

Science through awards of Information Science Research Studentships to J.M.B. and S.M.W. is acknowledged.

REFERENCES AND NOTES

- (1) M. F. Lynch, J. M. Barnard, and S. M. Welford, "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 1. Introduction and General Strategy", *J. Chem. Inf. Comput. Sci.*, preceding paper in the issue.
- (2) J. E. Hopcroft and J. D. Ullman, "Formal Languages and their Relation to Automata", Addison-Wesley, Reading, MA, 1969.
- (3) K. Jensen and N. Wirth, "Pascal User Manual and Report", Springer Verlag, New York, 1975.
- (4) N. Wirth, "Algorithms + Data Structures = Programs", Prentice Hall, Englewood Cliffs, NJ, 1976.
- (5) E. A. Geivandov, "Language for Notation of Generalized Structures of Organic Compounds Containing Alternative Delocalized Fragments (Markush Structures)", *Nauchno-Tekhn. Inf. Ser.* 2, (10), 21-24, 46 (1972).
- (6) A. V. Aho and J. D. Ullman, "Principles of Compiler Design", Addison-Wesley, Reading, MA, 1977.
- (7) J. Welsh and M. McKeag, "Structured System Programming", Prentice-Hall, Englewood Cliffs, NJ, 1980.
- (8) S. M. Welford, M. F. Lynch, and J. M. Barnard, "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 3. Chemical Grammars and Their Role in the Manipulation of Chemical Structures", *J. Chem. Inf. Comput. Sci.*, following paper in this series.
- (9) J. M. Barnard, M. F. Lynch, and S. M. Welford, "GENSAL (Generic Structure Language). Instruction Manual", University of Sheffield Postgraduate School of Librarianship and Information Science, in preparation.
- (10) R. J. Feldmann, G. W. A. Milne, S. R. Heller, A. Fein, J. A. Miller, and B. Koch, "An Interactive Substructure Search System", *J. Chem. Inf. Comput. Sci.*, 17 157-163 (1977).
- (11) P. A. Machin, J. N. Mills, O. S. Mills, and M. Elder, "CSSR Crystal Structure Search Retrieval", Revised Version, Daresbury Laboratory, Science Research Council, Warrington, England, 1977.
- (12) G. W. A. Milne and S. R. Heller, "NIH/EPA Chemical Information System", *J. Chem. Inf. Comput. Sci.*, 20, 204-211 (1980).

Computer Storage and Retrieval of Generic Chemical Structures in Patents. 3. Chemical Grammars and Their Role in the Manipulation of Chemical Structures

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

Postgraduate School of Librarianship and Information Science, University of Sheffield, Western Bank, Sheffield S10 2TN, United Kingdom

Received March 25, 1981

A simple topological chemical grammar is developed, and its possible applications to the computer manipulation of chemical structures are discussed. The generative and recognitive capabilities of the grammar are illustrated by examples. The paper concludes by identifying the role of such capabilities in a generic (Markush) structure search system.

INTRODUCTION

A small number of papers has appeared in the literature which report the application of formal linguistic theory to problems encountered in the representation and processing of chemical information. Linguistic considerations were implicit in the early development of systematic chemical nomenclature, and subsequent applications have been made primarily in the continued design of nomenclature and notational systems and their interconversion with other forms of structure representation.¹⁻⁷ Garfield³ used structural linguistics as the basis of an algorithm for generating molecular formulas from a subclass of chemical names. The conversion of systematic names of organic compounds into topological representations has been investigated by Vander Stouw⁶ and more recently by Carpenter.⁷

Fehder and Barnett⁸ reported early work on the automatic syntax analysis of linear structural formulas. This was proposed both as a preliminary to their conversion into chemical names and as a possible means of substructure search. More recent work along similar lines has been reported by Barker⁹ in which a syntax for molecular formulas is described, together with a simple parsing algorithm which enables the correctness of formulas to be verified. It was suggested that such capabilities should find application in interactive computer-based learning and chemical data base query systems.

The role of chemical structure diagrams in the natural language of the chemist is well recognized. Structure diagrams are used both as an aid to written communication and as a means of recording structural information for archival and retrieval purposes. Rankin and Tauber¹⁰ drew attention to the analogies that exist between chemical structure diagrams

and formal languages and suggested the value of chemical grammars as a means for structure analysis. In a subsequent paper,¹¹ two simple structural grammars were presented, a topological and a geometric grammar. As their names suggest, the first treats a chemical structure diagram as a topological graph, ignoring the actual physical dimensions such as bond lengths and angles, while the latter takes account of the geometric arrangement of atoms and bonds in the plane. Underwood and Kanal¹² provided a formalization of the topological grammar described by Rankin and Tauber in terms of a web grammar, a class of graph grammars previously introduced by Pfaltz and Rosenfeld.¹³

A number of applications of topological grammars can be identified. Syntactic analysis allows the "correctness" of structure representations to be verified. Checker programs are used routinely in retrieval systems which employ a notational representation of chemical structure.¹⁴ This capability assumes particular importance in interactive retrieval systems in which the user himself communicates directly with the computer. Online structure registration and query formulation via direct input of structure diagrams are becoming increasingly commonplace with the use of graphics terminals. In the case of compound registration, file corruption can be avoided by trapping invalid structure descriptions as they are entered. Similarly, wastage of machine time resulting from the submission of erroneous query formulations can be eliminated by prior syntax analysis.

A number of other workers have made either implicit or explicit use of linguistic techniques for chemical structure manipulation. The syntax of a language, as will be described, is defined in terms of a number of "productions" or

"replacement rules". Each production describes a permitted transformation among syntactic classes, symbol strings and individual symbols of the language vocabulary. This same concept of transformation is of course fundamental to our understanding of chemical reactions. It is not surprising therefore to find that computer-assisted organic synthesis design packages such as SECS¹⁵ and SYNCHEM¹⁶ make use of productions or "transforms" to describe feasible structural transformations. Whitlock describes a chemical grammar which defines a range of functional group transformations¹⁷ and makes explicit use of this solution to the "functional group switching problem"¹⁸ in a synthesis planning program.¹⁹

The objective of this paper is to investigate further the application of chemical grammars to the machine manipulation of chemical structure representations and, in particular, generic chemical formulas. Areas of application which have been identified include the description of structural classes, the enumeration and selective generation of instances of these classes, and the categorization of partial structure representations. The work reported here builds on the preliminary study by Krishnamurthy and Lynch²⁰ of the design of a machine representation for generic chemical formulas. Previous papers in this series describing work in progress at the University of Sheffield are published in this journal.^{21,22}

In the following sections we introduce some essential concepts from formal language theory and describe a simple topological chemical grammar. The generative and recognitive capabilities of this grammar are illustrated by examples, and the role of chemical grammars within a generic structure storage and retrieval system is discussed.

SOME FORMAL LINGUISTIC DEVICES AND DEFINITIONS

The study of the fundamental constructs and capabilities of language is a relatively recent departure for structural linguists. This has been brought about in large part by the opportunities offered by the introduction of computing power—opportunities for machine translation, language comprehension, document content analysis, etc. A significant body of knowledge has developed rapidly, while at the same time the real complexity of natural languages has begun to be appreciated. A great deal of success has been achieved in the investigation and modeling of subsets of these languages. For example, the specification of programming languages has been set on a more rigorous footing, a process initiated by the definition of the algebraic language ALGOL 60.²³ This has led to a greater understanding of their capabilities, limitations, and powers of expression.

A given language L , however this might be defined, comprises a set of sentences, $L(G)$. Each sentence is made up by the concatenation of a number of smaller units—in natural languages these consist of clauses, phrases, vocabulary units, and ultimately the primitive symbols of the language alphabet itself. A means of defining the language L is to specify the rules of operation of an abstract machine which is capable of recognizing all those and only those sentences which belong to the set $L(G)$. Such a machine provides the capability of determining the inclusion of a given sentence within the language. Conversely, we might specify the rules of operation of an abstract machine which is able to generate rather than recognize all of the sentences, and only those sentences, of the language. The latter forms the most convenient approach to language definition.²⁴ The rules of the generating machine comprise the syntax of the language.

Consider the simple language L_k , each of whose sentences comprises a sequence of n "a" symbols, followed by a sequence of n "b" symbols. More precisely, $L_k = \{aaaa \dots ab. \dots bbb; 0$

$< n \leq \infty; n_a = n_b\}$. Let us now define an abstract machine capable of generating these sentences. The machine takes the form of a number of transformation functions, each of which has the power of mapping a given symbol or symbol string into a nonidentical symbol string. The simplest sentence of the language L_k can be generated by one such function

$$R_1: S \rightarrow ab$$

which is interpreted as "symbol S is replaced by the symbol string ab ". We note the following: the lower case symbols "a" and "b" belong to the "terminal vocabulary" (V_T) of the language, as they appear in the sentences of that language. The upper case symbol S is not permitted to appear in any sentence of the language—it is a linguistic device known as a metasympol or class symbol. Such symbols belong to the "nonterminal vocabulary" (V_N).

Our single "replacement rule" R_1 is sufficient to generate only a single sentence of the language L_k . Does this imply that an infinite number of such rules is required to generate the infinite number of sentences of L_k ? Thankfully no. A single additional replacement rule is sufficient to complete the definition of this language:

$$R_2: S \rightarrow aSb$$

Note that R_2 is "self-embedding" and as such ensures the required symmetry of the sentences in L_k . Replacement rules of the type R_2 are known as "propagation rules" and those of type R_1 as "termination rules".

The combination of rules R_1 and R_2 is conveniently written

$$R_k: S \rightarrow ab \mid aSb$$

where the "|" represents the selection operator "or". Note that R_k is incapable of generating any string consisting of symbols "a" and "b" which is not a sentence of L_k .


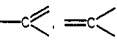

We now define a grammar for this language. The grammar G_k defines the language L_k in a recursive manner and can be stated as a 4-tuple:²⁴

$$G_k = \{S, V_N, V_T, R_k\}.$$

S is a distinguished symbol in V_N called the "sentence symbol". $V_N = \{S\}$ is the set of metasympols, $V_T = \{a, b\}$ is the set of terminal symbols, and $R_k = \{S \rightarrow ab \mid aSb\}$ is the set of replacement rules. V_N and V_T are disjoint symbol sets, their intersection being empty: $V_N \cap V_T = \{\emptyset\}$. Note that in this simple language the nonterminal set V_N exists as a singleton. This is rarely the case for more elaborate languages. A given language can generally be defined with equal validity by more than a single grammar. It is usually the objective of the linguist to design a grammar which combines both conciseness and economy of replacement rules, with maximum descriptive power.

The grammar G_k has been defined in terms of its capacity to generate sentences of the language L_k . As indicated previously, the abstract machine so defined has the additional capability of acting as a recognizer for sentences of the same language. That is, by reversing the direction of the transformation function of each replacement rule in the set R_k , a sentence of L_k can be reduced, or parsed, to the sentence symbol S by successive application of these inverted rules. Thus from the above example, the set of inverse replacement rules, $R'_k = \{R'_1, R'_2\} = \{S \leftarrow ab \mid aSb\}$. These rules describe

Table I. Bonded-Atom Species and Corresponding Terminal Symbols

bonded atom	terminal symbol
$-\text{CH}_3$	X
$-\text{CH}_2-$	1
$-\text{CH}=\text{CH}-$	2, 4
$-\text{C}\equiv\text{C}-$	3, 6
$=\text{CH}_2$	Y
$=\text{C}=\text{C}$	5
$\equiv\text{CH}$	Z
	7
	8, 9
	0

the inverse machine or recognizer for the language L_k , namely, G_k' .²⁵

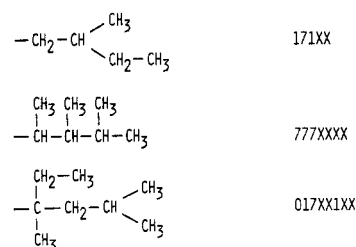
Consider the terminal sentence "aaabbb". Application of rule R_1' to the substring "ab" results in the partially reduced sentence "aaSbb". Two applications of the rule R_2' result in complete reduction to the sentence symbol S . The sentence "aaabbb" is therefore recognized as being a valid sentence of the language L_k . Note that complete reduction of the sentence "aaabbb" is not possible by using G_k' , and that because G_k' sufficiently defines the language L_k , incomplete reduction suffices to demonstrate the nonmembership of a given sentence in the language.

A GENERATIVE CHEMICAL GRAMMAR

In order to define a grammar capable of generating chemical structures, we require both a terminal vocabulary from which a structure representation can be assembled and a set of replacement rules which determine the way in which these symbols may be juxtaposed.

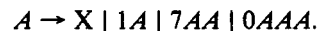
We have chosen to use bonded atoms as the structural building blocks, as they collectively provide a terminal vocabulary of convenient size. These reflect the various bonding patterns associated with a given atom. The normal bonding patterns for carbon are shown in Table I, together with the terminal symbols (V_T) used to represent them. The choice of these symbols is arbitrary, although there is some resemblance in principle to the unit symbols of the CROSSBOW connection table.²⁶ Hydrogen atoms are assumed to satisfy unspecified valencies. Using these symbols, an *n*-butyl radical can be represented linearly as 111X, in which the left-most symbol is understood to represent the externally connected (apical) atom.

We consider only monovalent acyclic radicals in the present discussion—that is, noncyclic partial structures in which there

**Figure 1.** Diagrammatic and linear representations of selected saturated hydrocarbon radicals.

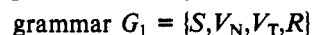
exists only a single apical atom. Multivalent substructures, such as polymethylene chains, are not considered further. It should be emphasized, however, that these are amenable to generation by using a suitably extended form of the grammar described below. Any acyclic radical can be considered as a tree, in which atoms and bonds are represented as nodes and edges, respectively. The connectivity of a given node represents the number of nonhydrogen connections of the corresponding atom. It is well-known that a tree can be represented in linear form either by using a parenthetical or by a so-called Polish notation. We have adopted the latter to provide a linear description of an acyclic radical. Hiz²⁷ and Eisman²⁸ have described Polish-type notations for acyclic and cyclic molecules. The Polish notation gives a depth first description of a tree and traces each branch (subtree) recursively until terminated before backing to the preceding branching point. The linear or string notations of some saturated acyclic hydrocarbon radicals are shown in Figure 1. The symbols X, 1, 7, and 0 comprise a subset of the bonded-atom symbols shown in Table I.

We can immediately specify four replacement rules which are sufficient to describe any structure of this type:

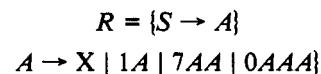


The metasymbol A is understood to represent any acyclic radical whose external connection is a covalent single bond. No further structural information is implicit in or intended by this symbol. The metasymbol A is found on the right-hand side of the three propagation rules by virtue of the fact that any tree (radical) can be defined recursively in terms of its component subtrees (branches).

A generative grammar for alkyl radicals can be defined as



where S is the sentence symbol ("structural symbol"), V_N , the nonterminal vocabulary, $= \{S, A\}$, V_T , the terminal vocabulary, $= \{X, 1, 7, 0\}$, and R is the set of replacement rules



This grammar assumes a normal carbon valency of four and allows for both ternary and quaternary branching atoms by the replacement rules $A \rightarrow 7AA$ and $A \rightarrow 0AAA$, respectively. Before illustrating the use of this simple grammar as a structure generator, we define an extended grammar capable of generating both saturated and unsaturated acyclic hydrocarbon radicals.

The rules which govern the concatenation of any two bonded-atom species must necessarily take into account the nature of the connecting bond. For example, it is clearly nonsensical to permit the juxtaposition of the species $-\text{CH}_2-$ and $=\text{C}=\text{C}$. The discussion in this and the preceding section has referred, although not explicitly, to context-free grammars.²⁴ This terminology is used to describe grammars each of whose replacement rules takes no account of the context

GRAMMAR $G_2 = \{S, V_N, V_T, R\}$
 WHERE S IS THE STRUCTURE SYMBOL,
 $V_N = \{S, A, B, C\}$
 $V_T = \{X, Y, Z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$
 $R = \{S \rightarrow A | B | C$
 $A \rightarrow X | 1A | 2B | 3C | 7AA | 8AB | 9AAA$
 $B \rightarrow Y | 4A | 5B | 9AA$
 $C \rightarrow Z | 6A \}$.

Figure 2. Grammar for the generation of acyclic hydrocarbon radicals.

in which the replacement symbols are situated. A more powerful class of grammars, but of correspondingly greater complexity, is the class of context-sensitive grammars. A degree of context sensitivity is implicit in the present case in which the assignment of a terminal symbol (V_T) is dependent upon the nature of the connecting bond implicit in the previous terminal symbol. This context dependency can be ignored in the special case of alkyl radicals, in which only a single bond type is possible. When considering the inclusion of unsaturated bonds, however, we need to make explicit in the replacement rules the concatenations which are possible between species of mixed bond type. We maintain the context-free nature of the grammar by introducing two additional metasymbols, B and C , into the nonterminal symbol set (V_N), and extending the set of replacement rules by the further addition of a number of context-independent replacement rules. While the metasymbol A describes any tree or sub-tree which has an apical single connection, the metasymbols B and C describe any tree which has an apical double and triple connection respectively.

The extended grammar G_2 is shown in Figure 2. This grammar defines the syntax and enables the generation of alkyl, alkenyl, and alkynyl radicals, their multiunsaturated derivatives such as dienyls, and radicals with unsaturated apical bonds. The terminal symbols used in the grammar G_2 are those of Table I. These symbols need not of course be associated solely with carbon atoms—they may be generalized to describe the bonding patterns of other atoms. Thus the symbols X , 1 , and Y can similarly be used to represent the oxygen species $-\text{OH}$, $-\text{O}-$ and $=\text{O}$, respectively. Extension of the symbol set would be required to describe additional bonding patterns not normally displayed by carbon. By qualifying each terminal symbol by the set of atoms which display that bonding pattern, the descriptive power of grammar G_2 can be extended to all saturated and unsaturated hydrocarbon and nonhydrocarbon acyclic radicals.

A means of substantiating this descriptive power is to utilize the grammar G_2 to generate the entire structure space of acyclic radicals, given selective constraints on atom number and type, and degrees of unsaturation and branching. Let us describe the mechanism of a generative grammar. We shall use for simplicity the alkyl grammar G_1 . Consider the following simple problem: the enumeration of saturated monovalent hydrocarbon radicals containing four carbon atoms.

We begin by applying the replacement rule $S \rightarrow A$, which replaces the sentence symbol S by the nonterminal symbol A , thereby initiating structure generation. We establish two variables, K and V_s . K is used to keep a count of the number of atoms which remain to be assigned to a partially completed structure at any stage in the generation process, while V_s records the number of unsatisfied (or pendant) connections present in each structure, excluding the single external connection. K and V_s are incremented or decremented after each application of a replacement rule, the values they adopt depending on the nature of the assigned terminal symbol. Each replacement rule has an associated condition, stated in terms of K and V_s , which determines the applicability of that rule.

Table II. Replacement Rules of the Grammar G_1 and the Corresponding Conditions Determining Their Applicability

replacement rule	conditions	modification to variables	
		ΔK	ΔV_s
$S \rightarrow A$	$K = K_0, V_s = 1$	0	0
$A \rightarrow 1A$	$K > V_s, V_s \geq 1$	-1	0
$A \rightarrow 7AA$	$K > V_s + 1, V_s \geq 1$	-1	+1
$A \rightarrow 0AAA$	$K > V_s + 2, V_s \geq 1$	-1	+2
$A \rightarrow X$	$K \geq V_s, V_s \geq 1,$ $K = 1$ iff $V_s = 1$	-1	-1

Table III. Redundancy in Isomeric Hydrocarbon Radical Generation

atom count	no. of alkyl radicals generated		redundancy, %
	non-redundant	redundant	
1	1	1	0
2	1	1	0
3	2	2	0
4	4	5	20
5	8	13	38
6	17	36	53
7	39	104	63
8	89	309	71
9	211	939	78
10	507	2905	83

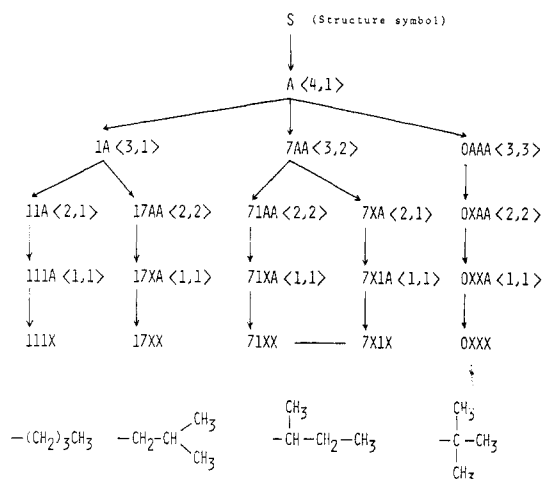


Figure 3. Iterative generation of isomeric hydrocarbon radicals. The values of K and V_s for each partial structure are indicated within angle brackets.

Table II specifies these conditions and indicates the modification made to each variable as a result of making a symbol assignment.

At the start of the generative process K and V_s are initialized, in this case to 4 and 1, respectively. Structure generation proceeds by the successive application of each replacement rule whose condition is satisfied by the values of K and V_s , until no atoms remain to be assigned ($K = 0$). This process is illustrated in Figure 3, together with the resultant structure space. We note immediately that the two linear representations 71XX and 7X1X describe identical structures. This is an example of redundant generation, and a means need be sought to eliminate such redundancy. The problem of redundancy grows more acute as the size of the structure space increases.

Table III illustrates the degree of redundancy experienced with the grammar G_1 if no means of suppressing redundant generation is applied. Redundancy in acyclic radical gener-

```

begin (* ALGEN *)
  K0, K <--- no. of atoms; J <--- 1; EXIT <--- FALSE
  STRING[J] <--- 'A' (* for radicals with an apical single covalent bond *)
  Put (INFILE,STRING)
  repeat
    read (INFILE,STRING)
    L <--- no. of symbols in STRING
    RR <---- first Replacement Rule
    repeat
      with RR do
        VSDT <--- (VS + VD + VT)
        if (VSDT ≠ 0) OR (L>J) OR (J=K0)
          then if K > (VSDT + (L-J))
            then STRING[J] <--- Replacement Sequence
              STRINGVAL (OK) (* determine canonicity *)
              if OK then write (OUTFILE,STRING)
            else EXIT <--- TRUE
          else EXIT <--- TRUE
        RR <---- next Replacement Rule
      until (RR = NIL) or EXIT
    EXIT <--- FALSE
  until End of File (INFILE)
  INFILE <---> OUTFILE; K <--- K-1; J <--- J+1
until K=0
end. (* ALGEN *)

```

Figure 4. Acyclic generative algorithm ALGEN. V_D and V_T record the number of double and triple bonds added to a structure as a result of application of a given replacement rule.

ation becomes possible at branching points. A general property of tree graphs is that the subtrees which extend from each branching node can be ordered by applying a chosen priority or "weighting" function. A unique linear representation for an acyclic radical can be identified as that which specifies the branches of the structure in a chosen order. All other equivalent representations are then redundant. We apply the following precedence rule as a means of selecting a unique structure representation:

"priority amongst the branches associated with a given branching atom is given to that branch which exhibits the earliest occurrence of an atom (excluding terminal atoms) of lowest connectivity".²⁹

This precedence rule has been found to be sufficient to identify a unique linear representation for a given acyclic radical represented in the form of a tree. An alternative approach, based not on the connectivity of each node of the tree but on its "distance" from the root, was adopted by Scoins³⁰ to derive a description of a tree in the form of an integer sequence. This allows both a canonical description to be identified and a lexicographic ordering of dissimilar trees to be made.

To minimize the computational requirements of the generative process described here, we require to eliminate provenly noncanonical partial representations at the earliest opportunity. The precedence rule described above can be applied after each

symbol assignment—partially complete but provenly canonical notations are retained while noncanonical notations are discarded. In the example of Figure 3, the incomplete notation 7X1A is identified as noncanonical and discarded, thereby suppressing generation of the redundant representation 7X1X. In this manner, by the application of a simple precedence rule, we are able to ensure nonredundant generation of acyclic radicals.

We have incorporated the grammar G_2 within an acyclic radical generator written in Pascal. The grammar is implemented as a linked list structure similar to that described by Cohen and Gotlieb.³¹ A more concise specification of the generative algorithm ALGEN for hydrocarbon radicals is stated in Figure 4.

The problem of exhaustive structure generation has been substantially solved within the DENDRAL project^{32,33} and elsewhere.^{34,35} Selective structure generation is of greater practical interest than exhaustive generation, although the latter has its value in confirming both the descriptive power of a grammar and the correctness of the generation algorithm. Constraints can be applied to ALGEN so that particular classes of acyclic radicals are enumerated. These take the form of requirements for the skeletal carbon atom count and, optionally, the number and type of branching atoms and unsaturations, rooted path lengths, and the number of heteroatoms. Table IV lists the standard constraints presently available,

Table IV. Parameter Identifiers and Their Meanings

parameter identifier	structural description
C	no. of skeletal carbon atoms
T	no. of ternary branch points
Q	no. of quaternary branch points
E	no. of isolated double bonds (ene)
Y	no. of isolated triple bonds (yne)
Z	no. of heteroatoms of unspecified type
P	required range of rooted path lengths

together with the corresponding parameter identifiers. Additional constraints can be stipulated, such as requirements for the number and type of specified heteroatoms. The use of such nonstandard parameter identifiers is described in the preceding paper in this series.²²

Implementing these constraints is particularly simple and requires only the extension of the conditions associated with each replacement rule. Thus if the space of unbranched hydrocarbon radicals of a given number of atoms is required, each of the parameters *T*, *Q*, and *Z* is set to zero. Replacement rules which assign ternary and quaternary branching atoms are suppressed and the inclusion of noncarbon atoms prevented. This simple approach to selective structure generation is similar in concept and intent to the more elaborate CONSTRAINTS function of the structure generator program CONGEN developed at Stanford.³² A greater sophistication can be envisaged by extending the terminal symbol set and replacement rules of the topological grammar to include particular multiatom assemblies and functional groupings in addition to single atom species.

An additional grammar is currently being developed for the description of cyclic structures. In its present form this grammar is sufficient to describe nonfused ring systems, and in conjunction with the acyclic grammar described here, it has been used for the selective generation of radicals containing arbitrarily substituted isolated rings. This grammar will be described in detail in a future publication in this series.

RECOGNITIVE CAPABILITIES OF A CHEMICAL GRAMMAR

We have previously mentioned the use of grammars for the syntax analysis of chemical nomenclature, linear formulas, and structure diagrams. This application constitutes a means of determining set membership. That is, it is possible to determine whether a given name, formula, or structure diagram belongs to the set of syntactically correct or the set of syntactically incorrect structure representations.

By imposing appropriate constraints on a grammar, it becomes possible to isolate one or more subsets of the language defined by that grammar. The constrained syntax then defines the boundaries of this subset of the language. Thus by setting the parameters *E*, *Y*, and *Z* of the acyclic grammar G_2 to zero, we effectively define the class of alkyl radicals. The recognitive capability of the "constrained" grammar enables us to determine not only whether a given structure representation is syntactically correct but also whether or not it constitutes, in this case, an alkyl group. It is clear that this offers a means of approximate substructure matching not otherwise provided for in existing substructure search systems. It assumes particular importance when dealing with generic chemical formulas in which homologous series and other generic expressions which describe classes of related substituent groups are commonly encountered.

A simple recognition algorithm has been implemented which makes use of the inverse of the acyclic grammar G_2 . The algorithm operates on a linear symbol string and is concep-

```

Generic Expression:---- '2-6C chloroalkenyl'
Parameter List:----- C<2-6> E1 'CL'1

Partial Structure:-----
      CH2-CL
      |
-CH2-C=CH2
Linear Representation:- 181X*CL*Y
Parse proceeds:----- 181X*CL*Y
                        181XB          -- Cl atom noted
                        181AB
                        18AB          -- double bond noted
                        1A
                        A
-- parse complete --
no. of C atoms = 4
no. of heteroatoms (Cl) = 1
no. of double bonds = 1
-- parameter requirements satisfied --

```

Figure 5. Illustration of the structure parsing algorithm, enabling the categorization of acyclic radicals.

tually similar to that described by Earley.³⁶ The linear structure representation used as input to the parser is derived from the atom-bond connection table which describes the given radical and consists of terminal symbols from the acyclic grammar G_2 . Each symbol describes a bonded-atom species as described previously and is qualified by atom type in the case of noncarbon atoms. The left-most symbol represents the apical atom of the radical. The derivation of the linear representation is relatively facile, each row of the connection table effectively describing an atom and its bonding pattern. The connections between atoms, implicit within the connection table, are made explicit in the linear representation by the order in which the symbols occur. The conversion algorithm traces a path through the connection table, backing up to previous branch points as appropriate. The resulting symbol string is an unambiguous although nonunique representation of the structure.

The parsing algorithm analyzes the linear structure representation in a right-to-left direction, tracing through the radical toward its root. The right-most terminal symbol is taken, the acyclic grammar is accessed, and the corresponding replacement rule is located.³⁷ The portion of the input string which matches the right-hand side of the replacement rule is replaced by the nonterminal symbol on the left-hand side of the replacement rule, thereby creating a new, "reduced" string. The right-most terminal symbol of the new input string is again taken and the algorithm iterates until either a single nonterminal symbol remains or no replacement rule can be found which permits further reduction. In the first case, complete reduction has been achieved and the structure representation is provenly syntactically correct. The second case indicates an error situation resulting from incorrect syntax.

The value of the parsing process lies, however, not so much in its ability to establish the correctness or otherwise of the syntax of a structure representation but in its ability to categorize that partial structure. That is, it enables membership within a defined class of rooted substructures to be determined. This is achieved by maintaining a cumulative count of the structural characteristics encountered during the parse. For example, when the parsing algorithm encounters the terminal symbol "4", a double bond is recorded. On completion of the parse the cumulated structural characteristics are matched against the constraining parameters used to define the intended class of radicals. If the former are included within or directly match the latter, then the partial structure is verified as being a member of that class.

Figure 5 illustrates the operation of the parsing algorithm. A simple unsaturated acyclic radical is parsed to determine whether or not it constitutes a member of the class described by the generic substituent expression "2-6C chloroalkenyl". The syntax of the parameter list which represents generic expressions of this type is described elsewhere.²² In the present case, the specification of the range of skeletal carbon atoms, together with a single chlorine atom and an olefinic unsaturation, is sufficient to characterize this generic expression. The symbol string *CL* is included within the linear representation to indicate that the preceding (left) symbol describes a chlorine atom. Note that although the partial structure includes a branching atom, the parameter list for "2-6C chloroalkenyl" imposes no restrictions on the number or type of branching atoms. Details of branching characteristics are therefore not cumulated.

The provisional cyclic grammar mentioned earlier has been used in conjunction with the acyclic grammar for the parsing and categorization of radicals containing isolated and connected monocycles. This makes possible, in a manner entirely analogous to that described, the determination of the inclusion of radicals containing monocycles within generic expressions such as "cycloalkyl", "aryl", etc. Extension of the cyclic grammar to permit the generation and recognition of polycyclic radicals containing fused rings is currently in hand. Details of this work will be reported in a future publication.

ROLE OF CHEMICAL GRAMMARS IN A GENERIC CHEMICAL STRUCTURE SEARCH SYSTEM

We have discussed in some detail the generative and cognitive aspects of a topological chemical grammar and have illustrated these by the generation and categorization of simple acyclic radicals. Simple generative and parsing algorithms have also been described. We turn now to discuss the opportunities offered by such capabilities and the role each may play within a generic chemical structure storage and retrieval system.

The use of generic terminology such as "alkyl", "alkoxy", and "aryl" to describe classes of substituent radicals needs no introduction for those with experience in handling generic chemical formulas. It is our belief that chemical grammars provide a convenient means of representing and manipulating generic expressions which are capable, in principle, of structural enumeration. Together with the appropriate constraints, a chemical grammar can be used to define the scope of expressions such as "alkyl", "alkoxycarbonyl", and "optionally unsaturated cycloalkyl". Individual members of such structural classes can be thought of as being held implicitly within the appropriately constrained grammar—particular instances may be made explicit by using the generative capability of the grammar.

We acknowledge that certain types of generic expression call into question the feasibility of structural enumeration and indeed its very possibility. The combinatorial implications of expressions such as "aryl" and "substituted heterocycle" are alarming, even for the selective generation of individual instances consistent with required structural characteristics. While the combinatorial complexity cannot be doubted, one should not underestimate the often remarkable descriptive power of even the simplest grammar. Topological chemical grammars are able to achieve this descriptive power, as we have attempted to demonstrate, not only as a result of the limited structural vocabulary required but also because of the restricted manner in which the symbols of this vocabulary may be assembled. Generic expressions which describe property-defined groups, such as "UV light stabilizing group" and "ester protecting group", are not amenable to structural definition and therefore fall outside the generative and cognitive ca-

pabilities of structural grammars. Alternative provision is required for expressions of this latter type within a generic structure storage and retrieval system.

In common with other structure storage and retrieval systems, a screening or file partitioning strategy is intended as a first-level search mechanism. Detailed consideration of the requirements and the type of screens most appropriate for the partitioning of files of generic chemical structures is deferred until a later paper. It suffices here to say that algorithmically defined screens appear to be most suitable, enabling the indexing of structures at varying degrees of generality and specificity. Considerable experience has been gained at Sheffield in the generation, manipulation, and information theoretic properties of such structural fragments,^{38,39} and their use in operational systems has been noted elsewhere.²¹ Efficient algorithms are required for generating screens of this type not only from the invariant part or parts of a generic structure but also from within variant parts and at points of connection between the two. The variant structural components typically consist of lists of well-defined alternatives as well as generic substituents both capable and incapable of enumeration.

Generative grammars would appear to have a role to play in screen generation, particularly where screens are required which extend into substituent groups described by homologous series and other structurally enumerable expressions. It is difficult to anticipate the problems likely to be encountered in screen generation, the most notable of which are likely to be associated with screen variety. These may be tempered somewhat by controlling the size and specificity of individual screens. It should be noted, however, that while this approach reduces the variety, the discriminatory capacity of the screens is also reduced. Preliminary investigations suggest that the variety of screens likely to be derived from the majority of the commonly occurring structurally enumerable expressions is not excessive.

The role of the cognitive capabilities of chemical grammars within the proposed system is less well defined. A possible application as an approximate structure-matching function has been illustrated. It is not yet clear at which stage in the retrieval process such a capability might be required. It may be considered as augmenting an iterative or atom-by-atom structure matching function, enabling query structures or parts thereof to be matched against the inexplicitly defined components of individual generic structures. The necessity for iterative structure matching, and indeed the appropriateness of such precise matching in the context of generic chemical structures, has not yet been established.

ACKNOWLEDGMENT

Thanks are due to the Department of Education and Science for the award for Information Science Research Studentships to S.M.W. and J.M.B. Thanks are also due to Peter Willett, members of the AIOPI Patents Study Group, and the staff of Derwent Publications Ltd. for helpful discussions.

REFERENCES AND NOTES

- (1) "IUPAC Definitive Rules for the Nomenclature of Organic Chemistry", *J. Am. Chem. Soc.*, **82**, 5545-5574 (1960).
- (2) "Survey of Chemical Notation Systems", Publication 1150, National Academy of Sciences-National Research Council, Washington, DC, 1964.
- (3) E. Garfield, "An Algorithm for Translating Chemical Names to Molecular Formulas", *J. Chem. Doc.*, **2**, 177-179 (1962).
- (4) P. M. Elliott, "Translation of Chemical Nomenclature by Syntax-Controlled Techniques", M.Sc. Dissertation, Ohio State University, 1969, unpublished.
- (5) D. J. Polton, "Conversion of the IUPAC Notation into a Form for Computer Processing", *Inf. Storage Retr.*, **5**, 7-25 (1969).
- (6) G. G. Vander Stouw, P. M. Elliott, and A. C. Isenberg, "Automated Conversion of Chemical Substance Names to Atom-Bond Connection Tables", *J. Chem. Doc.*, **14**, 185-193 (1974).

- (7) N. Carpenter, "Syntax-Directed Translation of Organic Chemical Formulas into Their Two-Dimensional Representation", *Comput. Chem.*, **1**, 25-28 (1976).
- (8) P. L. Fehder and M. P. Barnett, "Syntactic Scanning of Chemical Information", *J. Chem. Doc.*, **5**, 8-13 (1965).
- (9) P. G. Barker, "Syntactic Definition and Parsing of Molecular Formulas", *Comput. J.*, **18**, 355-359 (1975); **21**, 224-233 (1978).
- (10) K. Rankin and S. J. Tauber, "Linguistics as a Basis for Analyzing Chemical Structure Diagrams", *J. Chem. Doc.*, **11**, 139-141 (1971).
- (11) S. J. Tauber and K. Rankin, "Valid Structure Diagrams and Chemical Gibberish", *J. Chem. Doc.*, **12**, 30-34 (1972).
- (12) W. E. Underwood and L. N. Kanal, "Structural Descriptions, Transformation Rules, and Pattern Analysis", Proceedings of the First International Joint Conference on Pattern Recognition, Washington DC, 1973, pp 434-444.
- (13) J. L. Pfaltz and A. Rosenfeld, "Web Grammars", Proceedings of the International Joint Conference on Artificial Intelligence, Washington, DC, 1969, pp 609-619.
- (14) C. M. Bowman, L. C. Davison, and P. F. Roush, "On-line Storage and Retrieval of Chemical Information. I. Structure Entry", *J. Chem. Inf. Comput. Sci.*, **19**, 228-230 (1979).
- (15) W. T. Wipke, G. I. Ouchi, and S. Krishnan, "Simulation and Evaluation of Chemical Synthesis—SECS. An Application of Artificial Intelligence Techniques", *Artif. Intelligence*, **11**, 173-193 (1978).
- (16) H. L. Gelernter, A. F. Sanders, D. L. Larsen, K. K. Agarwal, R. H. Boivie, G. A. Spritzer, and J. E. Searleman, "Empirical Explorations of SYNCHEM", *Science (Washington, DC)*, **197**, 1041-1049 (1977).
- (17) H. W. Whitlock, "An Organic Chemist's View of Formal Languages", *ACS Symp. Ser.*, No. 61, 1977.
- (18) H. W. Whitlock, "A Heuristic Solution to the Functional Group Switching Problem in Organic Synthesis", *J. Am. Chem. Soc.*, **98**, 3225-3233 (1976).
- (19) P. E. Blower and H. W. Whitlock, "An Application of Artificial Intelligence to Organic Synthesis", *J. Am. Chem. Soc.*, **98**, 1499-1510 (1976).
- (20) E. V. Krishnamurthy and M. F. Lynch, "Coding and Analysis of Generic Chemical Formulae in Chemical Patents", *J. Inf. Sci.*, in press.
- (21) M. F. Lynch, J. M. Barnard, and S. M. Welford, "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 1. Introduction and General Strategy", *J. Chem. Inf. Comput. Sci.*, paper 1 in this series.
- (22) J. M. Barnard, M. F. Lynch, and S. M. Welford, "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 2. GENSAL, a Formal Language for the Description of Generic Chemical Structures", *J. Chem. Inf. Comput. Sci.*, preceding paper in this issue.
- (23) P. Naur, Ed., "Report on the Algorithmic Language ALGOL 60", *Commun. ACM*, **3**, 299-314 (1960).
- (24) A. V. Aho and J. D. Ullman, "The Theory of Parsing, Translating and Compiling", Prentice-Hall, Englewood Cliffs, NJ, 1972.
- (25) S. A. Greibach, "Inverses of Phrase Structure Generators", Ph.D. Thesis, Harvard University, Cambridge, MA, 1963.
- (26) J. E. Ash, "Connection Tables and Their Role in a System" in "Chemical Information Systems", J. E. Ash and E. Hyde, Eds., Ellis Horwood, London, 1975.
- (27) H. Hiz, "A Linearization of Chemical Graphs", *J. Chem. Doc.*, **4**, 173-180 (1964).
- (28) S. H. Eisman, "A Polish-type Notation for Chemical Structures", *J. Chem. Doc.*, **4**, 186-190 (1964).
- (29) A corollary of this rule is that terminal atoms which are connected directly to a branching atom are specified as late as possible, consistent with the requirements of a Polish notation.
- (30) H. I. Scoins, "Placing Trees in Lexicographic Order", *Mach. Intelligence*, **3**, 43-60 (1968).
- (31) D. J. Cohen and C. C. Gotlieb, "A List Structure Form of Grammars for Syntactic Analysis", *Computing Surveys*, **2**, 65-82 (1970).
- (32) J. Lederberg, G. L. Sutherland, B. G. Buchanan, E. A. Feigenbaum, A. V. Robertson, A. M. Duffield, and C. Djerassi, "Applications of Artificial Intelligence for Chemical Inference. I. The Number of Possible Organic Compounds. Acyclic Structures Containing C, H, O and N", *J. Am. Chem. Soc.*, **91**, 2973-2976 (1969), and subsequent publications.
- (33) L. Masinter, N. S. Sridharan, J. Lederberg, and D. H. Smith, "Applications of Artificial Intelligence for Chemical Inference. XII. Exhaustive Generation of Cyclic and Acyclic Isomers", *J. Am. Chem. Soc.*, **96**, 7702-7714 (1974).
- (34) Y. Kudo and S.-I. Sasaki, "Principle for Exhaustive Enumeration of Unique Structures Consistent with Structural Information", *J. Chem. Inf. Comput. Sci.*, **16**, 43-49 (1976).
- (35) C. A. Shelley, T. R. Hays, M. E. Munk, and R. V. Roman, "An Approach to Automated Partial Structure Expansion", *Anal. Chim. Acta*, **103**, 121-132 (1978).
- (36) J. Earley, "An Efficient Context-Free Parsing Algorithm", *Commun. ACM*, **13**, 94-102 (1970).
- (37) The acyclic grammar is described as deterministic. This is a consequence, in part, of the fact that each terminal symbol appears in only a single replacement rule in the grammar. The appropriate replacement rule is therefore readily located.
- (38) G. W. Adamson, J. Cowell, M. F. Lynch, A. H. W. McLure, W. G. Town, and A. M. Yapp, "Strategic Considerations in the Design of a Screening System for Substructure Searches of Chemical Structure Files", *J. Chem. Doc.*, **13**, 153-157 (1973).
- (39) G. W. Adamson, J. A. Bush, A. H. W. McLure, and M. F. Lynch, "An Evaluation of a Substructure Search System Based on Bond-Centered Fragments", *J. Chem. Doc.*, **14**, 44-48 (1974).

Specification and Unconstrained Enumeration of Conformations of Chemical Structures for Computer-Assisted Structure Elucidation

JAMES G. NOURSE

Department of Chemistry, Stanford University, Stanford, California 94305

Received January 22, 1981

A symmetry group called the conformation symmetry group (CFSG) which provides a method of uniquely specifying molecular conformation based on an appropriate discrete bond property (such as rotameric state) is formulated. The method is applicable to entire chemical structures as well as substructures. The CFSG can be used to build a simple "acyclic" conformation generator and leads to a solution for a heretofore unsolved problem in conformation enumeration.

For a number of purposes, including computer representation, the specification of an organic chemical structure starting from its molecular formula can be done in several stages. In the CONGEN program (CONstrained GENeration of isomers) for computer-assisted structure elucidation, we have chosen to do this chemical structure specification in three stages. Starting from the molecular formula, the first stage is to specify the constitution or atom-to-atom connectivity of the structure. The CONGEN program is capable of generating all the possible constitutional isomers for any molecular formula.¹ Furthermore this generation can be constrained to

just those possibilities consistent with partial substructural information.²

The second stage is to specify the configuration of stereocenters (chiral centers and double bonds) in the structure to give the configurational stereoisomer. The CONGEN program is capable of generating all the possible configurational stereoisomers for any molecular formula³ and this generation can also be constrained to just those possibilities consistent with partial stereochemical information.⁴ The third stage is to specify the conformation (i.e., position of torsional rotation about single bonds) of the structure, and the approach to this