g.*, the IUPAC-Dyson Code) as described by Dyson, etc.[2] The matrix from which Dyson starts for this purpose is easily generated from a LINCO-based storage. It may even be possible to base a system of giving unique names to compounds on the standard representation developed.

If desired, the standardized LINCO notation can also be used to arrange a large number of compounds in an index, for instance, on different tapes. The molecular formula, easily derivable from the storage list, can also be used for a rough arrangement.

Future Developments.—Many documents contain information which is insufficient to draw an exact chemical structural formula, but is nevertheless of sufficient value to be stored in a file. Data such as

"xylene" (which isomer?)
"alkyl aromatics" (how many alkyls? how many rings? are the alkyls also cycloalkyls? etc.)
"chlorinated styrene" (how many chlorine atoms? which position? still vinyl-present? etc.)
"polystyrene" (molecular weight?)

occur many times and should be covered in such a way that as much (and if possible not much more) information is covered as in the document itself. If the LINCO system or any other system were to be generally accepted, this author believes that it would be worthwhile to make the system chosen more sophisticated so as to cover this unclear information too, although of course the help of a chemist for coding would be essential, at least on the first occasion the expression is encountered.

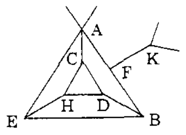
SUMMARY

An outline is given for a computer program for transforming the notation for a chemical compound, made by clerical staff directly from the structural formula and containing arbitrarily allotted numbers, into a unique, unambiguous notation, which can be translated into other codes. This notation makes it possible to search for whole compounds by simple comparison. Searches for substructures can be made in various ways. The advantages resulting from the use of the LINCO notation for this purpose as an input system are set out.

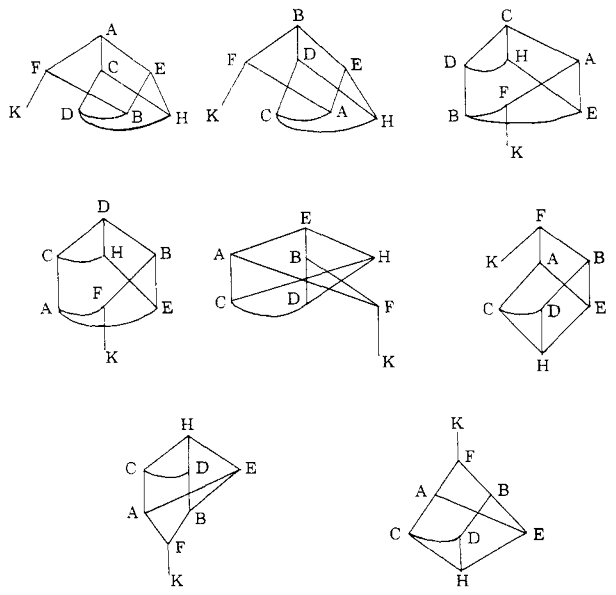
Acknowledgment.—The author wishes to express his gratitude to his colleagues in Shell and also to G. Salton for their inspiring discussions and suggestions.

APPENDIX
EXAMPLE AND PROGRAM OUTLINE

Example.—A network with the following structure and extending chains



the RN's of which are indicated by letters, can be represented by the following set of "pyramids" (ignoring the chains) with tops A, B, C, etc.



All of these pyramids have to be investigated.

If we take one of these, say C, as an example to see how the program below works, we get the results shown in Figure 7.

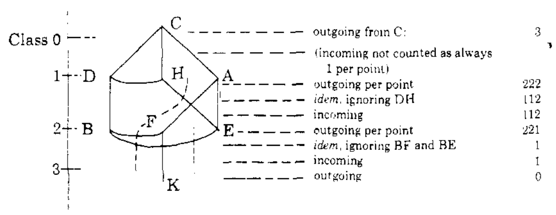


Figure 7.

The vertical broken lines indicate differences found between points, which are laid down in the subclasses of the program. By this means it is caused that the number 221 comes in this sequence and not 122, for example. The numbers given are found in this sequence in the program; the application of the program for C is continued as long as point C is in one group with at least one other point (in this case D). As soon as a difference is found between the series of numbers for one pyramid and those for all the other pyramids, the program stops for that particular point. The pyramid for K, for example, will give 1 as first number, while all the others give 3. Therefore K is not dealt with further.

It will be observed that in the C pyramid above not all the available information has been used; the points D and H could be interchanged, although they are not exactly equivalent. This could be corrected by a process (expressed in program item 16A), the application of which, however, does not seem to be necessary, since the redundancy involved in the use of all pyramids in all cases investigated cancels out this theoretical shortcoming.

If one wishes to use all available information one can proceed as follows: as soon as a difference is found between points in a certain class with respect to the incoming bonds (here class 2, where B and F have 1 incoming bond and E has 2), it is used to subdivide the previous classes accordingly. After the above series

```
  3
222
112
112
```

one goes back to class 1 and gives to the points for every bond going out therefrom to the next class (class 2) numbers based on the number of bonds going to the next class point to which the bond in question goes. Thus, DB goes to B (only one bond going to B) and HE goes to E (to which AE also goes), while the bonds from A go to F and E having 1 and 2 bonds, respectively. From these numbers 1, 2, 12, a difference between D and H can be obtained. This difference can be taken down to class 2 again (in the example this has no effect, but in other cases preference may be deduced therefrom).

A step-by-step exemplification is given below of all the actions performed by the machine to get the yardstick number for point C in the above example. It should be read side-by-side with the chapter "Program Outline" at the end of this Appendix.

Program no.

1 S will be:

RN:	A	B	C	D	E	F	H	K
Group:	1	1	1	1	1	1	1	1

2 Negative text result.

3 M′ will be:

Secondary class →											
	CLASS →										
↓	↓	RN	A	B	C	D	E	F	H	K
		A			1		1	1		
		B				1	1	1		
		C	1			1			1	
		D		1	1				1	
		E	1	1					1	
		F	1	1						1
		H			1	1	1			
		K						1		

M″, M‴, etc. will not exist.

4 S becomes:

RN:	A	B	C	D	E	F	H	K
Group:	1	1	1	1	1	1	1	1
Y:	3	3	3	3	3	3	3	1

5 New group numbers given. S becomes:

RN:	A	B	C	D	E	F	H	K
Group:	2	2	2	2	2	2	2	1

6,7 No change.

8 For each RN (apart from K, which is alone in its group) a matrix Mi is made.

9 M_C starts as given above for Mi, and after program point 9, it will have the following appearance (in which the changes to be made with respect to the result represented in the next drawing are indicated by a line at the bottom of the relevant squares).

M_C

Secondary class →											
	CLASS →		1			1			1		
↓	↓	RN	A	B	C	D	E	F	H	K
		A			1		1	1		
	$\underline{\quad}$	B				1	1	1		
	0	C								
	$\underline{\quad}$	D		1	1				1	
	$\underline{\quad}$	E	1	1					1	
	$\underline{\quad}$	F	1	1						1
	$\underline{\quad}$	H			1	1	1			
		K						1		

10 M_C

Secondary class →											
	CLASS →		$\underline{1}$			$\underline{1}$			$\underline{1}$		
↓	↓	RN	A	B	C	D	E	F	H	K
		A			1		1	1		
	2	B				1	1	1		
	0	C								
	2	D		1	1				1	
	2	E	1	1					1	
	2	F	1	1						1
	2	H			1	1	1			
		K						1		

11 M_C

Secondary class →											
	CLASS →		1.1			1.1			1.1		
↓	↓	RN	A	B	C	D	E	F	H	K
$\underline{\quad}$		A			1		1	1		
	2	B				1	1	1		
	0	C								
$\underline{\quad}$	2	D		1	1				$\underline{1}$	
	2	E	1	1					1	
	2	F	1	1						1
$\underline{\quad}$	2	H			1	$\underline{1}$	1			
		K						1		

Y for C: 222; the other Y's are found in the same way. Written in S, this results in:

RN:	A	B	C	D	E	F	H	K
Group:	2	2	2	2	2	2	2	1
Y:	222	222	222	222	222	022	222	

(for K no Y is applicable, as K is alone in group 1 and no matrix has been made for it).

12 After rearranging (5) and, giving new group numbers, S becomes:

RN:	A	B	C	D	E	H	F	K
Group:	2	2	2	2	2	2	2	1
Y:	222	222	222	222	222	222	022	
Group:	3	3	3	3	3	3	2	1

and after (5) has been dealt with, S becomes:

RN:	A	B	C	D	E	H	F	K
Group:	3	3	3	3	3	3	2	1

The matrix for F is cancelled (now alone in a group).

13,14

M_C

Secondary class →											
	CLASS →		1.1			1.1			1.1	
↓	↓	RN	A	B	C	D	E	F	H	K
1		A			1		1	1		
	2	B				1	1	1		
	0	C								
1	2	D		1	1					
	2	E	1	1					1	
	2	F	1	1						1
1	2	H			1		1			
		K						1		

Via (11) one finds Y for C: 112.
In S one gets, after rearrangement:

RN:	A	B	E	C	D	H	F	K
Group:	3	3	3	3	3	3	2	1
Y:	222	222	222	112	112	112		
Group:	4	4	4	3	3	3	2	1

in which the second and third lines are deleted.

15

M_C

Secondary class →											
	CLASS →		1.1			1.1			1.1	
↓	↓	RN	A	B	C	D	E	F	H	K
1		A			1		1	1		
	2.1	B				1	1	1		
	0	C								
1		D		1	1					
	2.1	E	1	1					1	
	2.1	F	1	1						1
1		H			1		1			
		K						1		

16 The count yields 1, 2, and 1 (*i.e.*, 2 subclasses 1 and 2) and this changes the new class numbers first into 2.11, 2.12, and 2.11, respectively; and, after replacement by a single decimal, into 2.1, 2.2, and 2.1, respectively. This gives in Y, as a rearranged count, the result 1, 1, 2. This is done for the other RN and, written in S, the result is:

RN:	A	B	E	C	D	H	F	K
Group:	4	4	4	3	3	3	2	1
Y:	1122	1122	222	112	112	22		
Group:	6	6	5	4	4	3	2	1

in which the second and third line are deleted. The matrices for H and E are deleted, as they are now alone in a group.

16A The columns for the RN A, D, and H get an additional subclass, so that their indications become 1.12, 1.11, and 1.12, respectively, which, after replacement by a single decimal, become 1.2, 1.1, and 1.2. The count, beginning with column D and followed by A and H, yields: 1, 2, 1, which, after arrangement of the last two figures (as they are in the same subclass), becomes 112.

For MD the result is also 112, and A and B also get the same number. Doing (12) does not result in doing (5); thus, one goes to (17), after changing the subclasses for the new class number. This latter step means that the indications for the rows B, E, and D—which became 2.1, 2.2, and 2.1, respectively, in (16)—now become 2.11, 2.22, and 2.12, and after replacement by a single decimal, 2.1, 2.3, and 2.2, as shown.

M_C

Secondary class →											
	CLASS →		1.2	—		1.1	—	—	1.2	—
↓	↓	RN	A	B	C	D	E	F	H	K
1		A			1		1	1		
	2.1	B				1	1	1		
	0	C								
1		D		1	1					
	2.3	E	1	1					1	
	2.2	F	1	1						1
1		H			1		1			
		K						1		

17 The bonds in column A, D, and H are removed.
18 The bonds in column C are removed.
19 New class numbers given *via* (10).

M_C

Secondary class →				—			—	—		
	CLASS →		1.2	3		1.1	3	3	1.2	3
↓	↓	RN	A	B	C	D	E	F	H	K
1		A					1	1		
	2.1	B					1	1		
	0	C								
1		D		1						
	2.3	E		1						
	2.2	F		1						1
1		H					1			
		K						1		

19:11 The count in rows B, E, and F yields 2, 1, and 2, resulting in two subclass indications, *viz.*, 1 and 2 for these rows, yielding 2.12, 2.31, and 2.22, which, after replacement by a single decimal, become 2.1, 2.3, and 2.2 (unchanged); the result of the count thus becomes, after arrangement,

221. Together with the results for the other counts written in S, this yields:

RN:	A	B	E	C	D	H	F	K
Group:	6	6	5	4	4	3	2	1
Y:	0211	0211		221	221			

19:12 As A and B, and also C and D, are together in a group and as, inside these groups, no difference in Y is found, Y not being zero, one proceeds.

19:13 Secondary classes are given.

19:14 Bonds at intersections are removed.

19:15 All class numbers 3 are removed for A and B (thus their Y is fixed at zero and M_A and M_B are cancelled), while for M_C and M_D only column K now has class 3. This is allotted the subclass from row F, yielding 3.2, which, after replacement, becomes 3.1.

M_C

Secondary class →				2			2	2		
	CLASS →		1.2			1.1			1.2	3.1
↓	↓	RN	A	B	C	D	E	F	H	K
1		A					$\underline{1}$	$\underline{1}$		
	2.1	B								
	0	C								
1		D		$\underline{1}$						
	2.3	E								
	2.2	F								$\underline{1}$
1		H					$\underline{1}$			
		K						1		

19:16 The count yields 1 for both M_C and M_D; thus, one proceeds *via* (12).

19:16A Skipped, as there is only one column with the new class number.

19:17 Bonds in rows having class 2 are removed.

19:18 Bonds in rows carrying secondary class number 1 are removed.

19:19 In (10) no new class number can be given for M_C and M_D, and their Y is fixed at zero. As all Y's are now zero, go to 20.

20 Four matrices are made. M_{AA} and M_{BB} do not contain bond indications and their Y is fixed at zero in (9). M_{CC} and M_{DD} are shown here. Their Y is fixed at zero in (10) and, as all Y's are now zero, one goes to 21.

M_C; M_D

Secondary class →										
	CLASS →									
↓	↓	RN	A	B	C	D	E	F	H	K
		A								
		B								
		C				1				
		D			1					
		E								
		F								
		H								
		K								

21 The count results in:

A	B	C	D
5	3	3	3

Treating as Y, yields in S:

RN:	A	B	E	C	D	H	F	K
Group:	6	6	5	4	4	3	2	1
Y:	5	3		3	3			
Group:	7	6	5	4	4	3	2	1

which after deletion of old data becomes:

RN:	A	B	E	C	D	H	F	K
Group:	7	6	5	4	4	3	2	1

22 From M′ it follows that C is bonded to E, D, and H, while D is bonded to B, C, and H, which, converted into group numbers, yields: 5, 4, 3 and 6, 4, 3 and, after arrangement, 3, 4, 5 and 3, 4, 6. Thus, S becomes (no rearrangement needed):

RN:	A	B	E	C	D	H	F	K
Group:	7	6	5	4	4	3	2	1
Y:				345	346			
Group:	8	7	6	5	4	3	2	1

As all RN's are now alone in their group, one goes *via* 2 to 27. In S one finds:

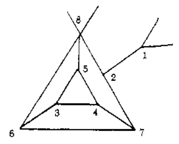
RN:	A	B	E	C	D	H	F	K
Group:	8	7	6	5	4	3	2	1

and this is applied to the storage list to replace the RN by the numbers indicated. The list will be

1-	43	72
1-	45	74
$\underline{12}$	$\underline{47}$	$\underline{76}$
21	53	8-
27	54	8-
$\underline{28}$	$\underline{58}$	82
34	63	85
35	67	$\underline{86}$
$\underline{36}$	$\underline{68}$	

As all the terms have a unique position, apart from 1-, 1- and 8-, 8-, the alphabetical arrangement is to be applied to these terms only.

The new RN would appear in the network as



Program Outline.—Remark: If no Reference Number (RN) is present in the storage list the single term present is reproduced backward; the terms now present are arranged in alphabetical order and the last one is deleted. If one or more RN's are present the program below applies.

(1) Write a series S of the RN's; consider this series as a group with group number 1.

(2) Test whether all RN's are alone in their group; if so, go to (27).

(3) Make from the storage list a binary square matrix M′, consisting of columns and rows, each one of which is assigned to a certain RN in the same sequence; in this matrix, connections between RN's (as found in those terms in the storage list which also have a RN at the back) are indicated by the figure 1. If a 1 is to be indicated on a place which is already occupied by a 1, make a second (third, etc.) matrix M″ (M‴, etc.) and

fill in the information not insertable in M′ in M″ (M‴, etc.).

(4) Count in M′ for each RN the frequency of bond indications per row, ignoring 1's in the diagonal containing the intersections of rows and columns with the same RN, these frequencies being noted as yardstick numbers Y in S.

(5) Arrange the RN's in S per group in decreasing order of Y and allot group numbers, starting with a 1 for the last RN in S, and increasing each time with 1 as either the previously used group number increases or as Y increases inside a series with the same previous group number. Delete previous group numbers and previous Y's from S. Do (2).

(6) Test whether the diagonal in M′ contains bond indications. If not, go to (7). If so, repeat (4) also counting the diagonal in order to get new Y's, delete in M′ the bond indications in the diagonal, and do (5).

(7) Test whether M″ exist. If not, go to (8). If M″ exist, count per RN the sum of the bond indications in M″, M‴, etc., ignoring the diagonal indications and use the result as Y; do (5). Repeat this, also counting those diagonal indications which are in M‴, M⁗, etc. Delete M″, M‴, etc.

(8) Reproduce M′ for each RN not alone in its group to get Mi. The following items must be done for each RN, not alone in its group, in its own Mi.

(9) Assign to the row for the RN i in Mi class number 0, and assign to the columns in Mi in which this row 0 has bond indications class number 1; if no class number 1 can be given the Y for this RN is fixed at zero and its Mi is cancelled; if all Y's are zero now go to (21); remove all bond indications in the row with class number 0.

(10) Assign to those rows in Mi in which bond indications are found in columns with the highest class number (hereinafter called the old class number) a class number which is 1 higher than the old one (hereinafter called the new class number). If no new class number can be given, the Y for this RN is fixed at zero as far as (20), and its Mi is cancelled. If all Y's are zero go to (20) if not yet met; otherwise, (21).

(11) Count the bond indications in the columns with the old class number and allot to the old class number including its subclass per column a decimal type (sub) subclass number in increasing order of the result, starting with 1, those which gave an equal result getting equal subclass numbers; replace the subclass indications obtained in increasing order by one decimal type number starting with 1 for the lowest subclass and ascending with 1 each time the subclass as a whole increases; arrange the result of the counts in increasing order of the subclass indication now obtained for the columns involved and write it in this sequence as Y in S.

(12) Test whether all Y's given are equal if their RN is in the same group. If not, do (5) and cancel Mi's for those RN's which are now alone in a group. If equal, text whether all Y's are zero. If zero, go to (20) if not yet met; otherwise, (21).

(13) Assign to the rows for the RN's, the columns of which have the old class number, a class number identical with the old class number (which assigned class number is hereinafter called the secondary class number).

(14) Remove all bond indications at intersections of columns with the old class number and rows with the secondary class number. Do (11) and (12).

(15) Remove assigned new class numbers where they are accompanied by the secondary class number. If all removed, the Y for this RN is fixed at zero until (20) and its Mi is cancelled. If all Y's are now zero go to (20) if not yet met; otherwise, (21). Allot to the novel class number per row the highest subclass number assigned to the columns with the old class number, with which it has bond indications in common.

(16) Count in the rows having the new class number the bond indications in columns with the old class number and allot to the rows a decimal type (sub) subclass number in increasing order of the result, starting with 1, those which gave an equal result getting equal subclass numbers; replace the subclass indication obtained in increasing order by one decimal type number starting with 1 for the lowest subclass, ascending with 1 each time the subclass as a whole increases. Arrange the result of the count mentioned above in increasing order of the subclass indication now obtained for the rows involved and write it in this sequence as Y in S. Do (12).

(16A)† Test whether there are more than two columns with the old class number and more than two rows with the new class number. If not, go to (17). If so, test whether in the rows for the new class number the number of bonds in common with the columns for the old class number is the same. If so, go to (17). If not, allot to the old class number per column the highest subsclass number assigned to the rows with new class numbers with which it has bond indications in common; if there is none in common, zero is assigned; replace the subclass indications obtained in increasing order by one decimal type number, starting with 1 for the lowest subclass, and ascending by 1 each time the subclass as a whole increases. Count in the columns with the old class number in increasing order of their subclass indication, the bonds they have in common with the rows having the new class number; arrange them in increasing order within the same subclass and treat the result as Y. Do (12), noting whether (5) has been applied. Allot to the new class number per row the highest subclass number assigned to the columns with the old class number with which it has bond indications in common, and replace the subclass indications obtained in increasing order by one decimal type number, starting with 1 for the lowest subclass, and ascending by 1 each time the subclass as a whole increases. If (5) has not been applied according to the note, go to (17). If (5) has been applied according to the note, test whether the old class number is higher than 1. If not, go to (17). If so, make a note of it, write a duplicate of M′ fo: all RN's not alone in a group, transfer the subclass indications found above to the same rows and columns in the duplicate M′, cancel the old Mi's, and go to (9), following the program and using Y only when 16A is reached in the step in which the noted old class number occurs as a new class number.

(17) Remove all bond indications in the columns with the old class number.

(18) Remove all bond indications in the columns carrying the same RN's as the rows with a class number one lower than the old class number.

(19) Go to (10), changing "rows" into "columns" and *vice versa*.

(20) Cancel Y in S, cancel Mi and reproduce M′ for a certain RN not alone in its group, leaving out all bond indications in rows and columns assigned to RN's in other groups. Repeat this for all such RN's to get Mii. Go to (9), reading Mii for Mi.

(21) Count in the storage list the number of terms present in sections pertaining to RN's not alone in their group and handle the result as Y. Do (5).

† This point is really redundant and can be deleted if desired.

(22) Select in M′ for a RN not alone in its group the other RN's to which it is directly bonded; select in S the group numbers for these RN's, arrange them in increasing order, and handle the result as Y. Repeat this for the other RN's of this type. If all Y's are equal go to (24); if not (23).

(23) Do (5), making a note if the highest group number given differs from the highest previous group number. If a note has been made, delete the note and repeat (22). If no note has been made go to (24).

(24) Count in the storage for the sections pertaining to RN's not alone in their group the number of atoms (including double or triple bonds) per term, arrange the numbers obtained per section in increasing order, and check whether the numbers obtained per section are equal. If equal, go to (25). If not, handle the result as Y and go to (23) reading for (24), (25).

(25) Arrange per section the terms without rear side RN in alphabetical order in front of the other terms. Arrange per section pertaining to those RN's which are not alone in their group the terms with rear RN and arrange these terms themselves in alphabetical order. Select from the sections pertaining to RN's which are not alone in their group the section which would come first alphabetically if the terms in that section were considered as one word. Allot to this section a preference number 1; repeat the selection for the remaining sections of this type, giving them also a preference number, which is equal to the previous one if the selected section is alphabetically identical with the previous one and one higher than the previous number if it is not identical. Test whether per group all preference numbers are equal. If equal, go to (26). If not equal, use them as Y and go to (23) reading for (24), (26).

(26) Test whether more than one group containing more than one RN is present. If not, go to (27). If present, allot to the RN's in one of these groups an ascending series of numbers starting with 1. Allot 1 to the other RN in these groups and handle the result as Y. Go to (23) reading for (24), "(26) but allotting the ascending number series to another group of that type."

(27) Allot to the RN's a number increasing by 1 and, starting with group 1, going to higher groups in S. Replace the RN's in the storage list by the numbers allotted to them. Arrange the sections of the storage list in increasing order of their front new RN. Arrange per section the terms in increasing order of their rear RN, taking no RN as zero, and, as far as two or more rear RN's in a section are equal, in alphabetical order of the terms in question.

REFERENCES

(1) H. Bouman, *J. Chem. Doc.*, 3, 92 (1963).
(2) G. M. Dyson, W. E. Cossum, M. F. Lynch, and H. L. Morgan, *Inform. Storage Retrieval*, 1, 69 (1963).
(3) G. Salton, and E. H. Sussenguth, private communication from The Computer Laboratory of Harvard University, Cambridge, Mass.

ChemSEARCh—An Operating Computer System for Retrieving Chemicals Selected for Equal, Analogous, or Related Character*

DAVID GOULD and EDWARD B. GASSER,
Colgate-Palmolive Research Center, New Brunswick, New Jersey

and

JOHN F. RIAN
Control Data Corporation, Rockville, Maryland
Received March 9, 1964

All chemists, and especially those whose responsibilities include synthesis of compounds having pharmacologic or medicinal effect, have felt the necessity for a reference file of structures and publications organized for their areas of interest. From this file, they wish to select: (1) specific compounds, (2) compounds related by containing specific subgroups of atoms, (3) compounds having a specified relation between groups, and (4) activities of compounds. From these components, they hope to derive structure-activity relationships to guide future research.

The organization of such a file has progressed through the alphabet, nomenclature as in *Chemical Abstracts*, and sorting by groups as in "Beilstein," The last still has considerable value to the chemist, particularly by use of edge-notched cards[1] with such systems as the first published by Frear.[2] When these approaches showed some difficulties owing mainly to ambiguity, Dyson[3] initiated complete, unique, and unambiguous codes. A serious problem with these is the complexity of the coding

* Presented at the Symposium on Advances in Chemical Structure Storage and Searching Systems. 147th National Meeting of the American Chemical Society, Philadelphia, Pa., April 6, 1964.