Encoding and Decoding WLN

GEORGE A. MILLER*
Heuristics Laboratory, Division of Computer Research and Technology,
National Institutes of Health, Department of Health, Education and Welfare,
Bethesda, Md. 20014

Received August 25, 1971

This paper deals with the encoding and decoding of a Wiswesser Line Notation (WLN). This problem so far has been addressed only from the point of a human. This paper discusses the encoding and decoding with exactness suitable for a computer, and is an outgrowth of a computer program now in operation at NIH which automatically encodes and decodes WLN.

The computer system for encoding and decoding WLN[1] can be broken down into four separate programs (see Figure 1). Two of the programs, Rand Tablet Program and Display, are used to facilitate the communication between the chemist and the computer.

The Rand Tablet Program allows a chemist to sketch a chemical structure free-hand on a special pad. The program then converts the information into a connection table for further processing by the Encoding Program. Examples of free-hand sketches of two chemical structures are given in Figure 2.

Conversely, the Decoding Program converts a connection table into a two-dimensional chemical structure. Examples of the results of this program are given in Figure 3.

Once the input/output considerations are put aside, we can concentrate on the core of the problem, namely, encoding and decoding. There is a further division which naturally suggests itself. Chemical structures can be broken into their acyclic and cyclic parts and handled separately.

*Present address: University of Pennsylvania, Moore School of Electrical Engineering, Philadelphia, Pa. 19140.

As one might expect the encoding and decoding programs are similar. The decoding algorithm seems to follow the encoding algorithm precisely in reverse. This paper discusses the general aspects of both algorithms glossing over most of the minor details and exceptions which make an algorithm tedious.

ENCODING ACYCLIC STRUCTURES

First assume that the chemical structure to be encoded is entirely acyclic. We can abstract the problem as follows: consider an acyclic network with each node being a letter as in Figure 4.

The linear notation for such a network or chemical structure will be a particular permutation of the letters of the nodes. The permutation is decided by the following abstracted Wiswesser rules:

Rule 1. Cite all chains of nodes letter-by-letter as connected

Rule 2. Resolve all otherwise equal alternatives in letter sequence by selecting the sequence that would be biggest

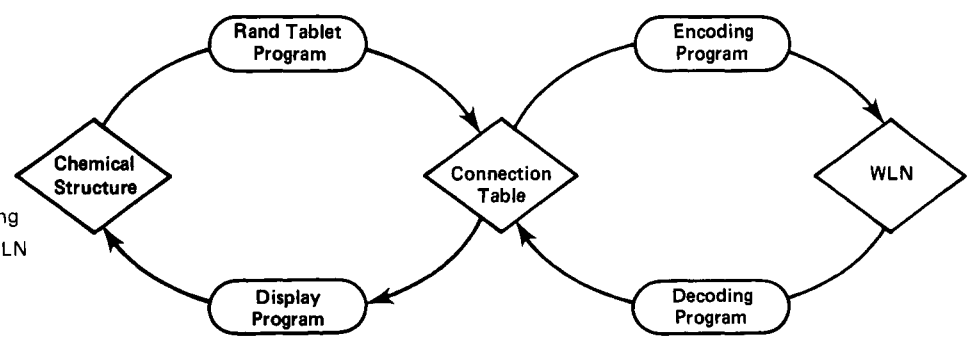
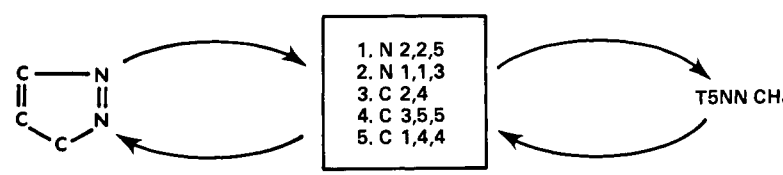
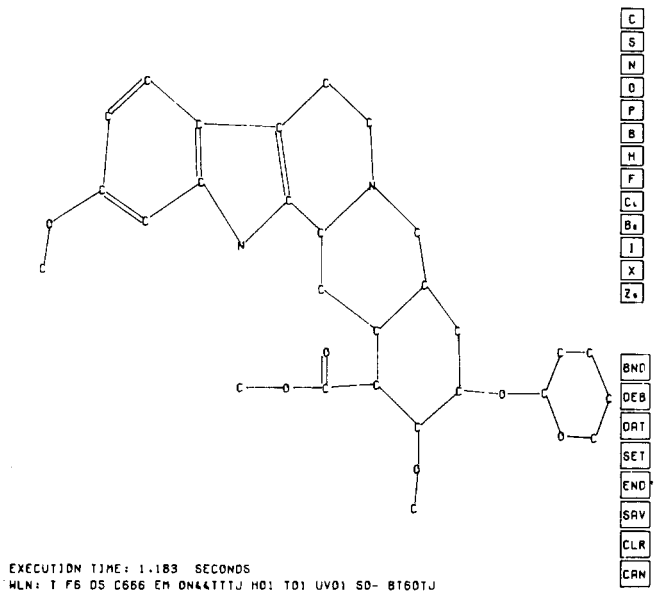


Figure 1. Over-all view of programming structure for encoding and decoding WLN

Example:





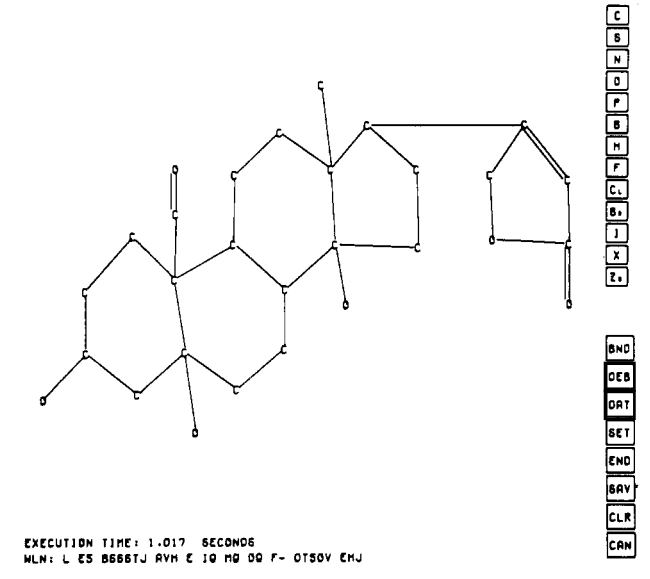
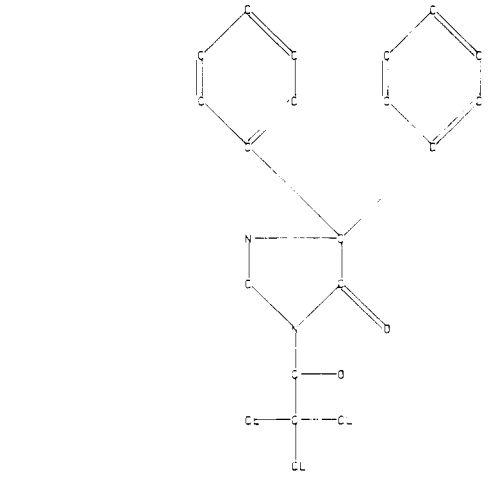


Figure 2. Sample results from the encoding program



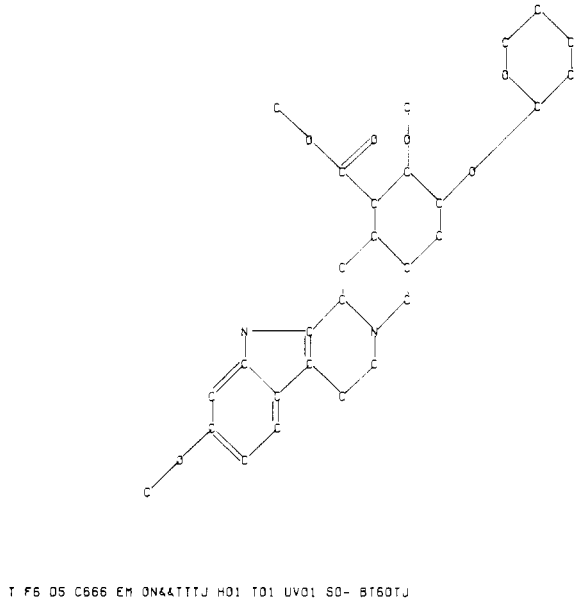


Figure 3. Sample results from the decoding program

Rule 3. Cite branched structures along that chain of nodes which includes, first, the largest possible number of branch nodes (a node connected to three or more other nodes) and after this, the largest possible number of nodes; start at the end of this chain required by Rule 2 and then follow Rule 4.

Rule 4. After each branch node, cite first, in the following order of choice, (a) the chain with the fewest branch nodes; and after this (b) the chain with the fewest nodes; and after this (c) the biggest chain.

Note that "bigger" is a relation defined for the sequence of letters of the same length. One sequence is said to be bigger if it follows another in alphabetic series. Conversely, one sequence is said to be "smaller" if it preceeds another in alphabetic series—e.g. "TWEF" is bigger than "TWEA" and "ABEDL" is smaller than "BBEDL".

Thus, the correct sequence for the above example would be:

Y V T Z I T I D C G B P C G B M

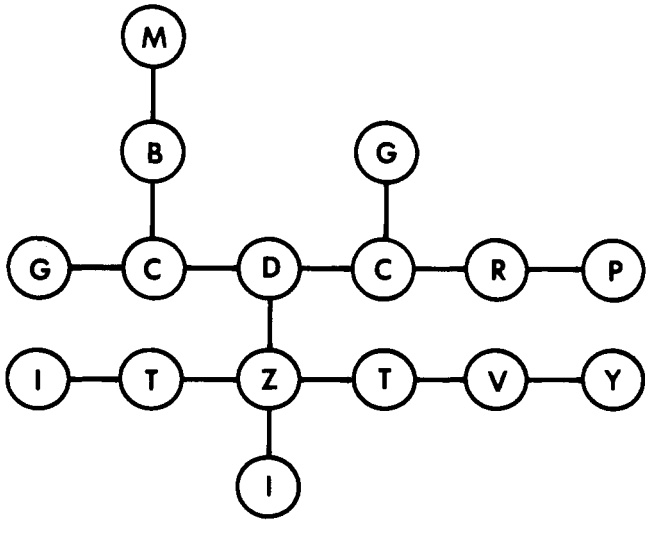


Figure 4. Example of an abstracted chemical structure

The rules explain what the correct coding should be but fail to indicate how a structure is to be encoded. The algorithm employed by the author can be thought of as a parallel process where the ends of the structure are "eaten" away, node by node, until there is only one node left, then that too is "eaten"; however, whenever a node is "eaten," the notation for the structure *up to this point* is formed so that when the structure is gone, the notation will remain.

Since we do not have a parallel processor, the algorithm proceeds in the following sequential manner:

> Step 1: Calculate the connectivity of all the nodes. Put a triangle around all the terminal nodes (nodes connected to just one other node) and put a square around all nodes connected to two other nodes. If no triangles were drawn, go to Step 4. (This is a weakness of WLN; in fact a new notation has been proposed in which the connectivity would be inherent and not required calculation.)
>
> Step 2: Find a triangle. If it is not connected to anything go to Step 4, otherwise eat it. If you can't find any go to Step 1.
>
> Step 3: If the node just eaten was not connected to a square, go to Step 2, otherwise eat this square and repeat this step.
>
> Step 4: There should be just one node left, devour it, and you are finished.

How is a node eaten? The process of eating a node not only destroys it but also results in four by-products which are passed along to the connecting node (realize that there is only one connecting node since only terminal nodes are eaten). The four by-products are:

1. The number of branch nodes counted to this point
2. The number of nodes counted to this point
3. The forward sequence of letters following rules 1-4 to this point
4. The backward sequence of letters following rules 1-4 to this point

The forward sequence is that sequence of notation going from the node just eaten to its connecting node. Similarly, the backward sequence is the notation going in the other direction. Both forward and backward sequence must be carried along since the correct direction will not be established until the end of the process.

The remaining bit of explanation describes how the four by-products are calculated. When a node is to be eaten, it is either the last node left or not. First we shall consider the latter case. The expiring node might have a number of lists of by-products from nodes which were previously connected to it (a list is simply an enumeration of the four by-products). All of these lists will be combined into a new list and passed on to the connecting node. The first by-product, number of branched nodes, is the sum of the numbers in the connecting lists plus one if the node being eaten has two or more lists attached to it. The second by-product is the largest number in the connecting lists plus one. The third by-product or the forward sequence is formed by first finding the sequence S1. This sequence is the one among the forward sequences of the connecting lists containing the most number of branch nodes, the most number of symbols, and the biggest. The new forward sequence is formed by concatenating the sequence, S1, with the letter of the eaten node with the remaining *backward* sequences in order of least number of branches, least number of nodes, and bigness. The fourth by-product of the backward sequence is formed by concatenating the letter of the eaten node with the backward sequence of the connecting lists in order of least number of branches, least number of nodes, and bigness. The process of devouring is exactly like that of eating a node except that the fourth by-product, the backward sequence, is calculated differently. The backward sequence for devouring is formed by first finding the sequence S2. This sequence is the *second* one among the forward sequences of the connecting lists containing the most number of branch nodes, the most number of symbols, and the biggest. The new backward sequence is formed by concatenating the sequence, S2, with the letter of the devoured node, with the remaining *backward* sequences in order of least number of branches, least number of nodes and bigness.

To alleviate any confusion at this point the acyclic structural example is encoded step by step as shown in Figure 5, which is based on a node eaten at each step.

The algorithm terminates with a list containing a forward sequence and a backward sequence. The resultant notation will be the bigger of the two sequences. In the case of the example, the resultant notation is the forward sequence or YVTZITIDCGBPCGBM. This algorithm is detailed elsewhere.[2]

DECODING ACYCLIC NOTATION

Here also the problem must be abstracted and grossly simplified in order to present the basic algorithm in a lucid fashion. The problem is to start with the notation of an acyclic chemical structure and to produce the structure. The notation can be thought of as a sequence of N symbols designated by S_i where i varies from 1 to N.

A new sequence of numbers, V, is produced from S by table lookup. V_i represents the valence of the chemical symbol S_i. It will be assumed that the valence is known and constant for each symbol when in reality many chemicals are variable valent. Next the sequence P is formed by the following formula $P_i = P_{i-1} + V_i - 2$ for $1 \leq i \leq$" (P_0 is defined to be 2). Finally the sequence F is created. The algorithm entails incrementing the value of i from 1 to N − 1. If the value of V_i is 1 do nothing otherwise, set F_{i+1} to i. Now if the value of V_i is 2 do nothing, otherwise, successively decrease the value of V_i and P_i by 1 until V_i is equal to 2. Each time the value is decreased, look ahead along the P sequence until a value of P_j is found equal to P_i. At this point set F_{j+1} equal to i. At the end of it all, set F_2 equal to 1. This algorithm is clearly outlined in Figure 6. The conclusion of the process results in the creation of the sequence F. This series of numbers describes how the structure is formed: First write the

> Step 1: numbers from 1 to N, then
>
> Step 2: go through F and for every i ($1 \leq i \leq N$) draw a line between the number i and the number F_i, finally,
>
> Step 3: substitute S_i for every number i and the structure is drawn.

To illustrate, we shall use the notation encoded above:

N = 16

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S:	Y	V	T	Z	I	T	I	D	C	G	B	P	C	G	B	M
V:	1	2	2	4	1	2	1	3	3	1	2	1	3	1	2	1
P:	1	1	1	3	2	2	1	2	3	2	2	1	2	1	1	0
F:	-	1	2	3	4	4	6	4	8	9	9	11	8	13	13	15

The steps are shown in Figure 7.

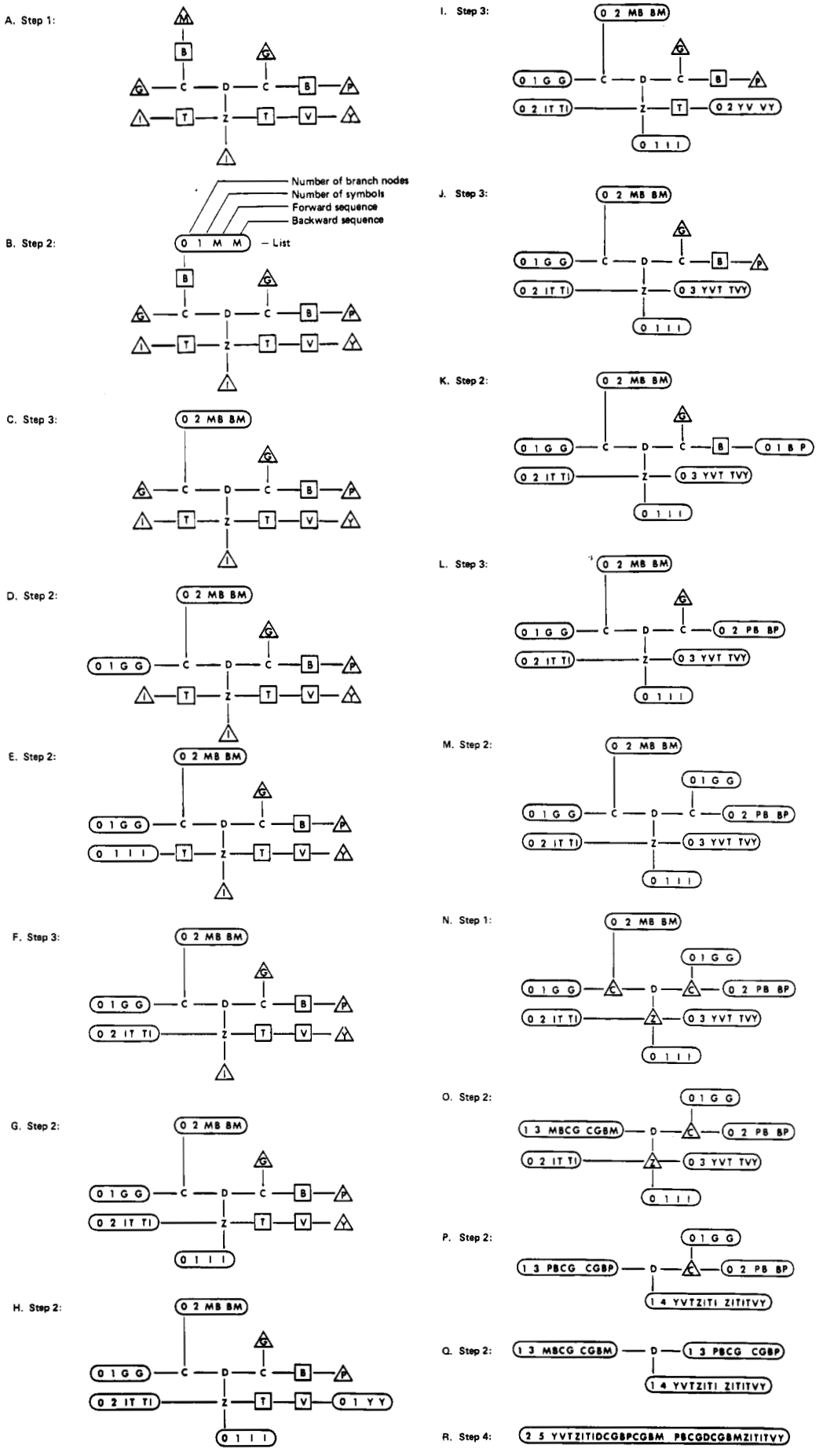
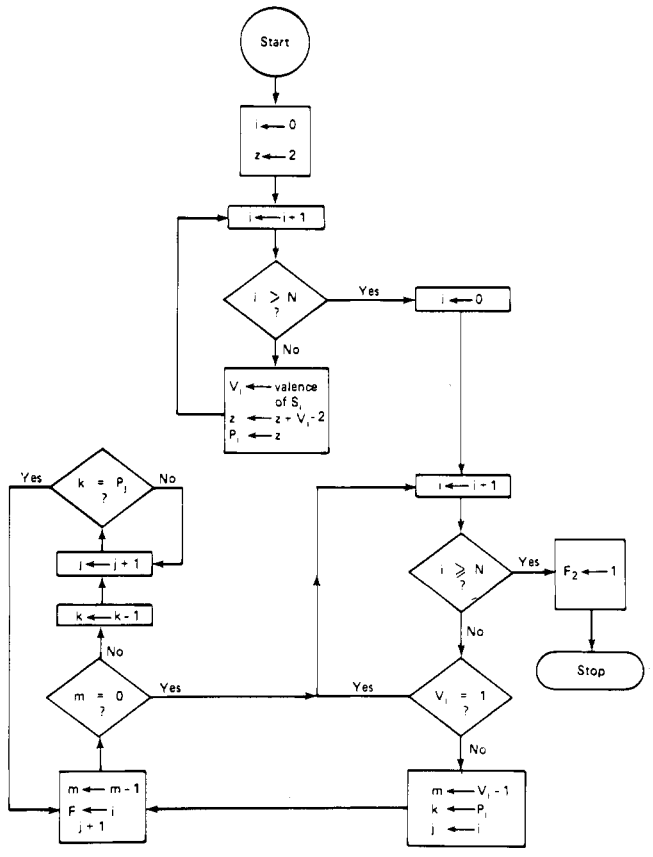

A. Step 1:
B. Step 2:
Number of branch nodes
Number of symbols
Forward sequence
Backward sequence
0 1 M M
– List
C. Step 3:
0 2 MB BM
D. Step 2:
0 1 G G
E. Step 2:
0 1 1 1
F. Step 3:
0 2 IT TI
G. Step 2:
H. Step 2:
0 1 Y Y
I. Step 3:
0 2 YV VY
J. Step 3:
0 3 YVT TVY
K. Step 2:
0 1 B P
L. Step 3:
0 2 PB BP
M. Step 2:
N. Step 1:
O. Step 2:
1 3 MBCG CGBM
P. Step 2:
1 3 PBCG CGBP
1 4 YVTZITI ZITITVY
Q. Step 2:
R. Step 4:
2 5 YVTZITIDCGBPCGBM PBCGDCGBMZITITVY

Figure 5

Figure 6. Flowchart for decoding acyclic notation



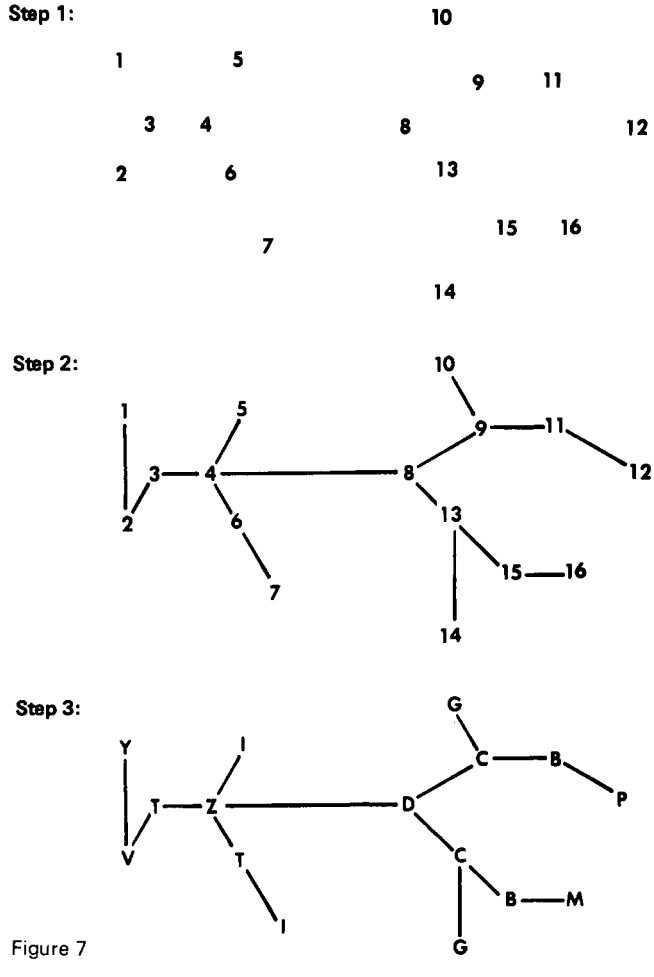


Figure 7

ENCODING CYCLIC STRUCTURES

To simplify the explanation, only ring systems will be considered. A ring system is a graph made up of nodes and edges with one restriction.

For any two nodes there exists at least two distinct paths connecting the nodes. (A path is a sequence of connecting edges where no edges are repeated.) Examples of ring systems are given in Figures 8 and 9.

The ring systems under consideration will be further restricted to those which contain a complete path (one using all edges).

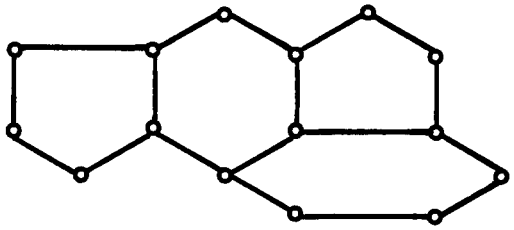


Figure 8. Example of a ring system

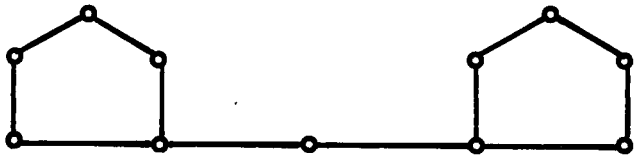
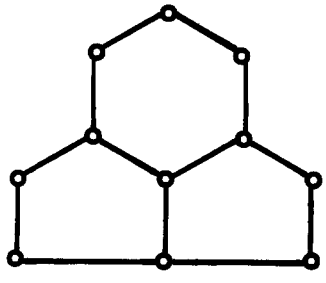
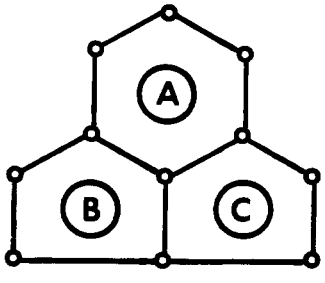


Figure 9. Example of two connecting ring systems

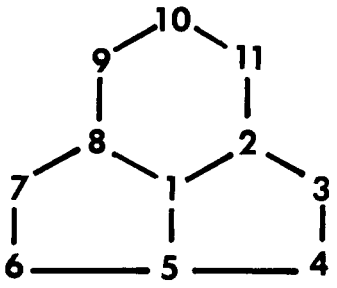
The notation is formed first by identifying each ring in the ring system. For each complete path two sequences of numbers are formed, L and M. If each node is numbered sequentially starting with 1 in order along the complete path, then M is formed by the number of nodes in each ring in order of lowest number of highest numbered nodes in each ring. The sequence L is formed using the lowest numbered node for each ring in the same order of rings as used for M. With such an explanation an example is in short order.



Step 1: Find three rings, A, B, & C



Step 2: Find a complete path



Note that the three rings consist of the following numbered nodes:

A: 1,2,11,10,9,8
B: 1,8,7,6,5
C: 1,5,4,3,2

Step 3: Calculate sequences L and M:

M: 5,5,6 since C has 5 nodes R has 5 nodes and A has 6 nodes.

L: 1,1,1 since C, B, and A all have 1 as their lowest numbered node.

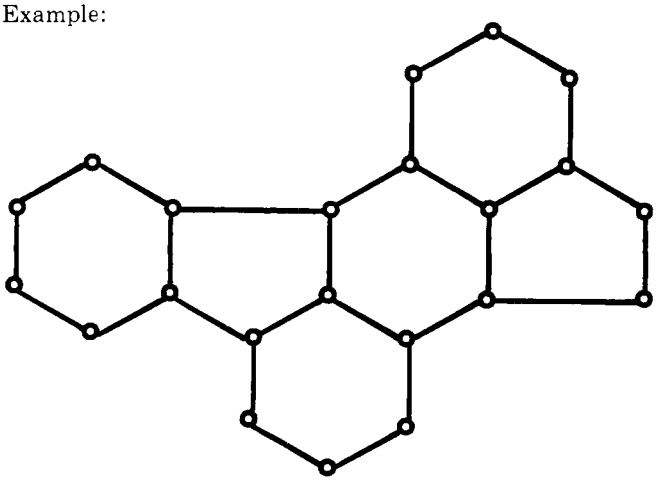
Notice that the highest numbered nodes for each ring were all unique thus defining a unique ordering. Later, we shall restrict the choice of complete paths to guarantee uniqueness in the ordering of rings.

The actual notation can be quite easily calculated from the sequences L and M, so the job of encoding cyclic structures will be considered complete when L and M are found. In general, different complete paths will produce different sequences L and M. The one final step in calculating the correct notation is to choose among all the different sequences of M and L, one set according to the following ordered criteria:

1. lowest sum of L
2. lowest sequence L
3. lowest sequence M

In other words, when comparing sets of sequences, add up the numbers in each L sequence and choose the smallest sum. If the sums are the same then compare the L sequences and choose the lowest (eg 1,2,14 is lower than 1,3,6). If the L sequences are the same, then compare the M sequences and choose the lowest.

Example:



For the above structure, the best sequences L and M are

L: 3,1,1,1,15,14
M: 6,5,6,6,6,5

The above description of notation is almost constructive enough to be left without further comment. However, it turns out that identifying the rings and finding the paths require more explanation.

If the structure is thought of as a conglomeration of bubbles, then, the rings can be found by "popping" the bubbles one at a time. The act of "popping" can be described by the following steps:

Step 1. Find a node with only two edges. Erase it. Go to one of the two nodes this connects to.

Step 2. If this node is connected to three or more others including the one just erased, call it A and go to Step 3, otherwise erase this node and repeat Step 2.

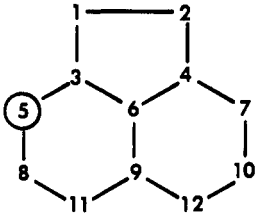
Step 3. Go to the other node connected to the one first erased.

Step 4. If this node is connected to three or more others including the one just erased, call it B and go to Step 5, otherwise erase this node and repeat Step 4.

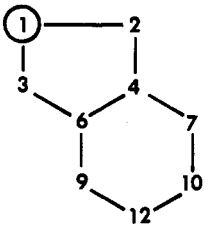
Step 5. Find the shortest path between node A and node B. If there is no path go to Step 1, otherwise, form a list, R, consisting of the nodes in the path from A to B and the nodes which were erased. Stop

Each time a structure is popped, a list, R, of nodes is formed. This list describes a ring of the structure. Successive popping of the structure will result in a description of all the rings in the structure. An example is shown in Figure 14.

Start:

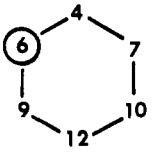


Pop 1:



R1 = 3,6,9,5,8,11

Pop 2:



R1 = 3,6,9,5,8,11
R2 = 6,4,1,3,2

Pop 3:

R1 = 3,6,9,5,8,11
R2 = 6,4,1,3,2
R3 = 6,9,12,10,7,4

Figure 14

After all of the rings have been identified, a sorting of structures is done. If there is only one ring then the notation is M: n and L: 1 where n is the number of nodes in the structure. If there is any node appearing in more than three rings, then the structure is a spiro ring system and cannot be handled by this encoding method. If there is any node which has only two edges but appears in more than one ring, then the structure is a bridged ring system and also cannot be handled by this method. The remaining structures, which fortunately make up the majority of known ring systems, can be further categorized into two groups, polyfused and perifused. Perifused ring systems are those remaining (after removal of monorings, spiro, and bridged) ring systems which contain at least one perifused node (a node appearing in three rings). Those remaining ring systems not having any perifused nodes are called polyfused.

Very little remains to be explained. For perifused ring systems, we must find all of the complete paths, calculate the sequence L and M for each path, and choose the best set of sequences. If there exists no complete path, then the perifused ring system is branched, and the encoding of branched perifused ring systems is not discussed in this paper. One heuristic applied is to consider only those complete paths starting with a perifused node. To calculate all the complete paths in a ring system from a given node a simple backtracking procedure[3] is employed.

For polyfused ring systems consider only those complete paths starting with fused nodes (nodes appearing in two rings). The final notation will consist of the two sequences, L and M, and the number of nodes n. Keep in mind that the procedure is limited to monocyclic, polyfused, and unbranched perifused ring systems. It should be noted that the encoding programs are parts of a larger program to encode automatically in WLN.[4]

DECODING CYCLIC STRUCTURES

This section will be concerned with the transformation of notation into structure. The notation is in the form of two sequences L and M of numbers and the number of nodes, n, of the structure. Implicit is the fact that there is a complete path through the structure. We shall start with this fact and construct a connection table of nodes; from the connection table, the structure can be drawn.

It is known that there are n nodes, so let us number them 1 through n; in fact, this can be the complete path. The connection table will be in the form of n sequences Ci ($1 \leq i \leq n$). Each sequence C will contain the numbers of the nodes connected to node i. Since our numbering is along the complete path we know that node i is connected to nodes $i - 1$ and $i + 1$. Therefore, we can start filling the sequences:

1. put $i - 1$ and $i + 1$ in Ci for all i ($2 \leq i \leq n-1$) (let $i - 1$ be on the left and $i + 1$ on the right)
2. put 2 into C1
3. put $n - 1$ into Cn

Example: If n = 4 then

C1 = 2
C2 = 1,3
C3 = 2,4
C4 = 3

and the structure thus far described would be:

1——2——3——4

The connection table now describes a linear chain of length n. The sequences L and M will be used to add fusion bonds forming the rings. One fusion bond will be added for every number in the sequence. Assume that j is the next number in the sequence L and k is the next number in the sequence M. Initially, set r equal to j.

Step 1. If C_r has less than three numbers go to Step 2, otherwise, decrement k by 1, increment r by 1, and repeat this step.

Step 2. If k equal 1 go to Step 3, otherwise, decrement k by 1, set j equal to the rightmost number in the sequence C_j and repeat this step.

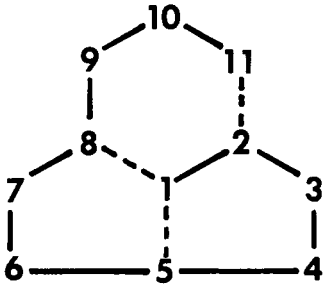
Step 3. Add the number j to the right side of the sequence C_r and add the number r to the left side of the sequence C_j.

If the above algorithm is applied to the set of sequences Ci ($1 \leq i \leq n$), for each pair of numbers in the sequence L, and M, at the end Ci will be the intended connection table.

Example: n = 11
L = 1,1,1
M = 5,5,6

	Initially	j = 1 k = 5	j = 1 k = 5	j = 1 k = 6
C1	2	2,5	2,5,8	2,5,8
C2	1,3	1,3	1,3	1,3,11
C3	2,4	2,4	2,4	2,4
C4	3,5	3,5	3,5	3,5
C5	4,6	1,4,6	1,4,6	1,4,6
C6	5,7	5,7	5,7	5,7
C7	6,8	6,8	6,8	6,8
C8	7,9	7,9	1,7,9	1,7,9
C9	8,10	8,10	8,10	8,10
C10	9,11	9,11	9,11	9,11
C11	10	10	10	2,10

The fourth column in the above example is the final connection table. From this connection table the structural diagram must be drawn. The procedure for doing this so that the structure is drawn neatly, free of crossed lines, and the way chemists like to see them is material enough for another full paper. For the sake of completeness, though, the above example would be drawn as so:



CONCLUSION

Much of the detail has been glossed over in an attempt to present the fundamental principles of encoding and decoding WLN as succinctly as possible. One of the omissions was an explanation of how to tie the acyclic structures together with the ring system since most chemical structures are combinations of the two. This problem in general

can be solved by treating a ring system as a node on an acyclic tree and then in turn treat a substituent acyclic structure as a node in a ring system.

What is presented in this paper seems relatively straightforward; however, if you must write a computer program to encode and decode actual WLN you will have to consider such details as double bonds, methyl contractions, carbon chains, variable valancy, and a host of other stumbling blocks. This paper is a result of such a program written and working at N. I. H. Figures 2 and 3 show examples of the program's products.

As the system stands, it is limited to unbranched perifusion and cannot handle multipliers or salts or ions; but, there is no reason why these restrictions could not be lifted if occasion and resources called for them.

Some of the applications of a program to encode and decode WLN are helpfulness in chemical information and retrieval systems,[5] an inexpensive method for chemists to communicate with a computer, and standardization of chemical notation.

The important conclusion from this project is that a new or improved line notation should be developed with automatic conversion in mind. We have reached the point where a complete repository of all chemical information with immediate automatic access by all chemists is needed. Clearly, a standardized notation is called for which is easily understood by both men and machines.

LITERATURE CITED

(1) "The Wiswesser Line Formula Chemical Notation," Elbert G. Smith, McGraw Hill, 1968.

(2) "*CARL*—Chemical Algorithm for Reticulation Linearlization," George A. Miller, internal documentation, Heuristics Laboratory, NIH, Bethesda, Md., August 1970.

(3) "Program Development by Stepwise Refinement," Vol. **14**, No. 4, pp. 221–8, Niklaus Wirth, CACM, April 1971.

(4) "Computer Generation of Wiswesser Line Notation," C. D. Farrell, A. R. Chauvenet, and D. A. Koniver, *J. Chem. Doc.* **11**, 52 (1971).

(5) "An Application of Interactive Computing: A Chemical Information System," Richard J. Feldmann and Stephen R. Heller, presented at the first workshop of International, Regional and Collaborative Reference Centers WHO- NIMH for Information on Psychotropic Drugs, Plitvice Lakes, Yugoslovia, June 1971.

Division of Chemical Literature Program

163rd Meeting, ACS, Boston, Mass., April 9–14, 1972
Stephen J. Tauber, Chairman
Charles E. Granito, Secretary

MONDAY MORNING

(April 10, 1972)

General—Computerized Manipulation, Storage, and Retrieval of Chemical Structural Information

A. E. Petrarca, Presiding

9:00		Introductory Remarks. A. E. Petrarca.
9:05	1.	Use of a Multi-Level Substructure Search System—Survey of User Queries. E. Hyde, D. R. Lambourne, L. A. McArdle.
9:35		Discussion.
9:40	2.	Substructure Searching by Set Reduction. J. Figueras.
10:10		Discussion.
10:15	3.	On the Nested Retrieval of Chemical Structures. R. J. Feldmann, S. R. Heller.
10:35		Discussion.
10:40	4.	Appraisal of Methods of Representing Organic Molecules—Case for Interconversion. J. Ash, E. Hyde, D. R. Lambourne.
11:10		Discussion.
11:15	5.	Computer Generated Wiswesser Line Notation from Graphical Input. D. A. Koniver, S. R. Heller.
11:35		Discussion.

MONDAY AFTERNOON

Symposium on Evaluation of Existing Chemical Information Services

B. M. Vasta, Presiding

2:00		Introductory Remarks. B. M. Vasta.
2:10	6.	Assessment of Computer-Based Bibliographic Retrieval Services. J. L. Carmon.
2:30		Discussion.
2:40	7.	Comparison of Service Centers and Document Data Bases—A User's View. C. H. O'Donohue, M. A. Manzelli, B. P. Walford.
3:00		Discussion.
3:10	8.	Literature Sources for Hazard Evaluations of Chemicals. H. H. Fawcett.
3:30		Discussion.
3:40	9.	Comparative Evaluation of Ringdoc and CBAC. J. S. Buckley, T. K. Devon, E. D. Taylor.
4:00		Discussion.
4:10	10.	Comparative Searching of Computer Data Bases. R. O. Beauchamp, Jr., M. Daugherty, J. L. Garber, J. D. Meyers.
4:30		Discussion.