A Linearization of Chemical Graphs*

By H. HIZ
Department of Linguistics,
University of Pennsylvania, Philadelphia, Pennsylvania
Received November 25, 1963

I. INTRODUCTION

A chemical graph, a widely used though imperfect representation of a molecular structure, is a finite, two-dimensional, connected, undirected, labeled multigraph.[1] For many purposes it is convenient to represent it by a linear formula. Several well-known systems of chemical ciphers give rules for writing such formulas.[2] Usually, however, these systems attempt not only to give a linear formula for a chemical graph but also to give a formula which is economical both in the alphabet used and in the length of the formula. Most authors tend to include in the cipher chemical information beyond that which is explicitly contained in the graph of the molecular structure. It may be of value to consider a linearization of chemical graphs which is neutral as to the "chemical properties" of the graph and which at a sacrifice in length of a formula gives a closer representation of the graph itself. When one reads a graph he follows a path. A path through a graph is itself a linearization of the graph. The system of rules which will be explained below gives for each graph and each path through it a formula which is a record of the path; the formula is the graph represented by a path through it. There are many possible paths through a graph and the well-known systems of ciphers establish in a more or less explicit way the preferred, or distinguished paths through a graph. This preference of a path over all others gives the system a property of uniqueness in the sense that each graph which results from decoding a formula, when again coded, yields only the original formula.[3] The uniqueness, desirable as it may be for some purposes, usually is based on a more or less arbitrary choice of the preferred path. But different choices may be best suited in different applications of the same system. If a property of a graph can be recognized most efficiently in one of its ciphers, that cipher should be used when this property is in the focus of study. We may want to group together graphs which have O doubly connected with N, or graphs which have a given type of cycle. It will be useful to be able to rewrite ciphers so as to emphasize this particular property uniformly in ciphers representing all graphs with that property. Their commitment to uniqueness limits the freedom of the systems to choose in each application ciphers which suitably exhibit these properties of the graphs which are most relevant to the application. The rules explained below have a large degree of flexibility in this respect. They give as many ciphers for a graph as there are paths through the graph.[4] The great laxity in uniqueness makes it possible to introduce limitations according to various purposes by means of additional rules in each case specially appended to the system. Such laxity is not absent in scientific investigations. One does not require uniqueness in representation of numbers.

Suppose we have an operation defined for chemical graphs which out of two graphs make a third one. We may wish to to find a corresponding operation for ciphers which preserves the assignment of ciphers to graphs. This task may be easier when we have more choice.

Finally, there are possibilities of applying the proposed method of ciphering to non-organic compounds. We will make a few hints on this topic at the end of this paper. A cipher of a graph is a formula. A set of rules for coding graphs into ciphers is correct if and only if each graph which results from decoding of a formula, when coded again, yields the original formula, and possibly some other.[3] The rules proposed in this paper satisfy the above definition of correctness. Here, however, one needs a more precise notion of *one graph*. Unfortunately graphs do not satisy all the requirements for representation of molecular structures. Chemists are familiar with situations where two graphs considered equivalent (topologically) are taken to represent two different structures. A typical example is provided by *cis* and *trans* isomers. In the theory of graphs two graphs are equivalent if they differ only by the angles between the edges. The inadequacies in graphing of molecular structures necessitate caution before a claim is made for a system that its rules are complete in the sense that every graph ever considered by the science will be representable by a cipher. The present paper considers only fairly ordinary chemical graphs. Each node is normally labeled only by a name of an element—though in some cases not labeled at all. Provisions can be made to handle in the ciphers this and other features.[5]

The notational devices used here are a development of the so-called Polish notation for logic and mathematics. In this notation the functor always precedes its argu-

* This work was supported by an NSF grant for analysis of chemical notations.

(1) For graph theoretical terminology, see C. Berge, "Théories des graphes et ses applications," Paris, 1958.

(2) *E.g.*, "Rules for I.U.P.A.C. Notation for Organic Compounds," 1961; also "Wiswesser Notation" (under revision), E.G. Smith, mimeographed.

(3) This definition of uniqueness, as well as several other definitions and formulations, are from J. Munz, "Report on the Feasibility Study," University of Pennsylvania, mimeographed. Munz's comments on the preliminary version of the present paper resulted in several improvements.

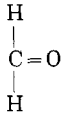
(4) Some limitations on the flexibility remain, due to the limitations on what can be formally considered a path through a graph. An intuitive concept of a path will be replaced here by a formal one. Perhaps one wants to consider as a path an order of nodes of a graph which will not be a path in the formal sense.

(5) Since we do not use the order of arguments as significant, one may use it for the purpose of distinguishing isomers, or distinguishing some spherical or other properties not accounted for by a normal two-dimensional graph.

ments.[6] If the number and character of the arguments are known, and depend on the functor solely, then there is no need for parentheses. However, the ciphering system will be more involved than Polish logical notation because the graphs we are dealing with are not exclusively trees.

II. CIPHERS OF TREES

The principal idea of linearization is based on the fact that each atom of a given element has constant valency, *i.e.*, has the same number of connections with other atoms. Thus valency of H = 1, valency of O = 2, valency of N = 3, valency of C = 4, valency of Si = 4, etc. Therefore, if we write a symbol for an atom we expect that it will be associated within the formula with the prescribed number of atomic symbols. Each labeled node of a chemical graph will be connected with a prescribed (by valency) number of nodes of the graph. The number of edges from a node is constant for all nodes of the same label (double connections counting twice). If we arbitrarily choose a node in the graph we may consider it as a functor which requires a definite number of arguments, the arguments being the nodes with which the chosen node is connected. In turn, the nodes which occurred as arguments themselves call for specific numbers of arguments. It may happen that the sum of valencies of all nodes occurring as arguments of a functor (*i.e.*, connected with a node which we have chosen as our starting point) is exactly the same as the number of expected connections from the functor. In such a case listing of the functor and after it of the arguments ends our job. The graph cannot be any larger. For instance, if in

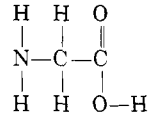


Graph 1

we chose C as the functor, then two H's and one O twice are its arguments. We list those arguments after C. We write two letters H and one letter O preceded by the numeral 2 to indicate O's double bond with C. We obtain a linear representation of the graph

C H H 2O (1)

The C calls for four connections. They are listed by showing the nodes to which these connections go from C. The order in which the arguments of any functor are listed is irrelevant. The only exception is the double use of the same node in a double bond, when the two arguments are listed together. The cipher (1) indicates that we scanned the graph starting by C followed by one H, another H (we do not know in which order—they are chemically and topologically undistinguishable) followed by O to which we have a double connection. Because H has valency 1, it does not need any argument. Its functor satisfies this valency. The same holds for O of which the valency is 2. The double connection with C satisfies this requirement. The situation is, however, different in the case of, *e.g.*, aminoacetic acid. The graph of it is



Graph 2

Let us start scanning this graph from the central C. It has as its arguments H, H, N, and another C. Had we listed these arguments after C we would have obtained

C H H N C

H having valency 1 behaves as before. But N has valency 3. It must be connected to three nodes. One of them is the central C, our functor, but the others are two H's. To indicate this we should list H H right after N. It amounts to now treating N as a functor from two arguments, one argument less than its valency. The formula C H H N C would indicate wrongly that the last C is an argument to N. But the last C is not connected in our graph with N. The correct way to proceed with our cipher is

C *H H* N H H

Now N is "saturated"; all of the arguments required by its valency appear in the cipher. It is singly bonded to its functor (the starting C) and has two H arguments with each of which it is singly bonded. Therefore the N in this cipher has already its three required connections. Each of the H's is also "saturated" by their functor N. Therefore the next symbol will be an argument to the last "unsaturated" symbol, *i.e.*, the opening C. Now we may list the C which occurs in the graph to the right.

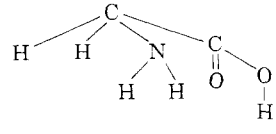
C H H N H H C

The last C again calls for arguments. It calls for three arguments, while the first C called for four arguments.

In any cipher each symbol (node) singly connected with its functor calls for one less argument than its valency. And each symbol doubly connected with its functor calls for two less arguments than its valency. H, unless it occurs at the beginning of a cipher, calls for no argument. The last C of our example is singly bonded to the first C. It, therefore, requires three arguments. It is doubly connected with an O and singly with another O. The first O is "saturated" by being a double argument to C (valency of O being 2). The second O requires one argument which is provided by the right-most H. The total cipher which we obtain in this way is

C H H N H H C 2O O H (2)

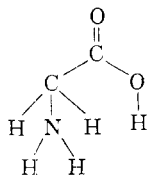
The cipher (2) can be viewed as a tree representation of graph 2. The tree



Graph 3

is equivalent to graph 2 and the cipher (2) is a typical Polish notation for graph 3, except for the double bond. If a graph does not have cycles it can be represented as a tree. Of course the tree of graph 3 is not the only tree equivalent to graph 2. And, correspondingly, the cipher (2) is not the only cipher of this graph. Another tree equivalent to graph 2 is

(6) A functor is a phrase which together with a specified number of other phrases, called arguments of the functor, forms a composite phrase. Thus, in $x = y$ (which in Polish notation is rendered as = *xy*) the equality sign is the functor and the two variables are the arguments.



To graph 4 corresponds the cipher

O 2C C HNHHHO H (3)

(which is also a cipher of graph 2 and of graph 3). Here are other ciphers of graph 2.

C C O H 2O N H H H H (4)

C C O H 2O H N H H H (5)

H O C 2O C H H N H H (6)

H C H C O H 2O N H H (7)

H N C C O H 2O H H H (8)

etc., (2)–(8) are all synonymous. Similarly, cipher (1) is synonymous with any of the following

H C 2O H (9)

H C H 2O (10)

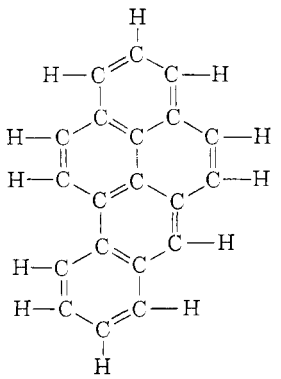
O 2C H H (11)

Note that a cipher contains exactly as many letters of a kind as there are atoms of a given element in the molecule. Thus counting the number of occurrences of each letter in a cipher we can obtain the molecular formula. From (2), and from all ciphers synonymous with (2) we have: $C_2H_5NO_2$.

III. CIPHERS OF CYCLIC GRAPHS

A graph is cyclic if it contains a sequence of nodes $x_1, x_2, \ldots, x_n$, each node connected by an edge to its successor where the node x_n is the same as x_1. In chemical graphs, $n > 2$. To cipher such graphs, a device is needed for referring to the same node more than once within a cipher. In mathematical notation such a cross reference is accomplished by the use of variables. A variable occurs at a given position in a formula and conforms to (looks the same as) a variable occurring in a different position in the formula. Thus, in the formula $x \cdot (y + z) = x \cdot y + x \cdot z$ it is the conformity between the three occurrences of x and the conformity between the two occurrences of y (and similarly for z) that assures us of the meaning. The first multiplied by the sum of the second and the third is equal to the sum of the products of the first and both the second and the third. After the substitution of constant values for variables, the variables disappear. And the intended cross reference disappears; *e.g.*, from the arithmetic formula we may get $3 \cdot (3 + 2) = 3 \cdot 3 + 3 \cdot 2$ which may also be a substitution of $x \cdot (x + y) = x \cdot x + x \cdot 2$. The chemical graphs are labeled and labels are like assigned substituted values for variables. For these reasons in chemical ciphers the labels themselves cannot convey the cross reference and we need something like a variable in order to code the cyclic graphs.

It appears that the only thing needed is a method to refer to an already coded node. This reference can be accomplished by naming this node, *e.g.*, by assigning to it the numeral corresponding to the place at which it already occurred in our cipher. We shall write it as X*n* where *n* is the position in which the node appears (for the first time) in our cipher. As an illustration let us consider a graph of benzpyrene (graph 5).

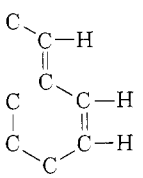


Graph 5

It contains 32 nodes. A linear representation of it, a cipher, will contain 32 letters not counting numerals or X.[7] A linear cipher will record a path through the graph when it is scanned by a man or a machine. Following the procedure explained in section II we start the scanning with any node we please, *e.g.*, with the uppermost C. The beginning of our cipher will read

C C H 2C C H 2C H C C C (12)

It covers the following part of the graph



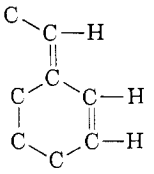
Graph 6

The path of scanning is here chosen at will. Presumably a chemist only rarely would leave the uppermost H to be accounted for later. Now we can go either to the left by 2C or close the circle by moving to the right. To do the latter, we must refer to the C which is singly connected with the last C of (12) and which is different from the next to last C of (12). But this C was already stated in the cipher, namely as the fourth letter of the cipher (not counting numerals). We cannot state it again as C following the last letter of (12). For then it would play the role of a functor. We would expect that after it there would appear the arguments that would "saturate" it. But these arguments already appear in (12) and the fourth node must become "saturated" as soon as it is listed as an argument of the last C of (12). Thus we want it to occur here as an argument only. Though this node was already stated the connection was not. We will use the letter X followed by a numeral corresponding to the place (in serial order) of this node as it appears for the first time in our cipher. Thus here we will use X4. We obtain the following part of the cipher:

C C H 2C C H 2C H C C C X4 (13)

(7) One should note that some chemical elements are usually symbolized by two letters, *e.g.*, Cl, Xe, Si. We will speak here as if each of these names is a single letter. For mechanical computation purposes we can assume the notation to be normalized.

It covers the following part of graph 5



Graph 7

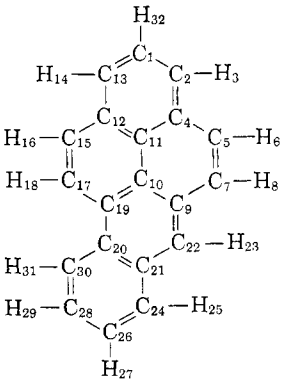
The upper left C of the cycle was the last functor stated. We also stated one of its arguments. Now we have to state other arguments of this C. There is only one, namely the doubly bonded C. We must write it next. This part of the cipher now takes the form[8]

C C H 2C CH 2C H C C C X4 2C

Proceeding in a similar manner we may obtain the cipher

C C H 2C C H 2C H C C C X4 2C C H 2X1
C H 2C H C 2X10 C 2C C H 2X9
C H 2C H C H 2C H X20 H (14)

It lists all the 32 nodes in the order indicated by numerals attached to the atoms in graph 8.



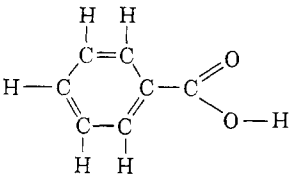
Graph 8

Let us explain why we need the numeral after X. The 22nd node is doubly connected to the 9th node, and it is not connected to the 20th node. Had we written only C H 2X, understanding, *e.g.*, that X is the last node which needed "saturation" by an argument, we would obtain that the 22nd node is connected to the 20th—contrary to the graph. For in the part of the cipher (14) up to and including the 22nd node the last node which could be joined to a new argument of the 22nd node is the 20th node.[9]

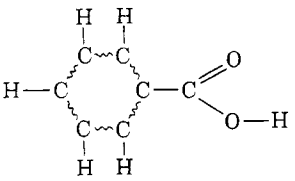
IV. DECODING

There is a simple algorithm to decode a graph from a cipher. As is well known a graph can be represented by a matrix; a graph of n nodes by an $n \times n$ matrix. Given a chosen order of nodes in the ith row and kth column position of the matrix there is 1 if there is a connection between the ith and kth nodes of the graph. Otherwise 0 appears there. If there are many kinds of connections, other numbers occur in that place. In the case of chemical graphs the values occurring in matrix usually are 0, 1, or 2 (in exceptional cases other numbers). The number indicates the number of bonds. To represent this matrix it is sufficient to consider n columns of numbers where in the ith column the number j occurs only if (j,i) in the matrix $\neq 0$. If $(j,i) = 2$, then j occurs twice in the ith column.

To illustrate, consider a shorter graph than the one discussed in the preceding section. The graph for benzoic acid (9)[10] has 15 nodes.



Graph 9



Graph 9′

A cipher for it is

C C 2O O H 2C H C H 2C H C H 2C H X1 (15)

If we number the letters (omitting numerals and X), then we obtain.

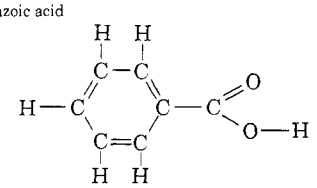
C	C	2O	O	H	2C	H	C	H	2C	H	C	H	2C	H	X1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	(16)

The matrix for graph 9 scanned as in (15) is shown in Fig. 1 or in a short form

C	C	2O	O	H	2C	H	C	H	2C	H	C	H	2C	H	X1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
2	1	2	2	4	1	6	6	8	8	10	10	12	1	14	(17)
6	3	2	5		1		9		8		13		12		
6	3				7		10		11		14		12		
14	4				8		10		12		14		15		

(8) The appearance of 4 just in front of 2 may lead to confusions. This is a problem faced by all processing of formulas involving numerals. Appropriate safeguards must be taken.

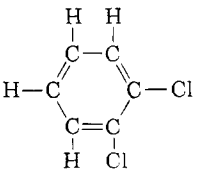
(9) Some elements fail to have a consistent valency (*e.g.*, N). This may be dealt with by using a different atomic symbol for each valency. This is how the Wiswesser system deals with the problem. Resonance presents another complication. If resonance is indicated in the graph by a special type of edge, *e.g.*, a wavy line (where two wavy lines are equivalent to three single-valency bonds), we can translate it into a cipher by preceding every argument so bonded to its functor with a special mark, *e.g.*, Z. Thus the benzoic acid

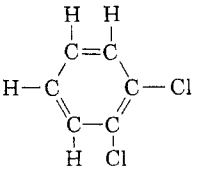


can be reduced into cipher

C H ZC H ZC H ZC C 2O O H ZC H ZC H ZX1

(10) Note that graph 9 is equivalent with graph 9′. Graphs 9 and 9′ have the same ciphers. However, the two graphs





have different ciphers (and are not topologically equivalent), though they are usually considered by chemists to be equivalent. In the Wiswesser system the letter R is used for any benzene ring. From the point of view of the present paper the letter R used in the Wiswesser system is a metaciphering device.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	0	0	0	2	0	0	0	0	0	0	0	1	0
2	1	0	2	1	0	0	0	0	0	0	0	0	0	0	0
3	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	2	0	0	1	1	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	1	2	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
14	1	0	0	0	0	0	0	0	0	0	0	2	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fig. 1.—Matrix for graph 9 as scanned in (15).

To decode a cipher is to reconstruct from it the short form of its matrix. We will state the rules of the algorithm for such a reconstruction and then we will illustrate their application by obtaining (17) from (15).

Rule 1: Assign consecutive numerals to consecutive letters of cipher, not assigning anything to the numerals of the cipher or to X.

In further rules we refer to the columns by the assigned numerals and to the letters of the cipher as the head letters of the columns. We will count the assigned numeral as one of the numerals entered in the column.

The following two definitions are needed for Rule 2.

We consider a column saturated if it has one more numeral than the valency of its head letter (letter of the cipher).

We consider a column i ($i > 1$) to have its scope completed if it has as many numerals as the valency of its head letter in case the head letter is not preceded by 2 in the cipher, and in case it is preceded by 2, if it has one less numeral than the valency of its head letter. For column 1 its scope is completed only with saturation.

Rule 2: Proceeding through the cipher from right to left, in each column j of incomplete scope put i, and in column i put j, if i is the nearest to the right ($i > j$) unsaturated column; in case i is preceded by 2 do it twice.

Rule 3: When, instead of a numbered column, Xm is encountered, apply rule 2 with these modifications: treat each occurrence of Xm as if it were a column of completed scope, saturated, after it was used with this rule, unsaturated before that. When Xm is the first unsaturated column to the right of column i, put m in the ith and i in the mth column. Do it twice if X is preceded by 2.

The scope of the column is completed when all the arguments of the head letter to the right of it are entered in the column. The column is saturated when all the arguments to the right as well as the (possibly double) functor to the left are entered.

The three rules will saturate all columns of every correctly built cipher of a graph. We may now say that a cipher is well formed if and only if rules 1, 2, and 3 saturate each column of its short form matrix. Whether all well-formed ciphers are ciphers of graphs of chemical compounds is debatable. We allow, *e.g.*, as well-formed ciphers strings of more letters than there are atoms in the universe. More important is the problem whether all chemical graphs have well-formed ciphers. Some relaxation of well-formedness is perhaps necessary when we consider unsaturated graphs and graphs of crystals or metals.[11]

To return to our example of benzoic acid, rule 1 leads from (15) to (16). Then we scan (16) from right to left. X1 has completed scope. Column 15 has its scope completed because the valency of H is 1 and there is one numeral, 15, under it. Next is 14. Its scope is not completed. We apply rule 2. The nearest unsaturated column on the right is 15. Thus we have

```
C  C  2O O  H  2C H  C  H  2C H  C  H  2C H  X1
1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
      *     *     *     *     *     *     *
                                  15
                                     14
```
(18)

We write asterisks here to indicate completion of the scope and we underline to indicate saturation. The next application of rule 2 is again to column 14 as its scope is not yet completed. Here we use rule 3, and obtain

```
C  C  2O O  H  2C H  C  H  2C H  C  H  2C H  X1
1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 ...
      *     *     *     *     *     *     *
14                                15
                                   1  14
                                   *
```
(19)

Now the scope of 14 is completed because its head letter is preceded by 2. We go to the next column with uncompleted scope. It is 12. Applying rule 2 we saturate 13 and applying rule 2 again we complete the scope of 12 by binding it doubly with 14.

```
C  C  2O O  H  2C H  C  H  2C H  C  H  2C H  X1
1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 ...
      *     *     *     *     *     *     *
14                                13    15
                                     12    14
                                  14    1
                                  14    *
                                   *    12
                                        12
```
(20)

Column 14 became saturated by this step. Applying rule 2 seven times

(11) See comments in section VI.

```
C  C 2O O  H 2C H  C  H 2C  H  C  H  2C  H  X1
1  2  3  4  5  6  7  8  9  10 11 12 13 14  15 ...
             *           *
14    *  5     7  *  9     11  *  13  *  15  *
         *  4              8  12 10  14 12  1  14
               8  6  10      *     14     *
                  10         8     *     12
                  *          8     10    12
                  6
```
(21)

The next column with incomplete scope is 2 (for 3 has completed scope, being preceded by 2). The nearest to the right unsaturated column is 3. After applying rule 2, column 2 will still have its scope incomplete and the next application of rule 2 will bind 2 with 4. Column 2 now having complete scope, we proceed to column 1 whose scope is its saturation. We will first connect it with 2, which step saturates 2, then twice with 6.

```
C  C 2O O  H 2C H  C  H 2C  H  C  H  2C  H  X1
1  2  3  4  5  6  7  8  0  10 11 12 13 14  15 ...
14 3  *  5  *  7  *  9  *  11  *  13  *  15  *
2  3  2  *  4  8  6  10 8  12 10 14 12  1  14
6  4  2  2     *     10    *     14     *
6  *           1     *     8     *     12
   1           1     6     8     10    12
```
(22)

All the columns are saturated in (22). The listing in each column is such that between the first numeral and the asterisk we have its arguments (to the right) and between the asterisk and the underlining its functors (to the left). For some processings of matrices we may use this distinction. For others (*e.g.*, for the test of synonymity in section V) it is not needed and asterisks and underlinings may be omitted. (22) differs from (17) only by the order in which the numerals appear in the columns. Clearly both are short forms of the same matrix which (together with the labels) is the graph 9.

Note that in the short form of a matrix its columns may be written in any succession. Thus instead of (22) we may write equivalently, omitting from the first line numerals and X

```
C C C  C  C  C  C  O O H H  H  H  H  H
1 2 6  8  10 12 14 3 4 5 7  9  11 13 15
2 1 1  6  8  10 1  2 2 4 6  8  10 12 14
6 3 1  9  8  13 12 2 5
6 3 7  10 11 14 12
14 4 8 10 12 14 15
```
(23)

If one stores a short form of a matrix instead of storing a graph or a cipher, for most purposes he does not need to store the columns with H, and he may omit from other columns these numerals which correspond to H columns. The short form matrix refers to each bond twice; each is recorded in two different columns. This was useful for counting saturation and will be useful for synonymity considerations in section V. Since the chemical graphs are undirected, *i.e.*, the bond does not distinguish between the two nodes it connects, the matrix representing a chemical graph is symmetric; hence the part below the diagonal contains all information (the diagonal has only zero elements). See Fig. 1. When convenient, we may therefore further abbreviate our short form matrix by omitting in each column entries smaller than the number of the column. This amounts to transcribing only the part of the matrix below the diagonal. (23) thus is reduced to

```
C  C  C  C  C  C  C  O  O
1  2  6  8  10 12 14 3  4
2  3  8  10 12 14
6  3     10    14
6  4
14
```
(24)

If further economy is required we can omit the first row of numerals, tacitly understanding that we will renumber the entire according to the order the letters appear in the top row. Thus instead of (24) we will have

```
C  C  C  C  C  C  C  O  O
2  8  4  5  6  7
3  8     5     7
3  9
7
```
(25)

This form of the reduced matrix may be useful and can be written in linear form

C 2 3 3 7 C 8 8 9 C 4 C 5 5 C 6 C 7 7 C O O (26)

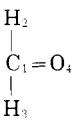
It is to be stressed that these linear forms of reduced matrices are by and large not shorter than the cipher explained in sections II and III. (26) is of the same length as (15). The short form matrix for benzpyrene (graph 5) is

C 2 10 10 C 3 3 C 4 8 C 5 5 C 6 7 C 7 16 16 C 8 13 13 C 9 9 C 10 11
C C 12 12 C 13 C 14 C 15 15 20 C 16 17 C C 18 18 C
19 C 20 20 C (27)

It is longer than the cipher (15).

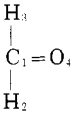
V. SYNONYMITY

The ciphers presented here are not unique as explained in section I. A graph has many ciphers. Note that if some nodes of the graph are undistinguishable (like two H,s attached to the same C) there will be no two different paths distinguishing between them. The very numbering of the nodes does not assure a path. Two seemingly different numberings of nodes may give the same path; *e.g.*, graph 1 in section 2 could be numbered



Graph 10

or could be numbered



Graph 11

but in both cases the path (and the cipher) is the same: C H H 2O. However, C 2O H H or H C H 2O or O 2C H H are different paths through the graph.

The problem remains of how to discover that two different ciphers are synonymous, *i.e.*, that they mark

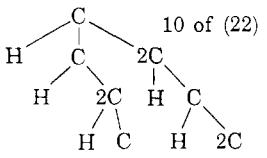
two paths of the same graph. In section IV an algorithm was presented to produce a short-form matrix out of a cipher. Two ciphers that are synonymous produce short-form matrices which differ only in numbering of the nodes; the cross references will be the same. If we are able to find a one to one mapping of elements of one short-form matrix into the elements of the other such that the mapping transforms the first short-form matrix into the other, then these matrices and the ciphers that produced them are synonymous. The problem of finding such a mapping, though in principal banal, in practice requires some ingenuity if we want to avoid very long computations. If we proceed by trial and error we will have an unmanageable amount of backtracking. The problem is misleadingly easy when one deals with such simple examples as graph 1. But even graphs of the size and complication of graph 9 present difficulties. As we have shown, a cipher for graph 9 is formula 15. Another cipher for the same graph is

C H C H 2C H C H 2C C H 2X1 C O H 2O (28)

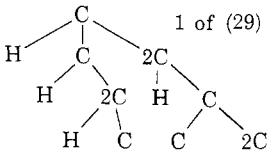
The algorithm for decoding (28) gives the short-form matrix

C	H	C	H	2C	H	C	H	2C	C	H	2X1	C	O	H	2O
1	2	3	4	5	6	7	8	9	10	11		12	13	14	15
10	1	4	3	6	5	8	7	10	11	10		13	14	13	12
10		5		7		9		12	1			15	12		12
2		5		3		9		7	1			15			
3		1		3		5		7	9			9			(29)

One may try to identify 1 of (22) with 1 of (29); they both are C; 1 of (22) is connected with 2,6,6,14 which are C,C,C,C, respectively, whereas 1 of (29) is connected with 10,10,2,3 which are C,C,H,C, respectively. Therefore it would be impossible to match arguments of 1 of (22) with arguments of 1 of (29). We will have to backtrack and start again to match 2 of (22) with 1 of (29)—which again would not be successful. Before we reach a correct match for the first element of (29) we may have to backtrack several times (it may easily be more than six); for second arguments may match but arguments of the second arguments may not. So, *e.g.*, if one starts matching 1 of (29) with 10 of (22) he will have the following comparison



Graph 12



Graph 13

It is only when one reads arguments of arguments of of arguments of the first C that one notices a difference; one has to withdraw from this attempt. To reduce backtracking and decide synonymity[12] of two ciphers we suggest a policy below. It will attempt to minimize at each step the probability of finding the synonymity. At each step we will look for a unique or least frequent partial path through the graph. Then a matching configuration in the matrix of the other cipher will be sought, step by step until synonymity is denied or the ciphers are exhausted. This results in writing a third cipher along the matching partial paths. If the two ciphers are synonymous with a cipher produced in this way, then they are synonymous.

A. Compare molecular formulas (by counting the number of occurrences of each letter); if they are not the same the ciphers are not synonymous.[13]

B. Choose the letter with the smallest occurrence. (It may happen that two, or more, letters have the same least number of occurrences, in which case we may take any of them for consideration.) If there is only one occurrence of it, state the new cipher by this letter as its first functor. In case there are more than one occurrence of this letter, compare the sets of their arguments and choose a unique combination of the arguments (*i.e.*, a different combination from the combinations of arguments with the other occurrences of the letter). Start the new cipher with this occurrence of the letter. In any step if two ciphers have the same arguments with the same functors compare arguments to these arguments until you find a unique argument.

We will illustrate. First to make the discussion more transparent we will copy (22) and (29) with letters added to each numeral as these numerals refer to the letters [see Fig. 2 for short-form matrices (22′) and (29′)]. (22′) and (29′) have identical sets of head letters. There are two O's. We start by, say, the last O in (29′). We will write O(C,C) to indicate that the O has connections with two C's. The other O, 13, is O(H,C). Therefore they are distinguishable. We easily find them in (22′). 15→3 indicates that 15 of (29′) corresponds to 3 of (22′); also 13→14, 12→2. We can start writing the new cypher which is a record of our procedure.

from (22′)	3	2	4	5	1	(30)
new cipher	O	2C	O	H	C	
from (29′)	15	12	13	14	9	

Among the arguments of 12 we have 15, already used and O and C. The O is 13→4 and C is 9→1. 9 has all its connections to C's, but one of the C's is double, 7; therefore 7 is uniquely distinguished from the others. We may proceed

3	2	4	5	1		(31)
O	2C	O	H	C	2C	
15	12	13	14	9	7	

From (22′) we find that 7→6, for 9→1 and in (22′) 6 is the only double argument of 1.

3	2	4	5	1	6	(32)
O	2C	O	H	C	2C	
15	12	13	14	9	7	

Among the arguments of 7 we have an H(8→7) and a C(5—8) besides 9 which was already used.

3	2	4	5	1	6	7	8	(33)
O	2C	O	H	C	2C	H	C	
15	12	13	14	9	7	8	5	

(12) The general problems of how to match two matrices and two chemical graphs in particular (with some suggestions similar to the course followed here) are dealt with in a paper by G. Salton and E. H. Sussenguth, Jr., "Automatic Structured-Matching Procedures and Some Typical Retrieval Applications," The Computation Laboratory of Harvard University (mimeographed).

(13) Note that in two synonymous ciphers the total number of terms (*i.e.*, an element symbol, a bond symbol, or an X together with its numeral) are equal, the number of element symbols of any given kind are equal, the number of double bonds are equal, and the number of variables are equal. Note also that the number of cycles in a graph is equal to the number of X's in its cipher.

C	C	2O	O	H	2C	H	C	H	2C	H	C	H	2C	H	X1	(22′)
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
14C	3O	2C	5H	4	7H	6	9H	8	11H	10	13H	12	15H	14		
2C	3O	2C	2C		8C		10C		12C		14C		1C			
6C	4O				1C		10C		8C		14C		12C			
6C	1C				1C		6C		8C		10C		12C			

C	H	C	H	2C	H	C	H	2C	C	H	2X1	C	O	H	2O	(29′)
1	2	3	4	5	6	7	8	9	10	11	...	12	13	14	15	
10C	1	4H	3	6H	5	8H	7	10C	11H	10		13O	14H	13	12C	
10C		5C		7C		9C		12C	1C			15O	12C		12C	
2H		5C		3C		9C		7C	1C			15O				
3C		1C		3C		5C		7C	9C			9C				

Fig. 2.—Short-form matrices (22′) and (29′).

The arguments of 5 are H(6) and doubly used C(3). Thus

3	2	4	5	1	6	7	8	9	10	(34)
O	2C	O	H	C	2C	H	C	H	2C	
15	12	13	14	9	7	8	5	6	3	

In the same way

3	2	4	5	1	6	7	8	9	10	11	12	13	14	(35)
O	2C	O	H	C	2C	H	C	H	2C	H	C	H	2C	
15	12	13	14	9	7	8	5	6	3	4	1	2	10	

10 has as its arguments H(11→15), C(1→12) doubly, already listed and C(9). Now, 9 though listed in (35) was not listed in connection with 10 and C(9→1) requires one more argument for saturation. A record to this effect can easily be kept, *e.g.*, by special marks on (29′). We have to list 9 as an argument of 10 but not as a functor. We use the X-notation. The 9 which we want to report occurs on the 5th place of the new cipher. Thus after C(10→14) we have X5. We check that the new X5 corresponds to 1 in (22′). We have succeeded in rewriting both (29′) and (22′) as

O 2C O H C 2C H C H 2C H C H 2C H X5 (36)

Therefore (29′) and (22′) are synonymous.

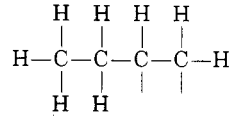
The program outlined here is relatively simple but it involves the reconstruction of the two short-form matrices. We may reflect that it is essentially a program to decide the identity of two chemical graphs, two matrices, two paths through the same graph.

There may be other algorithms for synonymity of two ciphers.

VI. FURTHER APPLICATIONS

The notational devices explained in sections II and III can be applied to other graphs, to the graphs of "open ended" compounds, to the graphs of metals, etc. We will introduce a new sign, a dash. It indicates a connection. In the cipher it should be counted like an unspecified X. We count it as an argument not as a functor. This time we do not refer to a letter already used in the cipher, but to a connection without stating the letter. As a rule the missing letter is not supposed to be any of those previously used in the cipher.

Let us illustrate by recording the following graph of butane linearly



Graph 14

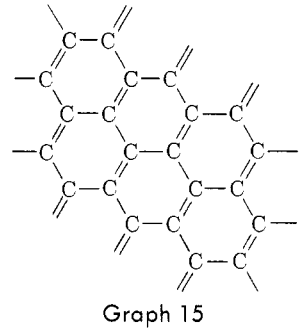
There are two C's with loose ends. Each of them will be recorded by a -.

C H H H C H H C H - C H H - (37)

The cipher for the saturated butane is

H C H H C H H C H H C H H H (38)

The graphs "without end", like the graph for graphite, are in practice drawn by the chemist as a finite graph with some open ends. We can scan such a graph recording those open ends. It may, again, be a very uninteresting path through the graph, but is is a correct linearization of the graph.



Graph 15

We can record it as, *e.g.*,

C - C 2- C 2C C C 2X1 - 2C - C 2- C 2C X4 C 2C C 2- X3

C 2- C - 2C C - 2C - C 2- C 2C X16 X11 C 2- X9 (39)