The original formula PAH periodic table (Table III) becomes expanded in the left-hand region when smaller ring sizes are permitted and in the upper right-hand region when larger ring sizes are allowed; there is a tendency for fewer internal third degree carbons (N_{Ic}) in the former case and for more internal third degree carbon atoms in the latter case. When $c = 0$ and $q_b = 0$, the following have been postulated: (a) Table III is applicable when $2r_4 + r_5 - r_7 - 2r_8 = 0$; (b) Table IV is applicable when $2r_4 + r_5 - r_7 - 2r_8 = 1$; (c) Table V is applicable when $2r_4 + r_5 - r_7 - 2r_8 = -1$; (d) Table VI is applicable when $2r_4 + r_5 - r_7 - 2r_8 = 2$; (e) Table VII is applicable when $2r_4 + r_5 - r_7 - 2r_8 = -2$.

The formula subscript numbers in Tables III–VII are simply subsets of the universal even pair set ($N \times N$). The universal even pair set $N \times N$ is an *equivalence relation in* N, where N is the set of all even, positive, nonzero integers (i.e., natural numbers). Thus $N \times N = \{\langle a,b \rangle:\ a, b \epsilon N,\ a > 0,\ b > 0\}$ and

$$\{\text{Table VI}\} = R_4{}^* = \{\langle a,b \rangle:\ aR_4b\}$$

$$\{\text{Table IV}\} = R_5{}^* = \{\langle a,b \rangle:\ aR_5b\}$$

$$\{\text{Table III}\} = R_6{}^* = \{\langle a,b \rangle:\ aR_6b\}$$

$$\{\text{Table V}\} = R_7{}^* = \{\langle a,b \rangle:\ aR_7b\}$$

$$\{\text{Table VII}\} = R_8{}^* = \{\langle a,b \rangle:\ aR_8b\}$$

$$R_4{}^* \subset N \times N,\ R_5{}^* \subset N \times N,\ R_6{}^* \subset N \times N,\ R_7{}^* \subset N \times N,\ R_8{}^* \subset N \times N$$

$$R_5{}^* \subset R_4{}^*,\ R_6{}^* \subset \{R_5{}^*,\ N_c = 2N_H + 6 \text{ row series}\} \subset \{R_4{}^*,\ N_c = 2N_H + 6 \text{ row series}\}$$

$$\{\text{Table VIII}\} = R_p{}^* \subset R_8{}^*$$

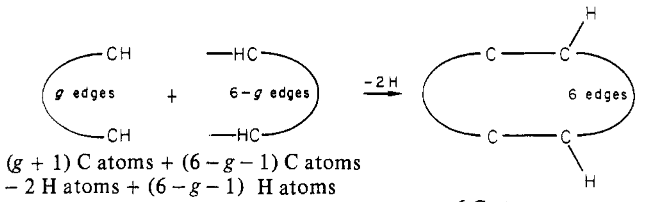
APPENDIX. INDUCTIVE PROOF OF EQUATION 2

Theorem: For exclusively fused polycyclic aromatic hydrocarbons composed of only hexagonal rings; $N_{Pc} = N_H - 6$.

Proof: The proof is by induction. This proposition is true for $r = 1$ and 2; i.e., for benzene ($0 = 6 - 6$) and naphthalene ($2 = 8 - 6$) let the graphs be recursively built up from the $r = 1$ case by successive fusion of additional hexagonal rings.

It will be shown that at each step N_{Pc} and N_H are incremented by the same amount. In any single step, a new hexagon will be formed by taking g existing adjacent peripheral edges ($g = 1$–5) of which the two extreme carbon vertices contain hydrogen atoms (the remaining $g - 1$ carbon vertices are peripheral third degree carbon atoms without adjacent hydrogen atoms) and adding $6 - g$ edges, i.e., $5 - g$ additional carbon vertices each possessing a hydrogen.

Since two hydrogens are lost, N_H increases by $3 - g$. Also, $g - 1$ peripheral carbon vertices become internal carbon vertices, and two former hydrogen-bearing carbons become peripheral ones, and thus N_{Pc} also increases by $2 - (g - 1) = 3 - g$.



$(g + 1)$ C atoms + $(6 - g - 1)$ C atoms
$- 2$ H atoms + $(6 - g - 1)$ H atoms

6 C atoms
$(3 - g)$ H atoms

This completes the inductive proof of eq 2.

REFERENCES AND NOTES

(1) Part 1: Dias, J. R. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 15.
(2) Balaban, A. T.; Harary, F. *Tetrahedron* **1966**, *24*, 1097.
(3) Harary, F.; Palmer, E. "Graphical Enumeration"; Academic Press: New York, 1973. Balaban, A. T. "Chemical Applications of Graph Theory"; Academic Press: New York, 1976.
(4) Balaban, A. T. *Rev. Roum. Chim.* **1981**, *26*, 407.
(5) Randić, M. *J. Am. Chem. Soc.* **1977**, *99*, 444.
(6) Herndon, W. *Tetrahedron* **1973**, *29*, 3.
(7) Randić, *Chem. Phys. Lett.* **1976**, *38*, 68.
(8) Randić, M. *Tetrahedron* **1977**, *33*, 1905.

WISENOM. A Formal Organic Chemical Nomenclature System

E. V. KRISHNAMURTHY

Department of Applied Mathematics, Indian Institute of Science, Bangalore 560012, India

Received April 28, 1981

A formal chemical nomenclature system WISENOM based on a context-free grammar and graph coding is described. The system is unique, unambiguous, easily pronounceable, encodable, and decodable for organic compounds. Being a formal system, every name is provable as a theorem or derivable as a terminal sentence by using the basic axioms and rewrite rules. The syntax in Backus–Naur form, examples of name derivations, and the corresponding derivation trees are provided. Encoding procedures to convert connectivity tables to WISENOM, parsing, and decoding are described.

I. INTRODUCTION

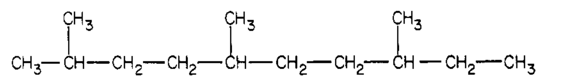
There are many systems for naming organic compounds.[1,2] Although there are several advantages in these systems, e.g., human comprehension, mechanical encoding and decoding, these names are difficult, and in some cases impossible, due to the lack of a formal grammar with a well-defined syntax.

Many names cannot be decoded into a structural formula, especially trivial names, and others that are highly context dependent require intuition and human skill. For example, consider

(i) 2,5,8-trioxadecane

$$CH_3\text{-}O\text{-}CH_2\text{-}CH_2\text{-}O\text{-}CH_2\text{-}CH_2\text{-}O\text{-}CH_2\text{-}CH_3$$

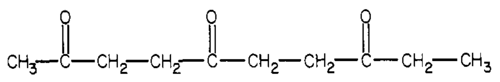
and (ii) 2,5,8-trimethyldecane



In (i), the oxygen atoms are along the main chain of 10 atoms, and in (ii), methyl groups are not in the main chain but are substituents.

These two examples illustrate that names with similar phrase structures need not necessarily describe structures of similar forms. This is not desirable since unambiguous mechanical decoding is then not possible.

For instance, on the basis of (ii), one may equally well decode (i) as



which stands for 2,5,8-decanetrione.

Clearly, these difficulties arise from the fact that existing nomenclature systems are not based on context-free grammars. Therefore, for unambiguous decoding of the names in these nomenclature systems, it is necessary to know other contextual information, for instance, the naming procedure, whether it is substitutive or conjunctive, or other related chemical information, such as valency and bonds. It is clear that contextual information cannot be deduced by mechanical means, unless explicitly incorporated in the name. Therefore in order that a nomenclature system be mechanically effective, in the sense of having an algorithm, it has to be based on a context-free formal grammar which besides being a formal language contains all the topological and chemical features of a given structure. In this paper we describe such a system with the following features:

(a) Easy human comprehension and pronounceability.

(b) Easy mechanical/human encoding and decoding/visual display.

(c) Unique, unambiguous, and context-free-grammar-based nomenclature system which is as close as possible to the existing systems.

(d) The topological description, locant labeling, and chemical description are based on WLN[3] and ALWIN,[4,5] except for some modifications, namely, the numerals describing ring or chain sizes and multiple prefixes are replaced by their linguistic equivalents as suggested by Goodson[2] and the atoms, groups, bonds, etc., are replaced by their shortened chemical names, e.g., hydroxyl for OH, carbonyl for CO, and so on.

As the basic approach and data structure are derived from ALWIN, this system is named WISENOM (abbreviation for WISWESSER NOMENCLATURE).

II. SYNTAX OF WISENOM

The definition of any formal language consists of several parts, namely

(1) basic symbols, e.g., the alphabet

(2) syntax, the rules which define how the basic symbols can be concatenated to larger syntactic units to complete a sentence—which in our case is a chemical name

(3) semantics (or semantic rules) which define the meaning of a given sentence

In organic nomenclature we have to be concerned with both the syntactic and semantic aspects—the former for checking the validity of the form and the latter for the implicit topological and chemical meaning.

The elementary nomenclature objects are the atoms (⟨ATOM⟩), radicals (⟨RADICAL⟩), functional groups (⟨FUNGROUP⟩), bonds (⟨BOND⟩), trivial chemical names (⟨TRIVIALNAME⟩), the locant labels (⟨LOC⟩), and topological properties such as trees or rings, their sizes, multiplicity, and how these are assembled. Accordingly, our alphabet consists of all the names of elementary chemical objects, English alphabet, digits, and punctuation symbols. The syntax of WISENOM is a set of rules which determine which string of symbols can be accepted as a well-formed or valid name of an organic chemical. This set of rules is given in Table I with explanatory notes.

To give a simple example of how this works, consider the definition

$$\langle \text{NAME} \rangle ::= \langle \text{PREFIX} \rangle \langle \text{PARENT} \rangle \langle \text{SUFFIX} \rangle^{+} | \langle \text{ATOM} \rangle$$

The angular brackets ⟨ ⟩ represent the nonterminals or metalinguistic variables—the quantities being defined as a set of possible strings of symbols; the ::=means is defined to be; | denotes the exclusive or; a superscript + denotes the sequence of at least one of the valid strings contained within ⟨ ⟩. For example, in rule 1 of Table I $\langle\text{RINGSIZE}\rangle^{+}$ means the set of items defined in ⟨RINGSIZE⟩ can recur with any length.

We use the convention that capital English letters enclosed in ⟨ ⟩ denote the set of nonterminal strings. Terminals are those enclosed in () or lower case English letters and numerals. The way of writing the syntactic definitions as above is called Backus–Naur form (BNF for short) in honor of J. W. Backus and P. Naur,[6] who suggested a notation for writing grammars that specify the syntax of programming languages.

The definitions such as

$$\langle \text{NAME} \rangle ::= \langle \text{PREFIX} \rangle \langle \text{PARENT} \rangle \langle \text{SUFFIX} \rangle$$

$$\langle \text{SUFFIX} \rangle ::= (\langle \text{LINK} \rangle \langle \text{BOND} \rangle (\langle \text{NAME} \rangle))^{+}$$

are called recursive definitions. In other words, these say that a ⟨NAME⟩ contains a ⟨SUFFIX⟩ which again contains a ⟨NAME⟩. Also ⟨LOC⟩ = a|b|c. . . where ⟨ALPH⟩ = {a,b,c,. . .,w} means the locants or nodes of the graph are labeled with terminal words such as a, b, ab,. . . which are sequences drawn from ⟨ALPH⟩ denoted by $\langle\text{ALPH}\rangle^{+}$; for counting the locant, these sequences are assigned weights in the positional base 23 (rule 2.111). In order to differentiate this system from the existing systems, we prefer to use letters for locants for both acyclic and cyclic structures. Note that both ALWIN/WLN use locant numbering only for cyclic structures.

III. ENCODING, SYNTACTIC REPRESENTATION, AND DECODING

An important preliminary step before encoding an organic chemical structure into a name or a nonconventional code (ALWIN or WLN) is the unique labeling of the atoms (or nodes) in the associated chemical graph, using certain invariant procedures and hierarchical rules that are free from subjectivity. The importance of using such procedures is easily understood, as different labeling procedures would otherwise lead to different names and codes.

In this paper, we assume that the Morgan labeling procedure and hierarchical rules described earlier by Krishnan and Krishnamurthy[5] are used as preliminary steps to arrive at a unique Morgan label for each node in the given structure. Also we refer to the same paper[5] for relevant graph-theoretic definitions, techniques, and algorithms on which this paper is dependent.

(a) Encoding Acyclic Structures. To encode an acyclic or a tree structure, we convert it into its equivalent parentheses form in which a main chain of (alphabetically successive) labeled nodes is identified and the branch structures are parenthesized, following each node label at which the branches occur. For this purpose, we use the following procedure.

Step 1 (*Starting Node*). Given a set of nodes constituting a tree, pick up as the starting node that node having the highest Morgan label and least degree.

Step 2 (*Traveling Rule*). From the starting node form a path sequence (through other nodes) by moving from each node to the neighboring lowest Morgan labeled node; continue until the terminal node is reached.

Step 3 (*Main and Branch Chain*). At each node record the branches, if any, within parentheses in left to right order, with the branch paths having their starting points ordered in the

Table I. WISENOM Syntax[a]

rule no.	function	definition
0	⟨NAME⟩	::= ⟨PREFIX⟩⟨PARENT⟩⟨SUFFIX⟩$^+$ | ⟨ATOM⟩ | ⟨RADICAL⟩ | ⟨FUNGROUP⟩ | ⟨BOND⟩ | ⟨ALKANE⟩ | ⟨TRIVIALNAME⟩. Recursive definition of a name through parent name (⟨PREFIX⟩ and ⟨PARENT⟩) and names of attachments (⟨SUFFIX⟩).
1	⟨PREFIX⟩	::= (he | ca)(tree | ring)(⟨NULL⟩ | 1 | 2)(⟨RINGSIZE⟩$^+$ | ⟨CHAIN⟩) | ⟨NULL⟩. Describes the nature (ca for carbo, he for hetero) of the atoms in a parent ring or a main chain (for tree or acyclic structures), the type of the ring system[5] (type 1 when the system has bridges; type 2 when the system has pseudobridges, rings with crossed bonds, and ring of rings), and ring or chain size, viz., the number of atoms in a ring or a chain.
1.1	⟨NULL⟩	::= empty string of symbols indicating absence of any specified phrase.
1.2	⟨RINGSIZE⟩	::= ⟨MULT⟩(tri | tetr | pent | hex | hept | . . .dec | . . .cent). Describes the multiplicity as well as virtual nature of a ring of specified size and the ring size.
1.21	⟨MULT⟩	::= ⟨NULL⟩ | (bini | terni | quarterni | quini | . . .centi | . . .quasi-). Describes multiplicity as well as virtual (quasi) nature of a ring of a specified size.
1.3	⟨CHAIN⟩	::= (meth | eth | prop | but | pent | hex | hept | . . .dec | . . .cent | . . .). Naming a carbon or hetero chain depending upon the number of atoms in it.
2	⟨PARENT⟩	::= ⟨TOPOLOGY⟩⟨RINGATOM⟩⟨SAT⟩= | ⟨NULL⟩= | ⟨CHAINATOM⟩⟨SAT⟩= | -cyclo⟨CHAIN⟩ane= | ⟨TRIVIALNAME⟩= | -ane=. Describes the topology of the parent ring system or the main chain, name of ring atoms or chain atoms, groups, saturation, and their locant labels. This is followed by punctuation =, indicating that the parent ring system in a cyclic structure or a main chain in an acyclic structure is described completely.
2.1	⟨TOPOLOGY⟩	::= (⟨TESS⟩⟨KNIT⟩⟨MKNIT⟩)$^+$ | ⟨NULL⟩. In the case of acyclic structures and monorings, this phrase is absent, since what we need is only their adjacency; this is, respectively, described by ⟨CHAINATOM⟩ or ⟨RINGSIZE⟩. In the case of general cyclic structures this phrase plays a very fundamental role. This consists of tessellation of rings in which two rings share a common edge and knit and multiknit of rings which specify interconnections. These can occur and recur in any order.
2.11	⟨TESS⟩	::= (/LOC)$^+$ | ⟨NULL⟩. The process of fusing two rings over a common edge in which the locants or node labels are alphabetically successive. Described by / followed by the lexicographically earlier locant.
2.111	⟨LOC⟩	::= a | b | c | . . . | aa | . . . | ww | . . . words belonging to ⟨ALPH⟩ = {a,b,c. . .,w}. Represented as positional numbers in base 23 with symbols {a,b,c. . .w}, the 23 lower case English letters. e.g., aa = 24; w = 23; ba = 47.
2.12	⟨KNIT⟩	::= (⟨:LOC⟩(⟨DIGIT⟩$^+$ | -)⟨LOC⟩)$^+$ | ⟨NULL⟩. The process of connecting any two labeled nodes or the same node through a known number of nodes (locants) to complete a real ring and some virtual or quasi-rings. Described by a colon followed by a triplet, viz., the initial locant, number of (nonzero) intermediate nodes, and the final locant or the initial locant, a hyphen indicating direct connection and the final locant.
2.121	⟨DIGIT⟩	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ⟨NULL⟩. A sequence of decimal digits to indicate the number of intermediate nodes in a knit or multiknit.
2.13	⟨MKNIT⟩	::= (;(⟨LOC⟩(⟨DIGIT⟩$^+$ | -)⟨LOC,⟩)$^+$)$^+$ | ⟨NULL⟩. The process of knitting several segments to complete a real and some quasi-rings. Described by a semicolon followed by a quadruple, viz., the initial locant, number of (nonzero) intermediate locants, final locant, and a comma. Unlike ⟨KNIT⟩, this sequence of quadruples recur following a semicolon, in order to complete a ring and some quasi-rings.
2.2	⟨RINGATOM⟩	::= ⟨NULL⟩ | (⟨LINK⟩(⟨ATOM⟩ | ⟨RADICAL⟩ | ⟨FUNGROUP⟩))$^+$
2.3	⟨CHAINATOM⟩	= ⟨RINGATOM⟩. Describes the locant and the nature of the atom (name, spiro, sat) or radical or group in a ring or the main chain.
2.4	⟨SAT⟩	::= (⟨SATA⟩ | ⟨SATU⟩)$^+$ | -ane. Describes saturation/unsaturation in a ring system by defining three mutually exclusive phrases: ⟨SATA⟩, ⟨SATU⟩, -ane.
2.41	⟨SATA⟩	::= ⟨LINK⟩⟨BOND⟩ | ⟨NULL⟩. Describes double–triple bond between alphabetically successive neighboring locants by specifying the earlier locant and the bond nature.
2.42	⟨SATU⟩	::= -⟨LOC⟩⟨BOND⟩⟨LOC⟩- | ⟨NULL⟩. Describes double–triple bond between neighboring locants which are not alphabetically successive by specifying the earlier locant label, bond nature, and the later locant label.
2.43	-ane	Indicates saturation.
3	⟨SUFFIX⟩	::= (⟨LINK⟩(⟨BOND⟩ | ⟨NULL⟩)(⟨NAME⟩))$^+$ | (⟨LINK⟩(⟨BOND⟩ | ⟨NULL⟩)(⟨LINK⟩⟨NAME⟩))$^+$ | (⟨LINK⟩(⟨BOND⟩ | ⟨NULL⟩)(⟨ATOM⟩ | ⟨FUNGROUP⟩ | ⟨RADICAL⟩))$^+$ | ⟨NULL⟩. Preceded by the delimiter =. This describes the attachments to the parent ring system or the main chain by explicitly citing the parent locant and the bond through which a substituent is attached. If the substituent is cyclic, its point of attachment is also cited; however, for acyclic substituents the point of attachment is treated as the first locant. Since a substituent may have one or more further substituents, these substituents are cited within () by their ⟨NAME⟩ to recursively compute their terminal names, if they are not already terminals.
3.1	⟨LINK⟩	::= -⟨LOC-⟩ | -⟨LOC,LOC-bini⟩ | -⟨LOC,LOC,LOC-terni⟩ | . . . | | -⟨LOC,LOC,. . .LOC-nini⟩. Describes in between hyphens a set of locants separated by commas and followed by the number of repetitions or multiplicity of occurrence of a specified atom, bond, group (bini, terni, quarterni, etc.).
4	⟨ATOM⟩	::= (terminals) noncarbon atomic names with known valency, saturated (sat), and spiro (spiro) atoms.
5	⟨RADICAL⟩	::= (terminals) radicals with known valency.
6	⟨FUNGROUP⟩	::= (terminals) functional groups with known valency.
7	⟨BOND⟩	::= (terminals) ene = double bond, yne = triple bond, ane = single bond.
8	⟨ALKANE⟩	::= (terminals) alkanes.
9	⟨TRIVIALNAME⟩	::= (terminals) such as benzene, toluene, with known structures and prelabeled locants.

[a] (1) Terminal/nonterminal symbols: The lower case English alphabet, a–z, digits 0–9, and punctuation symbols = / : ; , ( ) - ƀ are the terminal symbols; the rest are nonterminal symbols. (2) Alphameric hierarchy: Lexicographically earlier = ; -: / ( ) , - 0 1 . . . 9 ƀ a b . . . z (latest). (3) Functions, definitions, and their purpose: The nonterminals (functions) are enclosed in angular brackets ⟨ ⟩ and are written in upper case English alphabet. (4) Counting terminology for multiplicity and size. (a) Multiplicity. Units: uni, bini, terni, quarterni, quini, seni, septeni, octoni, noveni. Ten: geni. Hundred: centeni. Example: 223 = bini centeni, bini geni, terni; 12 = geni, bini. (b) Size. Units: mono, di, tri, tetra, penta, hexa, hepta, octa, nona. Ten: deca. Hundred: centa. Example: 223 = di centa, di deca, tri; 13 = deca, tri.

decreasing order of their respective Morgan labels.

Repeat steps 2 and 3 until all the paths are exhausted or, in other words, the tree is fully explored. The main chain then consists of the set of unparenthesized nodes.

Step 4 (*Labeling of Locants*). The main chain locants are then labeled from left to right with labels a_0, b_0, c_0, etc.; also a set of locants belonging to the *i*th level of parentheses (i = 1,2,3,. . .) and forming a continuous chain are given the labels

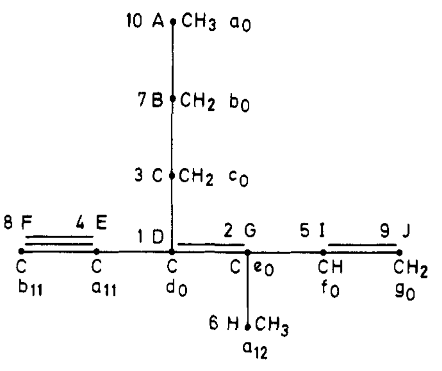


Figure 1. Morgan labeling.

Table II. Connection Table

node	label	atom/ group	connections	bond
A	a_0	CH_3	a_0–b_0	single
B	b_0	CH_2	b_0–c_0	single
C	c_0	CH_2	c_0–d_0	single
D	d_0	C	d_0–a_{11}	single
E	a_{11}	C	a_{11}–b_{11}	triple
F	b_{11}	CH	d_0–e_0	double
G	e_0	C	e_0–a_{12}	single
H	a_{12}	CH_3	e_0–f_0	single
I	f_0	CH	f_0–g_0	double
J	g_0	CH_2		

a_{i1}, b_{i1}, c_{i1}, etc.; if there are several disjoint main chains (say j) at the ith level, then the adjacent locants on these chains are given the labels a_{ij}, b_{ij}, c_{ij}, . . . where i denotes the level and j is the chain index. The attachment locants are then remembered for each branch, and these are put in the syntax form required for WISENOM. The numeral suffixes in a, b, c, etc., are then removed to obtain the formal name. Note that although the numeral suffixes are finally dropped, these can again be reconstructed from the parentheses levels. This is in fact the first step for mechanical decoding, (see section IIIb).

Example 1: Consider the structure in Figure 1. We initially dummy label the nodes of the graph by capital letters. The final assignment of Morgan labels to the nodes using Krishnan and Krishnamurthy algorithm[5] is shown in Figure 1, as numerals near the dummy labels.

Step 1: The starting node is A, since among the least degree nodes, it has the highest Morgan label.

Step 2: The path taken to obtain the main chain is

ABCDGIJ

since among the paths leading to E or G from node D, G has the lower Morgan label; also at G we move to I rather than H since I has a lower Morgan label.

Step 3: The tree with main chain and branches is now represented by

ABCD(EF)G(H)IJ

Step 4: The reorganized connection table is given in Table II; the new labels are shown in Figure 1. In order to arrive at ⟨NAME⟩ in WISENOM, we use Table I.

Consider the phrase ⟨NAME⟩; this consists of three parts:

⟨PREFIX⟩⟨PARENT⟩$\langle$SUFFIX$\rangle^+$

The ⟨PREFIX⟩ phrase describes the nature of the graph (hetero or carbo or tree or ring) and the ⟨RINGSIZE⟩ or ⟨CHAIN⟩. In this example we have a carbo tree (ca tree) and ⟨CHAIN⟩ is hept (rule 1.3). The phrase ⟨PARENT⟩ describes the topology. In a tree, we describe this by separately indicating the main chain ⟨CHAINATOM⟩ (rule 2.3) and in this indicate only the noncarbon atoms or groups, if any, and their locants. The branches, if any, are indicated in $\langle$SUFFIX$\rangle^+$ (rule 3) which describes the locants of a main chain ⟨LINK⟩ to which these are attached (rule 3.1), ⟨BOND⟩, the nature of bonding, the name of the branch,

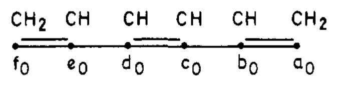


Figure 2. ca tree hex-a,c,e-terni ene =.

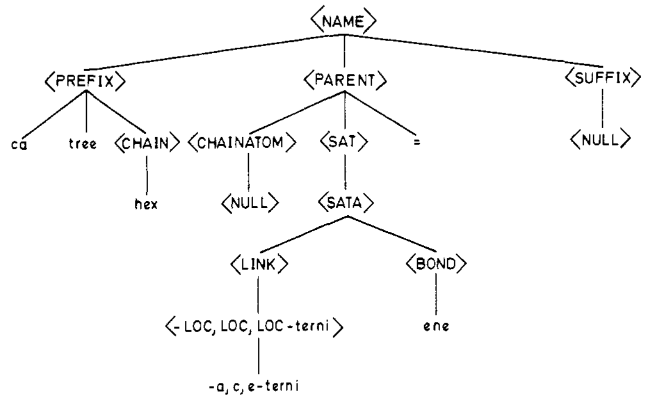


Figure 3. Derivation tree.

⟨NAME⟩, along with the linking node, if cyclic. If the branch is acyclic the ⟨NAME⟩ is only indicated. This ⟨NAME⟩ is a recursive definition, since a branch structure may also have branches. This recursion is continued until the ⟨NAME⟩ is resolved into a set of well-defined terminal names.

Thus, in the above case we have

⟨NAME⟩	::= ⟨PREFIX⟩⟨PARENT⟩$\langle$SUFFIX$\rangle^+$
⟨PREFIX⟩	::= ca tree hept
⟨PARENT⟩	::= ⟨CHAINATOM⟩⟨SAT⟩= (rule 2)
	::= ⟨NULL⟩$\langle$SATA$\rangle^+$= (rule 2.4)
	::= ⟨LINK⟩⟨BOND⟩= (rules 2.41 and 3.1)
	::= -⟨LOC,LOC-bini⟩⟨BOND⟩=
	::= - d_0,f_0- bini ene=
$\langle$SUFFIX$\rangle^+$	::= ⟨SUFFIX⟩⟨SUFFIX⟩ (rule 3)
	::= ⟨LINK⟩⟨NULL⟩(⟨NAME⟩)-⟨LINK⟩⟨NULL⟩⟨RADICAL⟩ (rule 3.1)
	::= -⟨LOC-⟩(⟨NAME⟩)-⟨LOC-⟩⟨RADICAL⟩
	::= -d_0-(⟨NAME⟩) - e_0-methyl
(⟨NAME⟩)	::= (⟨PREFIX⟩⟨PARENT⟩⟨SUFFIX⟩)
⟨PREFIX⟩	::= ca tree eth
⟨PARENT⟩	::= ⟨CHAINATOM⟩⟨SAT⟩=
	::= ⟨LINK⟩⟨BOND⟩=
	::= -⟨LOC-⟩ yne=
	::= - a_{11} - yne=
⟨SUFFIX⟩	::= ⟨NULL⟩
(⟨NAME⟩)	::= (ca tree eth - a_{11} - yne=)

and by substitution we obtain

⟨NAME⟩	::= ca tree hept - d_0 , f_0 - bini ene= - d_0 - (ca tree eth - a_{11} - yne=) - e_0 - methyl

The numeral suffixes are now dropped as they are unambiguous and we get

⟨NAME⟩	::= ca tree hept - d_0 , f_0 - bini ene= - d - (ca tree eth - a - yne=) - e - methyl.

Note that if we have a more complicated substituent, we shall proceed through the recursion of ⟨NAME⟩ several times and substitute the final name. In such a case the labels of nodes at the ith inner level of parentheses are given as a_i, b_i, c_i, . . . (i = 1,2,3,...). Since these are now in nested parentheses, we can drop all the numeral suffixes in the final name and yet will be able to reconstruct these by an algorithm which computes the parentheses levels.

Example 2: Consider the structure in Figure 2. We can give a tree representation of the name derived; such a tree called the derivation tree[7] is given in Figure 3. This gives

⟨NAME⟩ ::= ca tree hex-a,c,e-terni ene=

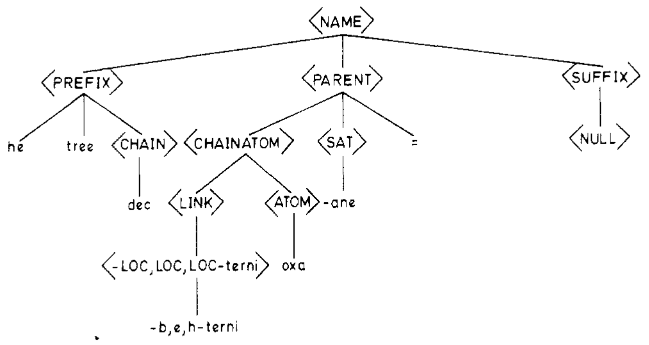


Figure 4. Derivation tree of he tree dec-b,e,h-terni oxa-ane=.

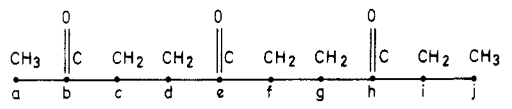


Figure 5. ca tree dec-ane = -b,e,h-terni ene oxa.

Example 3: Consider 2,5,8-trioxadecane. Using WISENOM grammar we get (Figure 4)

⟨NAME⟩ ::= he tree dec-b,e,h-terni oxa-ane=

Note that this phrase structure specifies the presence of three oxygen atoms in the main chain of a hetero tree, since the punctuation = separates out substituents or attachments. This helps in context-free decoding.

Example 4: Consider 2,5,8-trimethyldecane. The derived WISENOM is

⟨NAME⟩ ::= ca tree dec - ane = - b,e,h - terni methyl

indicating that all the 10 carbon atoms are in the main chain, and these have substituents in the locants b, e, and h (methyl groups).

Example 5: Consider 2,5,8-decanetrione in Figure 5. The derivation gives

⟨NAME⟩ ::= ca tree dec - ane = –b,e,h-terni ene oxa

If, however, we treat C=O as a group then

⟨name⟩ ::= he tree dec - b,e,h - terni carbonyl-ane=

(b) Decoding Acyclic WISENOM. For decoding, we need to ensure that the given name is syntactically well formed. The act of checking for well formedness is called parsing. We briefly outline the methods for parsing and checking validity or meaning of a name in section IV. In this section and in section IIId we shall assume that the names to be decoded are well formed.

The algorithm is as follows:

Step 1 (*Suffixing Locants*). Add suffix zero (0) to all the locant labels a,b,c,. . . in the main chain, viz., convert the labels to $a_0,b_0,c_0,\ldots$ Then level the parentheses and add a suffix $i1$ to the set of adjacent locants a,b,c,. . . in the ith level of parentheses, viz., convert these to $a_{i1},b_{i1},c_{i1},\ldots$ ($i = 1,2,3,\ldots$).

If there are several disjoint sets (say j sets) of adjacent locants (i.e., several chains) at the ith level, then add appropriate suffixes to these, viz., convert these to $a_{ij},b_{ij},c_{ij},\ldots$ where i denotes the level and j its chain index ($j = 2,3,4,\ldots$).

Step 2 (*Decoding* ⟨*PREFIX*⟩). If the first two words are ca tree or he tree decode; otherwise stop.

Step 3 (*Decoding* ⟨*PARENT*⟩). If ca tree is read, read next word which gives chain size; create appropriate number of carbon chain locants starting from a_0 and create the connection table a_0–b_0, b_0–c_0,. . . till the end of the chain. Otherwise, (i.e., he tree is read), read the next word which gives chain size; create appropriate locants starting from a_0, read ⟨CHAINATOM⟩ and fill the noncarbon atoms at appropriate locants.

Table III. Decoded Connection Table

locants	atoms	connections	bonds
a_0	C	a_0-b_0	single
b_0	C	b_0-c_0	single
c_0	C	c_0-d_0	single
d_0	C	d_0-e_0	double
e_0	C	e_0-f_0	single
f_0	C	f_0-g_0	double
g_0	C		

Table IV. Decoded Connection Table

locants	atoms	connections	bonds
a_{11}	C	a_{11}-b_{11}	triple
b_{11}	C		

Step 4. Read ⟨BOND⟩ if any at locant a_0 and fill appropriately between the locants; repeat until reading the delimiter =.

Step 5 (*Decoding* ⟨*SUFFIX*⟩): Read the locant at which substituent is attached (⟨SUFFIX⟩). If it is a terminal name, place them appropriately; else decode ⟨NAME⟩ using steps 2, 3, and 4; repeat for each substituent. Stop.

Example 6: Decode into connectivity table.

⟨NAME⟩ ::= ca tree hept-d,f-bini ene= -d-(ca tree eth-a-yne=)-e-methyl

Step 1: Suffix the locants.

⟨NAME⟩ ::= ca tree hept-d_0,f_0-bini ene = -d_0-(ca tree eth-a_{11}-yne =)-e_0-methyl

Step 2: Read ca tree.

Steps 3 and 4: ⟨CHAIN⟩ ::= hept; create the connection table (Table III).

Step 5: Fill in substituents.

locant	substituent
d_0	(ca tree eth-a_{11} -yne =)
e_0	methyl

Now we need to decode the substituent at d_0.

Step 2: Read ca tree.

Step 3: ⟨CHAIN⟩ ::= eth;.

Step 4: Create the connection (Table IV) table.

Step 5: No further substituents; stop. The structure is given in Figure 1.

(c) Encoding Cyclic Structures. The encoding procedure[5] for cyclic structures uses several major algorithms.

Algorithm 1: picking out a set of smaller size component rings in the given system (given as a connection table)

Algorithm 2: choosing a starting ring and successive rings for synthesis

Algorithm 3: assignment of locant labels

Algorithm 4: tessellation–knit–multiknit operations on the rings to synthesize the given system and recording this sequence of operations to arrive at a unique code for ⟨TOPOLOGY⟩

Algorithm 5: recording ring atoms and saturation in ⟨PARENT⟩ and substituents in ⟨SUFFIX⟩

The algorithms 1 and 2 above have been described by Krishnan and Krishnamurthy[5] in connection with encoding a chemical graph to ALWIN. Here I will assume that the smaller component rings are picked out, and the starting ring and successive rings for synthesis have been chosen by using the hierarchical rules described in that paper.[5] I will only describe algorithms 3, 4, and 5.

Algorithm 3: Locant Assignment Rule. This rule prescribes the hierarchy for assigning the alphabetic locants (small letters) to the ring atoms.

(i) Starting ring and substituent rings: If no node of the ring has been assigned a locant, then assign the highest order locant to that fusion node (called the starting node) which has a higher Morgan label; assign the successive locants for the other nodes traveling along the direction of the fusion edge of the ring.

If there is no fusion edge (as in monocycles), assign the highest order locant to that node which has a higher Morgan label. Then assign the successive locants traveling along that direction of the ring in which the neighbor of the starting node has a higher Morgan label.

(ii) Tessellating a ring system: While tessellating a new ring to a ring which has already been assigned locants, use the following rule: Assign the first locant in the new ring segment to that node which is adjacent to the lexicographically earlier locant of the fusion edge. The succeeding locants are then assigned along the direction of the formation of the new ring segment.

(iii) Knit: (a) The spiro ring of the first kind—while knitting a new spiro ring to a ring which has already been assigned locants use the following rule: Assign the first locant to that node in the new ring segment which has a higher Morgan label and continue assigning successive locants traveling along the spiro ring in that direction.

(b) Other rings—start from the node (in the ring for which the locants are already assigned) with the lexicographically earlier locant x and introduce successively the required locants along the ring segment toward the lexicographically later locant y of another ring (to which locants are already assigned).

(iv) Multiknit: In the case of multiknit we follow the same procedure as for knit. The knit which starts from a lexicographically earlier locant is cited first. If two knits start from the same locant, then the one which ends in a lexicographically earlier locant is cited first.

(v) Skipping of ring size for decoding: When an assemblage of rings is formed by using the above code, it might happen that while closing a ring by a knit/multiknit, some other ring might also be simultaneously closed. In this case the latter ring cannot participate in the decoding algorithm. We differentiate this by placing an adjective "quasi" to that ring size numeral; note that such quasi-rings are easily detected in the encoding process.

Algorithm 4: Encoding the topology by tessellation—knit operations. The topological description is based on the synthesis of the ring system by knitting or tesselating the various smaller size component rings in a definite order.[5] Tessellation is the process of fusing two rings over a common edge in which the locants or node labels are alphabetically successive. This is described by a slash (/) followed by the lexicographically earlier locant.

Knit is the process of connecting any two labeled nodes through a known number of nodes or locants to complete a ring. This is described by a colon (:) followed by a triplet, viz., the initial locant, number of (nonzero) intermediate locants, and the final locant or the initial locant, and a hyphen (-) indicating direct connection and the final locant.

By definition, no more than one tessellation is permitted on an edge. Additional rings, if any, on this edge have to be constructed by using only knit/multiknit operations.

Multiknit is the process of knitting several segments to complete a ring. This is described by a semicolon (;) followed by a quadruple, viz., the initial locant, number of (nonzero) intermediate locants, final locant and a comma or the initial locant, a hyphen indicating direct connection, and the final locant and a comma. Unlike knit, this sequence of quadruples can recur following a semicolon, in order to complete a ring (See Syntax).

Example 7: We will now encode into WISENOM the structure in Figure 6. Here the capital letters denote dummy labels, and the numerals denote Morgan labels. The smaller component rings identified are R_1, R_2, R_3, R_4, R_5, and R_6.

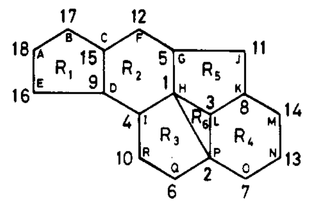


Figure 6. Encoding a cyclic structure.

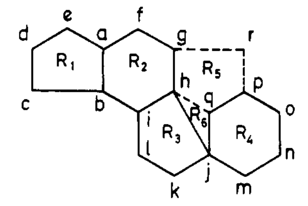


Figure 7. ca ring pent hex pent hex pent quasi-tri /a/h:j5j;glp,h-q,-j,h-bini sprio-ane =.

These rings are now assembled by the following operations using the hierarchical rules described by Krishnan and Krishnamurthy.[5]

Choose R_1 as the starting ring. The rings R_1 and R_2 have 15-9 as the fusion edge, and 15 being the higher Morgan label, we assign the label a to this node and b to the node with Morgan label 9, since this node is in the direction of the fusion edge of the rings. Assign successively the labels c, d, e to the nodes of R_1 (whose Morgan labels are 16, 18, and 17, respectively) as in Figure 7. In R_2, assign the label f to the node with Morgan label 12, as this node is adjacent to a, the hierarchically earlier locant and assign successively the labels g, h, i to the nodes with Morgan labels 5, 1, and 4, respectively. Since R_1 and R_2 have a fusion edge with two alphabetically successive neighboring locants a and b, this is a tessellation, and as a is the hierarchically earlier locant, it is denoted by /a.

Now we complete R_3: R_3 and R_2 have 1-4 as common edge with locants h and i. Assign the label j to the node with Morgan label 2, since it is adjacent to h, the hierarchically earlier locant in (h,i) and assign successively k, l to the nodes with Morgan labels 6 and 10, respectively. Since the R_3 and R_2 have a fusion edge with two alphabetically successive neighboring locants h and i, this is a tesselation, and as h is the hierarchically earlier locant, it is denoted by /h.

The next ring to be completed is R_4. Now R_4 and R_3 have 2 as the common node (assigned j in an earlier step). Assign the label m to the node with Morgan label 7 since it has the higher Morgan label among the nodes in the ring R_4 adjacent to j, and successively assign the labels n, o, p, q to the locants with Morgan labels 13, 14, 8, and 3, respectively. This is a knit operation in which j is the initial and final locant, and there are five intermediate locants. This is represented by :j5j.

Complete R_5 thus: R_2 and R_5 have 1-5 as a common edge; R_5 and R_4 have 3-8 as a common edge, and R_5 and R_6 have 1-3 as a common edge. Assign label r to the node with Morgan label 11. R_5 is formed by the process of knitting g and p with an intermediate locant r and h and q with no intermediate locants. This multiknit is represented by

;glp,h-q,

Incidentally, we observe that R_6 gets formed as a byproduct, while closing R_5 by a multiknit; this is a quasi-ring. Thus we have

⟨PREFIX⟩	::= ca ring pent hex pent hex pent quasi-tri
⟨PARENT⟩	::= ⟨TOPOLOGY⟩⟨RINGATOM⟩-⟨SAT⟩=
⟨TOPOLOGY⟩	::= /a/h:j5j;glp,h-q,

We proceed further to write down ⟨RINGATOM⟩, ⟨SAT⟩, and ⟨SUFFIX⟩ phrases.

Algorithm 5: Recording ring atoms, saturation and substituents. We now describe the rules for recording these in ⟨PARENT⟩ and ⟨SUFFIX⟩.

(a) ⟨PARENT⟩ rule: By the syntax rules 2 and 2.2

⟨PARENT⟩ ::= ⟨TOPOLOGY⟩⟨RINGATOM⟩⟨SAT⟩=

⟨RINGATOM⟩ ::= ⟨NULL⟩|(⟨LINK⟩(⟨ATOM⟩|⟨RADICAL⟩|⟨FUNGROUP⟩))$^+$

Here

⟨LINK⟩ ::= -⟨LOC-⟩|-⟨LOC,LOC-bini⟩| . . . (rule 3.1)

The phrase ⟨RINGATOM⟩ describes the locants, the nature of atoms, and their multiple occurrence. The phrase ⟨SAT⟩ indicates saturation/unsaturation. When a double bond occurs between two alphabetically successive neighboring locants, only the earlier-locant and bond nature is mentioned; if, however, there is a bond between neighboring locants *not* alphabetically successive, both the locant labels in alphabetical order are listed and the bond nature is included in between. If there are no double and triple bonds then the term-ane is added. Thus the syntax for ⟨SAT⟩ is given by rules 2.4, 2.41, and 2.42 (Table I).

When there is a saturated atom or spiro atom in a ring we denote this by sat or spiro, respectively, in that ring locant under ⟨RINGATOM⟩. Note that in the phrase ⟨ATOM⟩ (rule 4) we have included all noncarbon, spiro, and saturated atoms.

We also use the convention that the locants are ordered alphabetically in all the phrases.

(b) ⟨SUFFIX⟩ rule: When the substituent is acyclic, its ⟨NAME⟩ is recorded within parentheses with the ⟨LINK⟩ which describes the locant and ⟨BOND⟩, the nature of the bond. When the substituent is just an atom (⟨ATOM⟩), functional group (⟨FUNGROUP⟩), or radical (⟨RADICAL⟩), these are also indicated with ⟨LINK⟩ and ⟨BOND⟩, the nature of bond. When the substituent is cyclic, its ⟨NAME⟩ is recorded within parentheses along with the ⟨LINK⟩ which describes the linking main locant, bond nature (⟨BOND⟩), and its locant where it is linked (⟨LINK⟩). These aspects are clear in rule 3. In all the above phrases the locants are ordered alphabetically.

We now continue with example 7 where we arrived at the ⟨PREFIX⟩ and ⟨TOPOLOGY⟩ phrases.

Using the above rules we obtain

⟨RINGATOM⟩ ::= (⟨LINK⟩⟨ATOM⟩)
::= -⟨LOC,LOC-bini⟩ spiro = -j,h-bini spiro

⟨SAT⟩ ::= -ane

⟨SUFFIX⟩ ::= ⟨NULL⟩ = empty

⟨NAME⟩ ::= ca ring pent hex pent hex pent quasi-tri /a/h:j5j;glp,h-q,-j,h-bini spiro-ane=

The derivation tree for this name is shown in Figure 8.

Example 8: In Figure 9 the labeling of locants and the choice of parent ring are governed by the hierarchy rules for ALWIN.[5] Note that there are three levels of locant labeling:

a_0, b_0, c_0, d_0, e_0, f_0	for parent ring
a_1, b_1, c_1	for substituent attached to d_0
a_2, b_2, c_2, d_2, e_2	for the substituent attached to c_1

The name is written as

⟨NAME⟩ ::= ca ring hex-ane = -d_0-(he tree prop-c_1-oxa-ane= -c_1-(-d_2-he ring pent-e_2-thia-ane=))

We now remove the numeral suffixes as they are unambiguously (in a context-free sense) enclosed in different parentheses levels.

⟨NAME⟩ ::= ca ring hex-ane=-d- (he tree prop-c-oxa-ane= -c-(-d-he ring pent-e-thia-ane=))

(d) Decoding WISENOM for Cyclic Structures. As before,

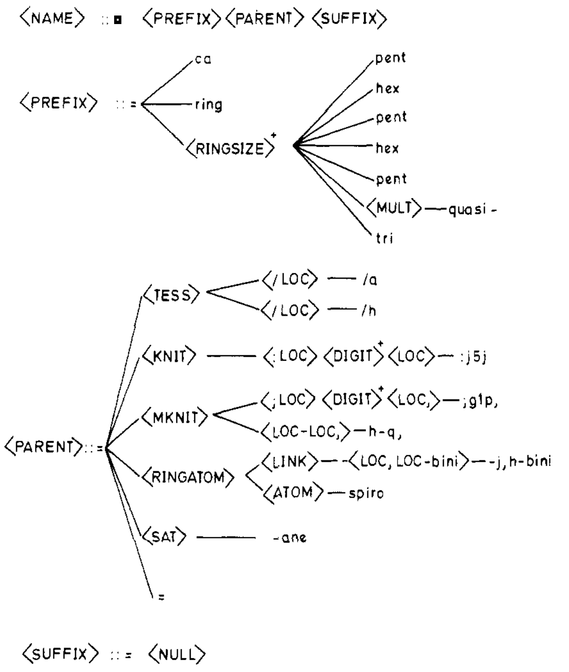


Figure 8. Derivation tree.

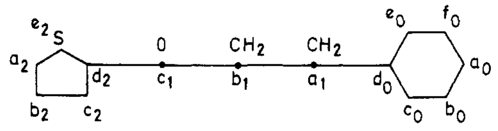


Figure 9. ca ring hex-ane = -d-(he tree prop -c-oxa-ane = -c(-d-he ring pent-e-thia-ane =)).

we assume that the name is syntactically well formed. The first two words—he or ca and tree or ring—are identifiers for starting the decoding process. Read the size of the first ring; generate this ring and assign its locants. Read the size of the next ring and the locant symbol in the ⟨TOPOLOGY⟩ phrase. If tesselation is indicated at this locant, we tesselate the new ring by generating the (second ring size − 2) succeeding locants, assigning the higher order locant to that node which is adjacent to a lexicographically earlier locant.

If, however, the ring to be placed next is by a knit or multiknit, form this new ring by generating the required number of new intermediate locants; check whether this satisfies the ring size; assign the higher order locant to that node which is adjacent to the lexicographically earlier locant.

The above process is repeated over again until all symbols in the tessellation-knit part are read and decoded. If a "quasi" symbol precedes a ring numeral (size), skip this ring formation and proceed to the next numeral before reading the locant symbol from either the tesselation or the knit part.

After the topology is decoded, the ring atoms, substituents and saturations are filled in at the appropriate locants.

Example 9: Decode

ca ring pent hex pent hex pent
quasi-tri/a/h:j5j;glp,h-q,-j,h-bini spiro-ane=

Here the first ring is a five-membered ring. Generate a five-membered ring and assign locants a–e to its nodes. The next ring (a six-membered ring) is tesselated to the first ring at edge ab. Assign locants f–i to its nodes as shown in Figure 7. Continuing this way, at step three we have a knit having five intermediate locants, both the ends connected to j. Now at step 4 we arrive at the structure shown as unbroken lines. We have multiknit consisting of a knit between g and p, with one intermediate locant r and a knit between h and q with no intermediate locants. The multiknit is shown by dotted lines (Figure 7).

The next ring size is preceded by a quasi, indicating that this quasi-ring does not take part in the decoding algorithm.

The ⟨RINGATOM⟩ contains only carbon atoms, and two of them are spiro. The total structure is saturated as indicated by -ane.

IV. PARSING WISENOM

Since WISENOM is based on a formal grammar with a well-defined syntax, the very first step is to check for the well formedness of the ⟨NAME⟩ by any suitable parsing technique.[7] The check for well formedness consists in looking for the correct ordering of phrases and punctuations and constructing the derivation tree to ensure unambiguous and faultless decoding of the given name. In this stage the syntactic errors are detected.

A few common examples of syntactic errors are (a) missing left or right parentheses, punctuation symbols—insertion, deletion, misuse, (b) misspelled words, and (c) incorrect phrase structure.

If the ⟨NAME⟩ is well formed, it does not imply that it is semantically and chemically correct. At this point it is worthwhile recalling the simple difference between syntax and semantics. The syntax of a language is a set of rules for assembling the basic symbol words into sentences. The semantics of a language is the set of rules for interpreting the meaning of a sentence. This meaning will depend upon the specific words chosen, the syntactic structure of the sentence, and often its context. For example, although "the apple ate the man" is an English sentence, its meaning is obscure. Thus the syntax is independent of the meaning; however, the converse is not true, that is, the meaning is quite dependent on the syntax. Accordingly, although by the rules of the syntax we can produce certain names resembling chemical names, not all of them are really chemically meaningful (of course, at the surface level our chemical meaning ends with valency checks—but not the existence of such a structure). However, since the rules of syntax must allow the semantics to be unscrambled, we can perform certain semantic checks—logical, topological, and chemical. If there is a failure, the ⟨NAME⟩ is then not decodable.

(i) Logical and Topological Errors. Typically these are errors of declaration, errors in the range of locants, and logical inconsistency.

(a) Number of rings: Since each tessellation (/); knit (:) and multiknit (;) complete a full ring (and in some cases a quasi-ring), the number of real rings (nonquasi) $N(\mathrm{R})$ is given by

$$N(\mathrm{R}) = N(/) + N(;) + N(:) + 1$$

Here $N(x)$ stands for number of occurrences of x in WISENOM, (x = /, :, ;). This is checked.

(b) Ring-size consistency: This check can be carried out only for knits/multiknits due to the redundancy available. The ring size of a knit/multiknit ring equals 2 + number of intermediate locants when the two end locants are distinct or 1 + number of intermediate locants when the two end locants are the same (as in a spiro, see Figure 7). This should be checked.

(c) Last locant while completing every ring: The value of the last locant (in base 23) is given by

n = size of first ring + sum of sizes of tessellating rings − $2N(/)$ + sum of numerals indicating intermediate locants in knit and multiknit

In Figure 7, we have $n = 5 + 6 + 5 + 5 + 1 - (2 \times 2) = 18 = \mathrm{r}$.

(d) Tessellation check: We recall that no fusion edge can have more than one tessellation, and tessellation is not permitted on the nth locant of a ring of size n (e.g., on the 6th locant in a 6- membered ring). Therefore, if tessellations occur at the same locants, or at the last locant of a ring [computed by (c)], the ⟨NAME⟩ is invalid.

Also when two successive tessellations occur at alphabetically adjacent locants say α and $\alpha + 1$, then a spiro at $\alpha + 1$ must be indicated.

(e) Undefined locants: If the ⟨CHAIN⟩ or ⟨RINGSIZE⟩ is declared and a description of a locant that is beyond the range occurs, such a name is invalid. This check is very useful during the decoding of ⟨TOPOLOGY⟩; at any intermediate step, if any undefined locant occurs, error is indicated.

(f) Declaration error: If a ring or chain is carbo (hetero) and the ⟨RINGATOM⟩ or ⟨CHAINATOM⟩ contains noncarbon atoms (no hetero atoms), this is a declaration error.

Another example of declaration error is when ⟨PREFIX⟩ describes tree (ring), and the ⟨PARENT⟩ describes a ring (tree).

(ii) Nonhydrogenic Valency Check. Since we assume that we are coding only the hydrogen-suppressed chemical graph, we can at the end of decoding compute the nonhydrogenic valency (NHV) from the bond nature between a node i and its neighbors.

If NHV of a node i is denoted by Δ_i, then the number of hydrogen atoms $N(\mathrm{H}_i)$ to be filled in at i is given by

$$V_i - \Delta_i = N(\mathrm{H}_i)$$

where V_i is the chemical valency. If $N(\mathrm{H}_i) < 0$, then error is indicated, and the ⟨NAME⟩ is invalid.

V. CONCLUDING REMARKS

Since the WISENOM is based on graph encoding for obtaining a unique name, very small changes in the structure (even in substituents or ring atoms) could result in a large change in the actual name. This is not a drawback of WISENOM only as every system which picks up a main or a parent system by a hierarchy and describes the remainder as substituents will have similar drawbacks. In other words, while assigning a logical name to a total (whole) entity in terms of its component names (parts), even if one can bring in uniqueness, one cannot guarantee that the total name will not mutate if one of its components undergoes even the slightest change.

Finally a few remarks are in order concerning the nomenclature systems. An ordinary chemist may voice his opinion as to why yet another new nomenclature system is needed when he is already overburdened with numerous such systems and especially so when the existing so-called systematic nomenclature system, while not completely systematic, is satisfying and systematic enough for him to encode and decode majority of the compounds. To them our answer is that the present attempt is *not to* devise a new system for its own sake in order to confuse the chemist *but for* obtaining a purely mechanical naming procedure for an organic compound on the basis of a formal set of rules. In this sense this system is bound to have the same psychological effects and drawbacks as the use of the binary number system (vs. the decimal system) or the polish arithmetic expression (vs. the conventional parenthesized arithmetic expressions). It is expected, however, that a context-free-grammar-based nomenclature system could prove very useful for computer information retrieval and display systems.

ACKNOWLEDGMENT

I am indebted to the referees and Editor Herman Skolnik for many valuable comments in improving the presentation of this paper. I am also grateful to Drs. R. Fugman, A. L. Goodson, K. L. Loening, M. F. Lynch, and William Wiswesser for constructive remarks and criticism through private discussions and correspondence.

REFERENCES AND NOTES

(1) Fletcher, J. H.; Dermer, O. C.; Fox, R. B. "Nomenclature of Organic Compounds"; American Chemical Society: Washington, DC, 1974.
(2) Goodson, A. L. "Graph-Based Chemical Nomenclature. 1. Historical Background and Discussion. 2. Incorporation of Graph-Theoretical Principles into Taylor's Nomenclature Proposal". *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 167-176.
(3) Smith, E. G. "The Wiswesser Line Formula Chemical Notation"; McGraw Hill: New York, 1968.
(4) Krishnamurthy, E. V.; Sankar, P. V.; Krishnan, S. "ALWIN-Algorithmic Wiswesser Notation System for Organic Compounds". *J. Chem. Doc.* **1974**, *14*, 130-141.
(5) Krishnan, S.; Krishnamurthy, E. V. "Compact Grammar for ALWIN Using Morgan name". *Inf. Process. Manage.* **1976**, *12*, 19-34.
(6) Backus, J. W. "The Syntax and Semantics of the Proposed International Algebraic language of the Zurich ACM-GAMM Conference". Information Processing, Proceedings of ICIP Paris, UNESCO, Paris, pp 125-132.
(7) Aho, A. V.; Ullman, J. D. "Principles of Compiler Design"; Addison-Wesley: Reading, MA, 1977.

Computer Storage and Retrieval of Generic Structures in Chemical Patents. 4. An Extended Connection Table Representation for Generic Structures

JOHN M. BARNARD, MICHAEL F. LYNCH,* and STEPHEN M. WELFORD

Department of Information Studies, University of Sheffield, Sheffield S10 2TN, United Kingdom

Received April 5, 1982

A data structure for the unambiguous representation of generic structures at the machine level is described. It is designed for automatic generation from structures encoded in the formal language GENSAL and is based on connection tables. Its relationship with other forms of representation is discussed.

INTRODUCTION

Paper 2 in this series[1] described a formal language, GENSAL, which has been designed for the encoding of generic, or Markush, structures in a form which is intelligible to a chemist or patent agent and also amenable to automatic analysis by computer. It was suggested that GENSAL is analogous to a high-level programming language and that the program which analyzes it can be thought of as equivalent to a compiler. To extend this analogy further one can compare the internal representation generated by the GENSAL interpreter program with the object code produced by a programming language compiler, and though unlike the object code for a programming language, the internal representation is a machine-level data structure rather than a set of machine-level instructions.

In a generic structure information system, this internal representation can be used to generate fragments for use in searching or used directly for atom-by-atom tracing in the final stage of a search. In order to enable it to perform these functions satisfactorily, and yet remain in a form which can easily be generated from GENSAL input, we have incorporated a number of features into its design, and these will be described in this paper. More detailed considerations of its use in fragment generation and searching will be the subject of future papers in this series.

REQUIREMENTS FOR THE REPRESENTATION

The first paper in this series[2] discussed the need for a full and unambiguous description of the generic structure, from which fragment screen descriptors of various types could be generated algorithmically, and the reasons for the selection of connection tables as the appropriate basis for this representation.

The purpose of the representation described here is not to store explicitly all the possible specific structures covered by a given generic structure but rather to contain sufficient information for exhaustive generation of all the specific structures to be possible, even though in most cases such an operation would be pointless, as well as computationally unfeasible where the number of specific structures covered in large, or even infinite.

Since the representation is to be built up from a generic structure input to the computer in GENSAL, the conversion problems will be greatly simplified if certain features of the representation mirror features of GENSAL. In particular, as the syntax for the definition of *substituents* in GENSAL[3] is essentially recursive, the structure of the internal representation should be recursive also.

GENSAL views a generic structure as consisting of a (possibly vestigial) constant part, to which are attached variable parts which can vary in their chemical nature, position of attachment, and multiplicity of occurrence and which may themselves be further substituted by other constant and variable parts down to any level. At each level, certain of the values for the variable parts may be alternative or additional to each other in complex nested Boolean relationships. This suggests two principal components for the internal representation, one containing information about the chemical nature of the constant and variable parts and the other containing information about the way in which they are connected together in terms of positions and frequencies of occurrence and the Boolean relationships between them. The successive levels of further substitution imply a hierarchical relationship between the different parts of the structure, though the exact nature of the hierarchy depends on the way in which the GENSAL description of the structure was constructed, which is to a certain extent arbitrary. It is also possible for the hierarchy to "loop back" to a higher level, in which case a recursive definition of a substituent (i.e., one in which the substituent is defined in terms of itself) will appear in the GENSAL. In this case there is no lowest level of substitution, and the structure in question is a polymer.

Together the two components of the internal representation can be considered as forming a topological graph, the chemical nature of the various parts of the generic structure being