Chemical Symbol String Parser†

J. FIGUERAS

Research Laboratories, Eastman Kodak Company, Rochester, New York 14650

Received August 30, 1982

A program is described that can accept chemical symbol strings and produce a connection table therefrom. The program, which is a subroutine in a structure entry program, allows input of functional groups and whole structures in notation that is familiar and natural to the organic chemist. Nested parentheses, contracted alkyl groups, functional groups containing subscripts, subscripted atoms, and condensed carbonyl and sulfonyl groups are accommodated. Error routines and warnings are incorporated to guard against ambiguous structure formulation by the user.

INTRODUCTION

Symbol strings play an important role in the chemist's vocabulary for structure representation. They occur widely in textbooks and chemical handbooks and are used for concise representation of substituents and connective and whole structures. Many structures are more conveniently represented as linear strings than as drawn structures; dodecane is more clearly represented as $C_{20}H_{42}$ than as a written-out chain of 20 carbon atoms. Fairly complex molecules can be represented by symbol strings; for example, the *N*,*N*-diethyl amide of acetoacetic acid is rendered intelligibly and compactly as $CH_3COCH_2CON(C_2H_5)_2$.

In the development of programs to handle chemical information, we consider facile handling of chemical structure input to be a first requirement in creating a "user friendly", familiar interface between our programs and chemist-users. We wanted to produce a system that allows the chemist to use structure representations that are natural to him, within the limits of computer hardware. A component of that "naturalness" is the chemist's informal language for the expression of chemical structure in the form of symbol strings. That language has a number of conventions that are widely used: for example, condensed forms such as C_2H_5 for an ethyl group, CO to represent a carbonyl group, with implied branching at the C–O bond, parenthesized strings to indicate branching, and so on. The fact that the symbol string language is informal complicates the writing of a program to interpret that language on a computer. However, certain practices appear to be universally used. For example, hydrogen usually follows and is immediately adjacent to its attached atom, except for "reversed" groups such as H_2N or $H_{11}C_5$, and the "reversed" configuration is easily recognized in a program by the occurrence of hydrogen in a left, terminal location. Another universal rule is that carbonyl compounds formulated as strings of the form COX require that X be attached to the carbonyl carbon atom. These easily recognized "universal" rules, plus constraints associated with geometry and valence, make it possible to treat the interpretation of chemical symbol strings algorithmically. A statement of rules is contained in the Appendix.

In this paper I describe a computer program that can accept symbol strings representing partial or whole structures and can translate those strings into a connection table. An initial approach to chemical symbol string parsing was published by Barker in 1975,[1] but Barker's parser was extremely limited in concept and was not designed to produce a connection table. It was, in fact, incapable of handling intricate chemical structures. The parser presented in this paper is used in a structure entry program and extends the capability for structure entry, allowing facile formulation of connectives, substituents, and whole molecules, which are typed on the keyboard of the terminal and placed on the CRT screen by means of a graphics tablet and stylus. We have found that formulation of connectives and substituents in this fashion, rather than by assembly from a menu, makes for rapid and convenient structure input. We have also found that the natural input language, since it hews so closely to standard practice among chemists, is very easy to learn. Computer-naive chemists are able to formulate and enter their own structures with about 30 min of personal instruction, and they report that they enjoy using the program.

† Presented in part at the 182nd National Meeting of the American Chemical Society, New York, Aug 23–28, 1981.

GRAPHICS INPUT, THE SOURCE OF THE STRUCTURE TABLE

The structure entry program in which the parser is embedded contains a group of subroutines that allows the chemist-user to create a picture of an acceptable structure on the CRT of the terminal. These subroutines are those commonly found in other structure entry programs:[2] create chains of carbon atoms, create multiple bonds, create rings of various sizes (aromatic and nonaromatic), insert heteroatoms, and so on. As mentioned above, symbol strings entered from the keyboard can be incorporated into the growing structure. The graphics input program is written for a Hewlett-Packard 2647A terminal. A graphics tablet is used for selection from a menu for placing structural elements in the work area. The programs were written in ANSI '66 FORTRAN IV and are implemented on an IBM 3033 computer under the Time Share Option (TSO). The graphics portion of the program does little interpretation (only enough to detect certain errors or ambiguities). The structure table created in this part of the program is passed to the INTERPRETER (described below) to obtain the final connection table.

AMBIGUITIES

Two kinds of ambiguity are encountered in the use of strings to represent structure: ambiguity of *representation* and ambiguity of *interpretation*. With respect to the first kind (representation), almost every symbol string is ambiguous because information about structure is deliberately discarded in order to obtain a compact string representation. The missing information is supplied by the chemist, who must *interpret* the string to extract meaning from it. It is evident that if such strings are to be handled by computer, the computer must *interpret* those strings in the same way that a chemist would. Consider the example of acetone, CH_3COCH_3; the chemist deduces from this string that the oxygen atom is doubly bonded to the preceding carbon atom, which in turn must be attached to the rightmost methyl group. This is part of the missing information that the chemist supplies when interpreting this string.

0095-2338/83/1623-0048$01.50/0

Allowed Symbol	Application
numerals	charge quantifier, subscripts
+, -, .	charge; radical center
$( )_n$	chain branching; group replication when used with a subscript
letters, caps and lower case	atom names
#	dummy atom used in substructure search queries to terminate a "dangling" bond
:	bond symbol for use within strings
hyphen	connecting point (left end or right end) for the input string; use is optional

Figure 1. Allowed symbols for string formulation.

Ambiguity of *interpretation* arises when certain symbol strings may represent several different structures. For example, although $CH_3SO_2CH_3$ would *probably* be interpreted as representing dimethyl sulfone, in certain contexts it might represent the methyl ester of methanesulfinic acid. Ambiguity of this type may be taken care of by arbitrarily selecting one of several interpretations as "the" interpretation. This means, of course, that a *rule* has to be enunciated for making this selection. The user of an interpretive program such as the one described here then faces the difficulty of knowing or remembering which interpretation prevails. The solution to this problem is to cause the program to display its interpretation of an ambiguous structure (feedback) so the user can decide whether the interpretation is that which was intended. In the program described here, groups subject to ambiguous interpretation are flagged, and "help" appears on the screen that describes succinctly how the groups will be interpreted.

OTHER PROBLEMS

Aside from ambiguity, formalization of interpretation of symbol strings is made difficult by the fact that *multiple* ways exist for drawing the same structure. For carboxyl, for example, six representations are possible: one in which all bonds appear explicitly and COOH, CO_2H, HOOC, HO_2C, and HOCO. Parallel cases can be written for acid moieties containing sulfur or phosphorus and for esters of all of these substances.

Determination of the multiplicity of an implied multiple bond in a symbol string is based not only on the valencies of the bonded atoms but also on context. In acetone, CH_3CO-CH_3, for example, the determination of the multiplicity of the bond linking oxygen and carbon must take into account the potential bond that will exist between the (future) carbonyl carbon atom and the terminal methyl group. The examination of the environment that leads to determination of bond multiplicity is complicated by the fact that the structure entry program would also allow acetone to be entered as an assembly of substrings, as in the structure below.

From	Symbol	To-Attachments	Bonds
1	C	2	0
2	H	3	0
3	3	4	0
4	C	5	0
5	O	6	0
6	C	7	0
7	H	8	0
8	3	-	-

Figure 2. Structure table for string CH_3COCH_3.

From	Atomic No.	To-Attachments	Bonds
1	6	2	1
2	6	3 4	2 1
3	8	-	-
4	6	-	-

Figure 3. Connection table to be extracted from the structure table in Figure 2.

STRING REPRESENTATION

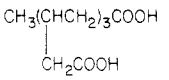
The interpretive program uses a tabular representation of the input string, which I refer to as the "structure table". This table resembles a connection table, except that the entity represented is a string of alphameric symbols rather than a chemical structure. The allowed symbols are summarized in Figure 1. For acetone, the structure table is shown in Figure 2. This table is developed by computer analysis of the input string. The last column assigns a bond order of zero for all those symbols that are not connected by *explicit* bonds, i.e., bonds that are explicitly shown in the structure diagram, as in the branched drawing for acetone above.

The function of the string interpreter is to transform the structure table into a connection table that has chemical significance; that is, the table in Figure 2 is to be converted into the one shown in Figure 3. In the sections that follow, the interpreter program is described with examples of its conventions and scope. The design of the interpreter determines the kinds of structure that the program "understands"; this design is based on conventions used by organic chemists for representing chemical structure. The conventions or rules used by the interpreter are stated in the Appendix.

INTERPRETER

Symbol Conversion. Atom symbols in the structure table (Figure 2) are replaced by atomic numbers. Lower-case symbols are deleted from the table. Subscripts are retained but are assigned "atomic numbers" from 120 to 129 (the value of the subscript + 120); this keeps subscripts in place for subsequent analysis-in-context and provides facile conversion from literal to numeric format. The allowed atom symbols comprise the whole periodic table plus deuterium and tritium. Other allowed symbols are parentheses [for constructing multiply occurring pendant groups or multiply occurring embedded connectives such as $(CH_2)_n$], colons (to represent bonds), charge symbols (+, −), and charge multipliers (+2, −3, etc.). Numerals occur only as subscripts or as charge multipliers. Radical centers are designated by a dot. Other symbols are not allowed and are flagged (and rejected) as invalid by the program. The graphics entry segment of the program also does context checking for symbol validity; for example, the lowercase "h" in Ch_3 would be rejected because the context for the use of "h" is wrong.

Parentheses Expansion. Parentheses may be used (1) to specify pendant groups as in $NH(CH_2CH_3)_2$ or $CH_3CH(CH_3)CH_2CH_3$ and (2) to specify repeated embedded connectives as in $Br(CH_2)_4CN$. Parentheses may be nested to any depth; the program correctly expands the structure $C(C(C(CH_3)_3)_3)_4$ and gives a connection table for its 53 carbon atoms. *Branched* parenthesized groups may also be used, as shown below. The



carboxymethyl side chain in this structure will be correctly replicated during expansion of the parenthesized group.

Parentheses are handled by copying the outermost parenthetical group n times (n = parenthesis subscript) at the end of the connection table.[3] If a side chain is attached to the parenthetical group, its attachment point is noted, and a separate connection table for the side chain is created, with atom numbering starting from unity. A copy of the side-chain connection table is appended to each of the replicated parenthetical groups in the connection table. Because the side-chain table is numbered from unity, it is very easily renumbered when it is appended to the replicates.

Nested parentheses are treated by a recursive routine that does parentheses expansion starting from the outside and working in. The program starts at the top of the connection table (atom 1) and scans for a left parenthesis. When it finds one, it looks for a matching right parenthesis. The whole group contained within parentheses is replicated n times at the end of the connection table. During this first expansion, replicated copies of any inner parentheses will be added to the end of the connection table. Scanning for matched pairs of left and right parentheses continues, and these replicated matched pairs will be reencountered and will in turn be replicated to the end of the connection table. This process of replication and expansion continues until all parentheses have been dealt with.

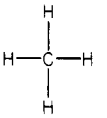
The replicate copies of the parenthesized group must be attached to the base molecule by entering appropriate To-Attachments in the body of the connection table. Analyses are made of the number of free bonds available at each terminus of the parenthetical group in order to determine whether the group is embedded or pendant and, if it is pendant, whether it is bonded from the left atom or from the right atom. For example, such analysis of CH_3CH_2 would classify this group as *pendant* (one free end for bonding). This analysis is based on the valence of the terminal atoms and context.

Alkyl Group Expansion. Saturated alkyl chains conforming to the empirical formula C_nH_{2n+1} are expanded by adding n carbon atoms to the connection table. The program searches the atom list for subscripted carbon and proceeds with the expansion if the search succeeds. The original set of symbols comprising "C_nH_{2n+1}" is deleted from the table, and its link to the molecule is transferred to the start of the newly added chain. Long alkyl chains can also be expressed by use of the polymethylene construct $(CH_2)_n$, which is particularly useful for long chains that are branched or unsaturated. Alkyl chain expansion is particularly useful because it allows standard contractions: C_2H_5 for ethyl, C_3H_7 for propyl, and so on.

The program does *not* expand perfluorocarbons, C_nF_{2n+2}, which would require addition of fluorine atoms to the connection table along with carbon.

Remove Hydrogen Atoms. Connection tables generally ignore hydrogen, although for our purposes, we carry the hydrogen count for each atom in a separate vector. The successful operation of the interpreter requires that symbol strings include explicit hydrogen atoms.[4] However, once the hydrogen count information is recorded, the hydrogen atoms and their subscripts are deleted from the connection table. Any "attachments" to hydrogen or to its subscript must be moved to a suitable center. For example, in CH_3CH_3, as shown, the right carbon is "bonded" to the preceding subscript.[3] When H_3 is removed, this "bond" must be moved to the left carbon atom; when this is done, a real single bond is created.

Hydrogen atoms that appear at the end of "stick" bonds are properly recognized. Methane could be entered in its compact form, CH_4, or as the stick structure



Distribute Multiple Charge. Multiple charge (e.g., expressions such as +2 and −3) is handled by adding $(n - 1)$ charges to the end of the connection table. Charge is entered as a pseudoatom with atomic number 132 (+) or 133 (−). A pointer in the connection table provides a link *from* the charge *to* the charged center, with bond type zero.

Distribute Subscripted Monovalent (Halogen) Atoms. Subscripted halogen atoms may appear as terminal groups, as in $CHCl_3$, or embedded in a chain, as in $CH_3CCl_2CH_3$. The procedure for handling these groups is similar to that used for multiple charge: n copies of the halogen atom are added to the end of the connection table, and the X_n group in the table is deleted. Any "bonds" to the subscripts n are transferred to the central atom, and To-Attachments are entered in the body of the table pointing from the central atom to the added halogen atoms.

Generate Implied Multiple Bonds. Symbol strings containing implied multiple bonds are at once the most common and the most difficult to handle. In the program, a series of three loops is used to locate and interpret such bonds. In the first loop, a search is made for all unsaturated carbon atoms that participate in at least one zero-type (unassigned) bond. The determination of unsaturation depends upon carbon valence (equal to 4), the number of attached hydrogen atoms, and context. Once an unsaturated carbon atom is spotted, the adjacent right neighbor is examined to determine whether it can support formation of a double bond: sulfur, nitrogen, and oxygen are the only neighbors considered. The number of unsatisfied valences is determined for the unsaturated carbon atom and for its neighbor, and the smaller of the two valences is used to determine the multiplicity of the bond.

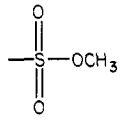
If a suitable adjacent right neighbor is not present for multiple bond formation, the adjacent left neighbor is examined for possible multiple bonding to the left. If a suitable left neighbor is not found, the program will adjust the hydrogen count at the unsaturated atom and signal an error.

After a carbon–heteroatom pair is marked as eligible for multiple bond formation either to the left or to the right, the connection table is adjusted to establish a link between the central carbon atom and any atoms lying beyond the multiply bonded heteroatom; for example, in COOH a link will be created between carbon and OH. Strings with subscripted oxygen, as in $-CO_2CH_3$, are transformed by replacing the subscript with an oxygen atom so that $-CO_2CH_3$ and $-COOCH_3$ can be treated with the same code. (The program is written generally, so that a group such as $-CS_2CH_3$ would be handled the same way.) This loop of the program allows interpretation of esters, amides, ketones, acid halides, etc., i.e., groups in the general class –COX, or XOC–. Aldehydes and formates, with hydrogen at the carbonyl carbon, are treated as special cases.

The search for implied multiple bonds continues in a second loop, which seeks out elements after the first row of the periodic table as centers for multiple bond formation (halogens are excluded). Sulfur and phosphorus would be the most important representatives of this category. Candidates for attachment to the central atom are restricted to O, N, and S;

oxygen, of course, is ubiquitous. The program looks first to the right of the central atom, and if it cannot find a suitable atom for multiple bond formation in that location, it looks to the left. The adjacent atom (usually oxygen) is examined to ascertain that it is available for multiple bond formation. If the oxygen atom is subscripted, the value of the subscript is noted, and the subscript is replaced by an oxygen atom. Additional atoms required by a subscript larger than 2 are added to the end of the connection table, with appropriate entries to establish links to the central atom. Finally, groups beyond the oxygen atom are linked to the central atom.

Oxygenated sulfur derivatives offer the most frequent problems in ambiguity of interpretation. Part of the problem arises from the various valencies sulfur can show, which allow it to have from one to four nonhydrogen substituents. The rules built into the program are the following. (1) Priority is given to formation of double bonds between S and O, but no more than two may be formed. (2) All remaining substituents, including oxygen, must be singly bonded to S. These rules are used generally; they will apply to phosphorus as well. Under these rules, the string $-SO_3CH_3$ will be expanded to represent a methyl sulfonate ester



Applied to the lone connective $-SO_2-$, the program generates the conventional sulfonyl group, with two oxygen atoms doubly bonded to S.

The last interpretive loop for finding implied multiple bonds searches for unsaturated first-row elements other than carbon. The principal need for this segment is to interpret nitro and nitroso substituents. This loop also completes the interpretation of certain composite function groups such as isocyanate, -NCO. For -NCO, the first interpretive loop will create C=O, and the last interpretive loop will create N=C, to give -N=C=O for the whole group.

The sequence in which these three interpretive loops are applied is quite important. For example, double-bonded sulfur (C=S) is interpreted in the first loop, and higher oxidation states of sulfur are considered in the second loop. Since the first loop removes all cases of divalent unsaturated sulfur, it is safe to assume that all cases of unsaturated sulfur in the second loop must involve higher oxidation states. In this way, we can deal with the problem of the multivalence of sulfur without the need to determine that valence explicitly in a given case. Sequencing of the interpretive loops also facilitates interpretation of composite groups such as -SCN (thiocyanate), -NCO (isocyanate), and -NCS (isothiocyanate). Priority of assignment of bonds is as follows: (1) carbon doubly bonded to the right, (2) carbon doubly bonded to the left, (3) sulfur (P) multiply bonded to the right, (4) sulfur (P) multiply bonded to the left, (5) first-row elements (not C) multiply bonded to the right, (6) first-row elements (not C) multiply bonded to the left. Under these priorities, the group -OCO- will be interpreted as -O—C=O, and the sulfur atom in $-O_2SO-$ will be doubly bonded to *one* oxygen atom, not to two.

Subscripted oxygen atoms occurring in unsaturated groups are taken care of by these routines. However, subscripted oxygen atoms in *saturated* groups (the peroxy, $-O_2-$, group) will not be properly interpreted. Therefore, a clean-up routine is used that looks for subscripted, nonnitrogen atoms. The program signals an error for subscripts larger than 2. Subscripted nitrogen atoms are handled as separate cases because they involve unusual charge-separated structures.

Interpret Subscripted Nitrogen. The program searches for nitrogen followed by a subscript 2 or 3. If the subscript is 2, a decision must be made as to whether the N_2 group is present as diazonium or as diazo; the distinction is made on the basis of the presence of unsaturation at the attached carbon atom; for a diazonium group, the attached carbon will not have open bond sites. The structures assigned by the program to these two cases are $C{=}N^+{=}N^-$ (diazo) and $C{-}N^+{\equiv}N$ (diazonium). If the positive charge is erroneously omitted from the diazonium nitrogen during structure entry, the program will construe this as an error and will put in the correct charge. These groups are expanded by substituting a nitrogen atom for the subscript.

In the case of N_3, the subscript is replaced by a nitrogen atom, and one nitrogen atom is added to the end of the connection table. The arrangement of bonds and charge in N_3 corresponds to the structure $-N{=}N^+{=}N^-$.

Interpret Colon Bonds. The colon is used to specify bond type inside a symbol string. A vinyl group is keyed in as $CH{::}CH_2$ (left bonded) or $CH_2{::}CH$ (right bonded). This symbolism was chosen because it permits logical extension to representations of a triple bond as three successive colons. The "obvious" choices (hyphen for a single bond, equal sign for a double bond) force an arbitrary choice of some unobvious symbol for a triple bond. The use of colons to indicate bonds also has a long tradition. The colon bond is useful for "forcing" a particular interpretation of a string. The string $C{:}OCH_3$ forces the interpretation of the oxygen atom as an element in a methoxy group and prohibits the use of the oxygen atom as carbonyl oxygen. Therefore, although OCO will be interpreted as O—C=O according to the priorities discussed earlier, OC:O will be interpreted as O=C—O.

CONCLUSION

The interpreter discussed here offers an extension of the ways in which chemical structure may be entered into a computer. Most structure entry programs depend upon construction of functional groups from a fixed menu of components; for example, a carbomethoxy might be built up from a carbonyl group and a methoxy group taken from the menu. The fixed menu offers the advantage that the vocabulary used in the structure entry program is under the complete control of the programmer, which decreases the probability that the user will produce structures that fall outside the capabilities of the program; that is, the program enables the construction of only those patterns that it can interpret. This limitation makes menus ideal for use by novices or by people who have little chemical training. On the other hand, menu selection of functional groups can be clumsy, for large menus are required to handle a wide variety of cases, and the construction of large groups is by no means facile.

The use of the string interpreter with keyboard input introduces great facility and flexibility, but the trade-off is an increased probability of producing an unintended structure. Provided the chemist does not do anything "wild", the program will yield a sensible result. In fact, the first requirement for input is formulation of a sensible, unambiguous structure. Of course, valence checks and screening for ambiguities help the user to formulate a correct structure, but certainly, not *all* possibilities for ambiguity have been foreseen in the writing of the program. On the other hand, since the program does reflect the practices of the organic chemist in writing structure, the average chemist can learn the system very rapidly and finds it very "natural", which was one of our design aims.

APPENDIX: SUMMARY OF RULES FOR STRING FORMULATION

(1) A string is a sequence of up to 80 allowed symbols (Figure 1).

(2) Subscripted and nonsubscripted monovalent atoms follow immediately their attached atoms unless the monovalent

atoms initiate the string (e.g., $H_2NCH_2CF_2$).

(3) Subscripted or nonsubscripted monovalent atoms of different types may occur jointly to the right of their attached, central atom, but only one such entity may occur to the left of the central atom and must initiate the string. It is recommended that hydrogen (with or without a subscript) always be placed immediately adjacent to the central atom.

(4) Unsaturation within a chain is denoted by pairs or triples of colons (e.g., $CH_2::CH_2$). (a) Colons cannot start or end a string. (b) A single colon forces the interpretation of a bond as a single bond and overrides any implied interpretations.

(5) Hyphens may be used only at the start or the end of a string to force the attachment point. Default attachment (attachment at the left end of the string is given preference in ambiguous cases) is overrriden by the hyphen.

(6) Multiply bonded heteroatoms X must be written immediately adjacent to the central atom C and may follow or precede that atom (CX and XC). Any group lying beyond X (group Y in CXY or YXC) is presumed to be attached to central atom C. Central atom C may be any polyvalent atom with valence greater than 2.

(7) Multiply bonded heteroatoms X may appear with subscripts before or after the central atom C (X_2C or CX_2, as in O_2S or SO_2). The assignment of bonds to the CX pairs is determined by the valence of central atom C. For first-row elements, the maximum number of CX double bonds is one (a CO_2 group is interpreted as O—C═O). For central atom C beyond the first row, the maximum number of double bonds is taken as two. For structures such as CX_nY or YX_nC, the group Y is presumed to be attached to the central atom C (examples are ClO_2S and SO_2Cl).

(8) For strings containing the substring XCY where X and Y are heteroatoms and C is a central atom, priority of bond assignment is as follows: (1) carbon doubly bonded to the right, (2) carbon doubly bonded to the left, (3) sulfur or phosphorus multiply bonded to the right, (4) sulfur or phosphorus multiply bonded to the left, (5) noncarbon first-row elements multiply bonded to the right, (6) noncarbon first-row elements multiply bonded to the left. As each multiple bond is assigned in accord with this table of priorities, the residual valence of the central atom is decreased. The priority table therefore determines directly the distribution of multiple bonds in a complex string. Example: SCO is S—C═O; OCS is O—C═S.

(9) Parentheses may be used in two ways: (1) to specify pendant groups, which may be replicated by means of a subscript attached to the right parenthesis; (2) to specify the occurrence of a repeated, embedded chain. Examples are $(CH_3)_3C$, $(CH_2)_3$, and $CH(SH)CH_2$. Parentheses may be nested to any depth. Side chains may be attached to strings appearing inside parentheses. Parenthesized strings must begin and end with an atom symbol, a subscript, or a charge; terminating bonds (hyphens or colons) are not allowed.

ACKNOWLEDGMENT

I am indebted to Craig Shelly, who assisted materially in the debugging of the interpreter and suggested a number of improvements.

REFERENCES AND NOTES

(1) Barker, P. G. "Syntactic Definition and Parsing of Molecular Formulae. 1. Initial Syntax Definition and Parser Implementation". *Comput. J.* **1975**, *18*, 335–359.

(2) For references see: Howe, W. J.; Hagadone, T. R. "Molecular Substructure Searching: Computer Graphics and Query Entry Methodology". *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 8–15.

(3) Note that the interpreter operates directly on the structure table to form the connection table. Hereafter, I will refer to this table as a "connection table", although it is not such at the beginning of the process.

(4) Ring building and point-to-point carbon chain formation, in which a VERTEX defines the location of a carbon atom and all bonds are shown explicitly, do not require that hydrogen atoms also be explicit. In these cases, it is assumed that bonds remaining after counting explicit ones are used for hydrogen atoms. For symbol strings, the bond distribution at each atom is unspecified (bond order zero), and indeed, the principal task of the interpreter is to discover what the bond distribution should be. The hydrogen count is an essential piece of information for this discovery.