

- (15) Dias, J. R. *Acc. Chem. Res.* **1985**, *18*, 241.
 (16) Dias, J. R. *J. Macromol. Sci., Chem.* **1985**, *A22*, 335.
 (17) Dias, J. R. *Nouv. J. Chim.* **1985**, *9*, 125.
 (18) Dias, J. R. *THEOCHEM* **1986**, *137*, 9.
 (19) Cyvin, S. J. *MATCH* **1986**, *20*, 165.
 (20) Cyvin, S. J.; Gutman, I. *Z. Naturforsch., A: Phys., Phys. Chem., Kosmophys.* **1986**, *41A*, 1079.
 (21) Cyvin, B. N.; Brunvoll, J.; Cyvin, S. J.; Gutman, I. *MATCH* **1986**, *21*, 301.
 (22) Cyvin, S. J.; Brunvoll, J.; Cyvin, B. N. *Z. Naturforsch., A: Phys., Phys. Chem., Kosmophys.* **1986**, *41A*, 1429.
 (23) Cyvin, S. J.; Cyvin, B. N.; Gutman, I. *Z. Naturforsch., A: Phys., Phys. Chem., Kosmophys.* **1985**, *40A*, 1253.
 (24) Cyvin, S. J.; Gutman, I. *THEOCHEM* **1987**, *150*, 157.
 (25) In the previous work⁸ it was overlooked that the branched catacondensed benzenoids in fact also can be generated specifically: Start from triphenylene and apply additions of hexagons in the position [1], i.e., so that one added hexagon shares exactly one edge with the original benzenoid.
 (26) Gutman, I. *Croat. Chem. Acta* **1974**, *46*, 209.
 (27) Cyvin, S. J.; Gutman, I. *Comp. Maths. Appl.* **1986**, *12B*, 859.
 (28) Balaban, A. T.; Harary, F. *Tetrahedron* **1968**, *24*, 2505.
 (29) Hosoya, H. *Comp. Maths. Appl.* **1986**, *12B*, 271.
 (30) Cyvin, S. J.; Bergan, J. L.; Cyvin, B. N. *Acta Chim. Hung.* (in press).
 (31) Brown, R. L. *J. Comput. Chem.* **1983**, *4*, 556.
 (32) Cyvin, S. J.; Gutman, I. *J. Serb. Chem. Soc.* **1985**, *50*, 443.

Computer Storage and Retrieval of Generic Chemical Structures Using Structure Attributes

SOICHI TOKIZANE* and TSUKASA MONJOH

Japan Association for International Chemical Information, Bunkyo-Ku, Tokyo, 113 Japan

HIDEAKI CHIHARA

Faculty of Sciences, Osaka University, Toyonaka, 560 Japan

Received November 17, 1986

A method to store and search chemical structure information with generic expressions using digital computers was developed. The method uses specially designed sets of attributes to represent chemical characteristics of both generic and specific expressions. Three types of attributes, ring, chain, and atom attributes, are defined and described. Attributes are expressed as bit-map vectors, which makes them easy to handle by digital computers. A search algorithm and procedure using a modified connectivity stack method to find the match between a query structure and a file structure with generic expressions are described.

INTRODUCTION

Chemical structure searching is a most sophisticated type of information retrieval using computers. One of the biggest and most comprehensive chemical structure search systems, the CAS ONLINE Registry File, is now available commercially worldwide from Chemical Abstracts Service, a division of the American Chemical Society. The Registry File is searchable online by creating a query structure interactively on a graphic terminal and then conducting a search of the file of over 8 000 000 chemical substances collected from journal and patent references since the 1960s. The answer structures can be displayed on the same graphical terminal. Any current implementation of the structure search system can handle so-called specific chemical structures only.

Generic expression of chemical structure has been widely used since the early days to describe a broad range of chemical structures with a concise drawing of the structure containing variables and class expressions. Generic chemical structures are most extensively used in patent literatures, where it is natural to express the scope of their invention as widely as possible, and the use of generic structures (or Markush structures after the name of an American inventor) is generally accepted as a rule. The current status of the handling of generic chemical structures is discussed thoroughly in recent conference proceedings.¹ Several papers were published describing basic concepts and techniques to handle generic chemical structures on a computer. Lynch and his co-workers^{2-8,10,11} and Scholley⁹ have been working on this topic extensively. A method of handling generic chemical structures was also proposed by Kudo and Chihara.¹²

A typical generic (Markush) chemical structure is shown in Figure 1. There are several characteristics that are used in generic (Markush) chemical structures as seen in Figure 1. They are (1) alternative structure components (as in R1), (2) generic expressions (as in R2), (3) nonfixed attachments

Table I. Types of Chemical Structure Matching by Fisanick

	query structure	file structure
overlap matching criteria (1)	full	specific
overlap matching criteria (2)	full	generic
overlap matching criteria (3)	Markush	specific
overlap matching criteria (4)	Markush	generic
embedment matching criteria (1)	substructure	specific
embedment matching criteria (2)	substructure	generic
embedment/overlap matching criteria	substructure	generic

(as in R3), (4) undefined structure components (as in R3), (5) variable counts (as in R1), and (6) ring node variables (as in X).

Variable structure definitions such as 1, 3, 4, and 5 can be handled without much difficulty if enough computing power is available. Existing systems are already able to handle query structures containing variable components. It is thus possible to extend this capability to handle file structures with variable parts. One approach to this topic is the Colored Complete Graph method described by Kudo and Chihara.¹² On the other hand, storing and retrieving chemical structures with generic groups as seen in the group R2 is a very difficult topic in principle. We will focus on how to store and search structures containing generic groups in this paper.

A generic group may be described by the following components: (1) specific or range of atom count, (2) specific or range of element count, (3) specific or range of element types, (4) specific or range of bond composition (usually other than single bonds), (5) specific or range of branches or substitutions, (6) specific or range of ring size, (7) specific or range of number of component rings, and (8) other textural information.

The above expressions may appear either independently or combined. An example of the generic expression is shown in Figure 2. Numbers in brackets correspond to the group numbers specified above. Unknown data are regarded as

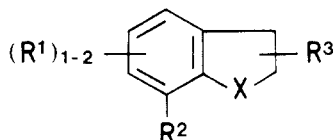


Figure 1. Example of generic (Markush) chemical structure. R1 = Cl, Br; R2 = C₃₋₁₀ alkyl or alkenyl; R3 = any substituent; X = N, S.

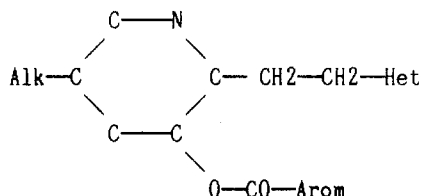


Figure 2. Example of generic structure expression. Numbers in brackets refer to the generic expression category numbers in the text. Alk = alkyl groups with C₁₋₅ chain [2,5]. Arom = aromatic rings (i.e., benzene or naphthalene) [4]. Het = heterocyclic (i.e., rings containing one or more non-carbon atoms) groups with ring size of 5 or 6 [3, 6].

having the widest range. For example, an unknown ring size means allowing any ring size defined in the particular system.

"Other textural information" is such information that does not specify any structural identity. "Electron-withdrawing group" is an example of this type. This categorization is naturally arbitrary and affected by system design.

Detailed discussion of types of Markush and generic groups is done by Lynch's group.² They also proposed a very useful coding system called GENSAL in the same paper in describing generic structures.

Various requirements of generic structure searching are thoroughly discussed by Fisanick.¹³ He has defined several types of matching of chemical structures as summarized in Table I. Here overlap matching is a one-to-one match between a query structure and a file structure. Embedment matching is a so-called substructure searching widely used in current chemical structure retrieval systems. Thus the first, third, and fifth criteria are already achievable in current systems to some extent, although the range of Markush queries has limitations. Please note Fisanick used the term generic expression with almost the same meaning as Markush expression, i.e., having variable, repeating, and generic structure groups.

One type of matching Fisanick did not address was superstructure matching. This matches a file substructure embedded in a query structure. Since this is the reverse of substructure searching, we will not discuss it in detail.

In summary, the system we are going to develop has to process the following types of search requests: (1) searches of specific structure files by specific structure, (2) searches of specific structure files by generic structure, (3) searches of generic structure files by specific structure, and (4) searches of generic structure files by generic structure. All of these searches include exact structure, substructure, and superstructure searching.

METHODOLOGY

We define a generic group as a single chemical unit with nonspecific chemistry. A group with multiple definitions of specific chemistry will be called a variable group. In this paper we will concentrate on how to handle nonspecific information of chemical structure.

A generic group may contain a ring, multiple rings fused together, or a straight or branched chain. A substituent to a ring is considered a separate entity, and thus is not included in the ring generic group, although substitution to a chain by

atoms of the same element type is considered a branch of the chain generic group. This definition is further discussed under Differentiation of Rings and Chains.

Our method can handle substructure searching as well as exact matching. Both the query and file structures can be either specific or generic. Our system uses a sort of connection table to represent chemical structures. In addition to specific atoms (nodes) to describe specific structures, we use generic atoms (nodes) to describe generic structures. For traditional specific structures, a one-to-one correspondence between query atoms and the bonds connecting them and file atoms and bonds is examined.

For structures containing generic parts, this requirement needs to be modified. There will be no more one-to-one correspondence. Two or more atoms of one structure may correspond with a generic component (node) of the other structure. This generic component is composed of one or more atoms and the bonds connecting them. Still, for an exact match, there should be no atoms or generic components that do not correspond with atoms or generic components of the other structure. Thus, all atoms of a pyridine ring match a generic node, a heterocycle, and the generic node is satisfied with the assembly of six atoms of the pyridine ring. Similarly, the generic node heterocycle is satisfied with a pyridine ring of methylpyridine, but its methyl radical does not correspond with this generic node.

Much more complex is finding matches between different expressions containing chemically homologous groups. For example, an alkyl chain of 3-5 carbons satisfies a query containing an alkyl chain of 3 or 4 carbons or even a query with an alkyl chain of 4-10 carbons, because the query is an OR combination of individual structures having 4-10 carbons, respectively. Another example is matching rings. A six-membered heterocyclic ring with no bond specificity will satisfy a query structure of an aromatic ring containing one or two nitrogens.

The above requirement will be fulfilled by defining a list of chemical characteristics, which we call an attribute. Each generic chemical group is assigned a set of attributes, each expressing the presence or absence of a specific chemical characteristic of a specific or generic atom. A similar idea is Lynch's parameter list.⁴ Our attribute is unique in that it is expressed by a bit-map vector rather than a set of identifiers. Here the value 1 means the presence of a specific characteristic assigned to the position in the vector, and 0 means the absence of the same. An attribute is also unique in that it is assigned to specific atoms as well as generic atoms to allow full comparison of both atoms.

In a very simplified example below, an attribute vector that consists of only two bits may be defined for each ring atom as in Figure 3. Thus the HY atom of the query structure will be found in the file structure because the attribute of node 4 of the file structure matches with the attribute of HY. The condition of the match is that whenever the value of a column of the query attribute vector is 1, the corresponding column of the file attribute vector should be 1. Otherwise, any value will do for the file attribute columns.

In another example shown in Figure 4, all the atoms in the heterocycle of the query structure match with the single generic node 5, which is illustrated as HY. To allow this type of comparison of ring moiety, it is necessary to differentiate rings from chains clearly as described later.

In a more complex example in Figure 5, we consider a 2-4 carbon chain to match exactly with a 3-5 carbon chain, simply because there are overlapping conditions, a 3 or 4 carbon chain.

To make this type of match detectable, the attribute vector should contain columns assigned to each value of carbon count. Table II shows how an attribute vector is defined for a

FILE STRUCTURE	QUERY STRUCTURE
<p>STRUCTURE</p> <p>1 2 3 4 5 6 Ph-C-C-C-C-C-C</p> <p>ATTRIBUTE 4 = Heterocyclic</p>	<p>STRUCTURE</p> <p>a b c C-C-HY</p> <p>ATTRIBUTE HY = Heterocyclic</p>
ATTRIBUTES	
Carbocyclic Node (Bit one)	Heterocyclic Node (Bit two)
File Node	
1	0
4	1
5	1
6	1
7	1
8	1
9	1
Query Node	
c	1

Figure 3. Simple example of attributes.

FILE STRUCTURE	QUERY STRUCTURE
<p>STRUCTURE</p> <p>1 2 3 4 5 CB-C-C-C-C-HY</p> <p>ATTRIBUTE CB = Carbocyclic HY = Heterocyclic</p>	<p>STRUCTURE</p> <p>a b c d e f C-C-C-C-C-C</p> <p>Node b, c, d, e, f, and g = Heterocyclic</p>

Figure 4. How to match query and file structures with generic expressions.

FILE STRUCTURE	QUERY STRUCTURE
<p>STRUCTURE</p> <p>1 2 3 PH-(CH2)-OH</p> <p>■ = 3-5</p> <p>ATTRIBUTE 2 = 3-5 carbon chain</p>	<p>STRUCTURE</p> <p>a b c PH-(C)-O</p> <p>n = 2-4</p> <p>ATTRIBUTE b = 2-4 carbon chain</p>

Figure 5. Partial matching of generic expression.

structure of a 3-5 carbon chain as both file structure and query structure. Here the value 1 designates the presence of a specific characteristic described, and 0 designates its absence. For file structures, all the bit positions corresponding to any conditions possible in accordance with varying values of n are filled by the value 1. On the other hand, for queries, only the bit positions corresponding to the conditions that are always met regardless of which possible value of m is taken are filled by the value 1.

For a specific structure, e.g., a chain of 3 carbons, the attributes are easily assigned. Those bits specifying chain length of one or more, two or more, three or more, three or less, four or less, and five or less are filled by bit one. Similar attributes will be assigned for C_4 and C_5 chains. If a variable structure C_{3-5} is a file structure, the attribute vectors of each component structure are ORed, which means satisfaction of

Table II. Definition of Attribute Vector of a Chain with Variable Length

bit position	chain length	C_2	C_3	C_4	C_5	file str (OR)	query str (AND)
1	1 or more	1	1	1	1	1	1
2	2 or more	1	1	1	1	1	1
3	3 or more	0	1	1	1	1	0
4	4 or more	0	0	1	1	1	0
5	5 or more	0	0	0	1	1	0
6	1	0	0	0	0	0	0
7	2 or less	1	0	0	0	0	0
8	3 or less	1	1	0	0	1	0
9	4 or less	1	1	1	0	1	1
10	5 or less	1	1	1	1	1	1

FILE STRUCTURE	QUERY STRUCTURE
ATTRIBUTE	If g, h, and i can be either a part of a ring or a part of a chain, then no chain attributes are generated.

Figure 6. Condition where chain attributes are not generated.

a query characteristic by any one of the values will constitute a hit. On the other hand, if a similar variable structure C_{2-4} is a query structure, the attribute vectors are ANDed, which then means satisfaction of only commonly found characteristics of the query by a file structure is sufficient for a hit. This treatment is very important when both query structures and file structures have generic nodes and there is a possibility of so-called partial match. Thus, a file node of C_{3-5} chain, which has a chain attribute vector of 11111 00111, will satisfy a query node of C_{2-4} chain, whose chain attribute vector is 11000 00011 from the attribute definition of Table II. On the other hand, a query node of C_1 , whose vector is 10000 11111, will not satisfy the above file vector. This query vector requires $C_{\leq 1}$ (sixth bit is 1), but the query vector indicates $C_{\leq 2}$ is not possible (sixth and seventh bits are 0).

The use of twin bit strings to represent one chemical aspect (C_3 is represented by the third and eighth bits in the above example) somewhat resembles the twin bit string adapted by Welford et al.⁷ The fundamental difference is that while they use in the twin bit string method to distinguish essential fragments (MUST) and variable fragments (MAY) of a generic structure, we use it to match characteristic ranges of file and query structures.

TYPES OF ATTRIBUTES

Differentiation of Rings and Chains. It is necessary to first differentiate rings from chains or vice versa. It is theoretically possible to define attributes that cover both rings and chains at the same time, but, in practice, it is much more efficient to define two types of attributes separately. If a ring is drawn there, it is a ring. But when a chain is drawn in a query or file structure, the resolution is not so simple. There are chances that the chain is part of one or multiple unknown rings. In many cases one can divide the original structure into two structures, one where the chain is a pure chain and one where it is part of a ring (Figure 6).

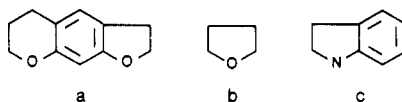
When it is not possible for a part of the structure to be fragmented into rings and chains, e.g., a C_{6-12} group possibly containing a ring, again atom attributes are the only way of matching generic parts. In such a case where a query or file structure contains a couple of substituents to a ring, which may form an additional fused ring, the structure can be divided in

Table III. Examples of Ring Attributes

bit position	description	F1 ^a	F2 ^b	F3 ^c	Q1 ^d	Q2 ^e	Q3 ^f
	element type (may be present)						
1	heteroatom	1	1	1	1	1	1
2	oxygen	1	0	1	1	0	0
3	sulfur	1	0	0	0	0	0
4	nitrogen	1	1	0	0	1	0
5	other element	1	0	0	0	0	0
	element type (may not be present)						
6	heteroatom	0	0	0	0	0	0
7	oxygen	1	1	0	0	1	0
8	sulfur	0	1	1	1	1	0
9	nitrogen	1	0	1	1	0	1
10	other element	1	1	1	1	1	1
	no. of component rings (minimum)						
11	1	1	1	1	1	1	1
12	2	0	1	1	0	1	0
13	3	0	1	1	0	0	0
14	4	0	1	0	0	0	0
	no. of component rings (maximum)						
15	1	1	1	0	0	0	1
16	2	1	1	0	0	1	1
17	3	1	1	1	0	1	1
18	4	1	1	1	0	1	1
	size of rings (may be present)						
19	3	0	1	0	0	0	0
20	4	0	1	0	0	0	0
21	5	1	1	1	1	1	0
22	5 (2 or more)	0	1	0	0	0	0
23	6	1	1	1	0	1	1
24	6 (2 or more)	0	1	1	0	0	0
25	6 (3 or more)	0	1	0	0	0	0
26	7	0	1	0	0	0	0
27	8 or 9	0	1	1	0	1	0
28	10 or more	0	1	1	0	0	0
	size of rings (may not be present)						
29	3	1	1	1	0	1	1
30	4	1	1	1	0	1	1
31	5	1	1	0	0	0	1
32	5 (2 or more)	1	1	1	0	1	1
33	6	1	0	0	0	0	0
34	6 (2 or more)	1	1	0	0	1	1
35	6 (3 or more)	1	1	1	0	1	1
36	7	1	1	1	0	1	1
37	8 or 9	1	1	0	0	0	1
38	10 or more	1	1	0	0	1	1
	one or more aromatic rings						
39	yes	1	1	1	0	1	0
40	no	1	0	1	0	0	0

^a Five- or six-membered ring containing sulfur. No fusion. Other elements are also allowed. ^b Pyridine ring with possible fusion with carbon rings.

^c Structure a, below. ^d Tetrahydrofuran (fusion with carbon rings allowed) (structure b, below). ^e 2,3-Dihydroindole (no fusion allowed) (structure c, below). ^f Six-membered ring with one sulfur or one oxygen.



two, one containing chain substituents and the other containing a fused ring.

Ring Attributes. A ring system may be described by several characteristics.

(1) Type of elements: Ring systems containing elements other than carbon are usually called heterocycles.

(2) Number of atoms: This is the total number of atoms of a ring system, which may be composed of several component rings.

(3) Number of component rings: The number of component rings is calculated by the equation

$$(\text{no. of rings}) = (\text{no. of bonds}) - (\text{no. of atoms}) + 1$$

This is the number of cuts required to convert the ring system into a single open chain and is also the minimum number of component rings required to describe a ring system in most cases, except some rings like cubane.

(4) Size of component rings: The sizes of all the rings embedded in a ring system should be described here, not only the smallest set of smallest rings.¹⁴ This is necessary to assure complete retrieval of all relevant structures.

(5) Aromaticity: The definition of aromaticity can be different from person to person. CAS uses ring alternating bonds instead of aromatic bonds in creating records for their Registry System.¹⁵ Here we simply consider aromaticity as $(6 + 4 \times n)$ -membered rings with alternating bonds, as well as those defined as aromatic by the authors.

Table III shows an example of ring attributes. F1–F3 represent file structures, and Q1–Q3 represent query structures. Positions 1–5, 11–14, 19–28, and 39 identify the presence of certain characteristics, and positions 6–10, 15–18, 29–38, and 40 identify the absence of them. More specifically, for a file structure the bit value 1 at a “may be present” position means the structure “may” have the corresponding

chemical characteristic, and the bit value 1 at a "may not be present" position means the structure "may" lack it. For a query structure, on the other hand, the bit value 1 at a "may be present" position specifies the characteristic "should" be present, and the bit value 1 at a "may not be present" position requires it "should" not be present.

Heteroatoms in positions 1 and 6 indicate the presence (or absence) of any non-carbon atoms. "Other element" in positions 5 and 10 corresponds to any non-carbon atoms other than O, S, or N. It will certainly be useful to include the number of specific or all heteroatoms here in addition; it will improve the relevancy of the retrieval.

When looking at the size of a ring, one should consider all possibilities of envelope and other component rings. Thus the ring F3 contains 9-, 10-, and 13-membered rings as component rings.

As shown in Table III, the technique described above is used to define attributes. For example, when the file structure attributes are created, all the possibilities are ORed, and when the query structure attributes are created, they are ANDed. Thus a file ring containing a sulfur atom and no other heteroatoms has a vector 10100 01011, from positions 1 to 10, and a file ring containing a sulfur and an oxygen atom will have a vector 11100 00011. A file ring with two or three component rings will have a vector 1110 0111, corresponding to positions 11-18. The result of the OR operation (F1) is that the positions specifying "heteroatom may not be present" and "sulfur may not be present" are left 0 with all other bits filled with 1. On the other hand, a query ring containing a sulfur atom will have a vector 10100 01011 in the same positions, and a query ring containing an oxygen atom will have a vector 11000 00111. Similarly, a query ring with two or three component rings (no other fusion) will have a vector 1100 0011, corresponding to positions 11-18. The attribute vector of Q3 is a result of the AND operation and thus has a vector 10000 00011.

From attribute matching, the attribute vector of Q1 is satisfied by that of F3; i.e., the value of any position where the value is 1 in the Q1 vector is also 1 in the F3 vector. For example, from positions 11 to 18, the vector of Q1 is 1000 0000 and that of F3 is 1110 0011. Also the attribute vector of Q2 is satisfied by that of F2, which is a false hit. Both structures have a nitrogen, a six-membered ring, and aromatic bonds. But the two are indeed unmatched structures. If there is a connection table of the pyridine ring, an atom-by-atom search will eliminate this false retrieval. In order to avoid this type of false coordination, the ring attributes should carry more information, such as five-membered ring nitrogen and six-membered ring nitrogen. But there is always some chance of false coordination in generic structure searching at the attribute level. Finally, Q3 matches with F1 when attributes are compared. The structure F1 clearly allows a six-membered sulfur-containing ring. The positions specifying any types of heteroatoms (positions 1 and 6) are necessary, because otherwise the query structure Q3 will match with any six-membered carbon rings. It would be better to have such attributes specifying the number of heteroatoms if the size of the attribute vector allows.

Thus, any combination of specific rings, generic rings, and ring substructures will be successfully compared by using this type of ring attribute. The point is that the ring attribute is very powerful in matching generic expressions such as F1 and Q3, and at the same time it is also powerful in screening unwanted structures quickly, as seen in matching Q1 and F3.

In F2 of the above example, at least a part of the ring is specifically described. Then it is possible for the system to have supplementary connection tables for such substructures. Thus, after the attribute matching is successfully achieved,

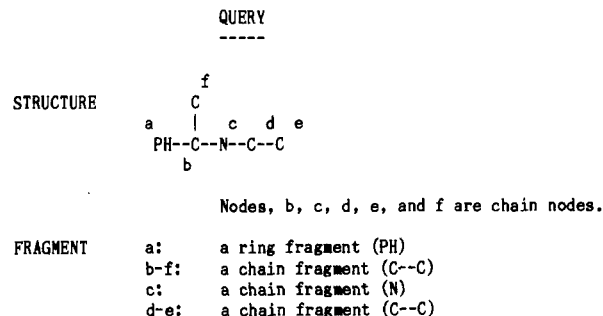


Figure 7. Example of chain structure with multiple elements.

the nodes of the query structure that match with the particular generic ring will be further examined to match atom-by-atom with the supplementary connection table.

Chain Attributes. Segmentation of a Chain. First, it is important to note that chain attributes are created only when the atom group is explicitly defined as a nonring group. To make the system simpler, it is advisable to segment a chain structure into subchains made of a single element type. A generic node is considered to be a sort of non-carbon atom for this purpose. An example is shown in Figure 7.

Thus ambiguous expressions such as a chain containing silicon in between may not be allowed. Such expressions should be handled by using atom attributes, but for file structures only.

Also the following chain is considered to have 4-6 carbons, rather than 3-5:



Note that acetone is considered to be a C₃ chain here with an oxygen connected to it.

Types of Chain Attributes. The characteristics of chains are classified into several groups.

(1) Type of elements: It is assumed only one type of element appears in a chain defined here.

(2) Number of atoms.

(3) Presence of substitutions: If more than two atoms of different element types are connected to a particular chain, it is defined to be substituted.

(4) Presence of branches: This specifies branching within a chain.

(5) Presence of double bonds.

(6) Presence of triple bonds.

An example of a chain attribute vector is shown in Table IV.

The procedure to generate the above attributes is quite similar to that applied to ring attributes. Substitution and branching are always possible when there are more than two connections from a chain. When there are more than two connections and all the connecting atoms (whose element type should be other than that of the chain) are defined, then no branching is possible. Since Q1 has one open bond at the central carbon, branching is possible, although not necessary.

In attribute matching, Q1 will match with F2, although the match will not be found in atom-by-atom searching. The chain Q2 will match with both F1 and F2. Since atom-by-atom matching is not possible for Q2, this result is final.

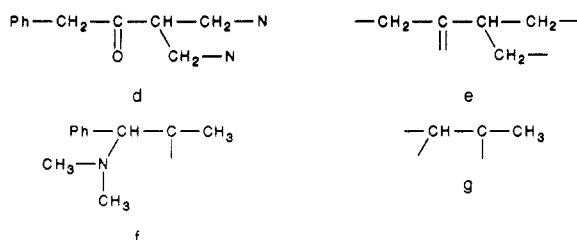
Types of Atom Attributes. It is also useful to define atom attributes for such a case where one or more atoms in a given query or a file structure are not specified as part of either a ring or a chain. Thus atom attributes describe the types of elements and connecting bonds around each atom in a specific structure as well as each implied atom in generic groups. An example is shown in Table V.

It should be noted that these attributes look into an atom environment only. In a generic node, this attribute vector is

Table IV. Examples of Chain Attributes

bit position	description	F1 ^a	F2 ^b	Q1 ^c	Q2 ^d
element type					
1	carbon	1	1	1	1
2	oxygen	0	0	0	0
3	sulfur	0	0	0	0
4	nitrogen	0	0	0	0
5	other elements	0	0	0	0
no. of atoms					
6	1 or more	1	1	1	1
7	2 or more	1	1	1	1
8	3 or more	1	1	1	1
9	4 or more	1	1	0	1
10	5 or more	1	1	0	0
11	6 or more	1	0	0	0
12	8 or more	1	0	0	0
13	12 or more	0	0	0	0
any substitutions					
14	1	0	0	0	0
15	2 or less	0	0	0	0
16	3 or less	1	0	0	0
17	4 or less	1	0	0	0
18	5 or less	1	1	0	1
19	6 or less	1	1	0	1
20	8 or less	1	1	0	1
21	12 or less	1	1	0	1
any branches					
22	yes	0	1	1	0
23	no	1	0	0	0
any double bonds					
24	yes	0	1	0	0
25	no	1	0	0	0
any triple bonds					
26	yes	1	0	0	0
27	no	1	1	1	0
any triple bonds					
28	yes	1	0	0	0
29	no	1	1	1	0

^aStraight terminal chain with 3–10 carbons. ^bStructures d and e, below. ^cStructures f and g, below. ^dChain of 4 or 5 carbons.



an OR result of all atoms that may be present in the generic group (node). Here an atom in an file aromatic ring (F2) is considered to have a single bond, a double bond, and aromatic bonds to allow broadest retrieval. Although Q1 has an explicit single and double bond only, it can match with F2 because of this treatment. Q1 also matches with F3, and Q2 matches with F1, F2, and F3. An (carbon) atom of Q2 will match with a carbon atom of F2.

MATCHING METHOD

As we have already discussed, there are three types of matches, namely, exact, substructure, and superstructure matches. Since the superstructure match is a mirror image of the substructure match and the exact match is a special case of the substructure match, we will concentrate mainly on the substructure match.

We chose here substructure searching, where a match is successful when a given query structure is logically involved or embedded in a candidate structure. Thus, an examination is over either when all query nodes are matched with corresponding file nodes or when it is found such query–file node correspondence does not exist. All the query nodes should be

Table V. Examples of Atom Attributes

bit position	description	F1 ^a	F2 ^b	F3 ^c	Q1 ^d	Q2 ^e
element type						
1	carbon	1	1	1	0	1
2	oxygen	0	1	0	0	0
3	sulfur	0	1	0	0	0
4	nitrogen	0	1	1	1	0
5	other element	0	1	0	0	0
bond type ^f (may be present)						
6	S (more than 1)	1	1	1	1	0
7	S (more than 2)	1	0	1	0	0
8	S (more than 3)	0	0	1	0	0
9	S (more than 4)	0	0	1	0	0
10	D	1	1	1	1	0
11	T	0	0	1	0	0
12	ring	0	1	1	0	0
13	aromatic ring	0	1	1	0	0
bond type ^f (may not be present)						
14	S (more than 1)	0	0	1	0	0
15	S (more than 2)	0	1	1	0	0
16	S (more than 3)	1	1	1	0	0
17	S (more than 4)	1	1	1	0	0
18	D	0	0	1	0	0
19	T	1	1	1	1	0
20	ring	1	0	1	0	0
21	aromatic ring	1	0	1	0	0

^aStraight carbon chain with possible double bonds. ^bAromatic ring with N. ^cAny carbon atom group containing nitrogen. ^d—N=. ^eAny carbon atom group. ^fS, single bond; D, double bond; T, triple bond.

examined for matches, but not all the file nodes, as long as the above condition is fulfilled.

There are several methods to find the presence of a substructure match or to search substructures. Two widely known methods are the iterative, or atom-by-atom searching, and the set reduction methods.^{16,17} Attributes will be applicable to either method. We chose the connectivity stack method, a derivative of the iterative searching, to demonstrate how to conduct generic structure searching.

The connectivity stack method was first proposed by Kudo and Sasaki to enumerate possible structural formulas of the same molecular composition.^{18,19} The connectivity stack method was used there to generate all possible bonding combinations of a group of atoms (superatoms) of a certain molecular formula and thus enumerate all possible isomers of the molecular composition. The same method may be used to generate all possible numbering combinations of atoms instead. When the node number vector of a query structure matches with the vector produced from one of the numbering combinations of a file structure by comparing the element types and connectivity information of each column of both vectors, then there exists a substructure match.

In a very simple example in Figure 8, a match will be found in the sixth numbering combination (vector) of the file structure. The third vector does not match with the query vector because there is no bond between node 1 and node 4, as required in matching the corresponding vector elements of the query. Here the vectors are generated by permutation of the first vector. Permutation happens at the first column from the left where no match is found. The vector element at the column is exchanged with the element to the right of it that has a node number greater than its node number and examined for matches. When no further permutation is possible at the column, i.e., no element to the right of it has a greater node number, then the element is placed further right. The next move is to begin permutation at the next column to the left.

In the generic structure searching, it is necessary to examine not only the element type and connectivity information but also the attribute information for each column of vectors. In addition, special treatment of numbering as described later is necessary to handle generic nodes that may consist of

FILE STRUCTURE						QUERY STRUCTURE					
-----						-----					
STRUCTURE											
1	2	3	4	5		1	2	3			
C	N	C	O	C		N	C	O			
Query Structure vector											
(Column Number)											
	1	2	3	4	5	RESULT					
-----						-----					
Element Composition											
File Structure vectors											
(Element Composition)											
Vector 1	1(C)*	2(N)	3(C)	4(O)	5(C)	failed					
Vector 2	2(N)	1(C)	3(C)*	4(O)	5(C)	failed					
Vector 3	2(N)	1(C)	4(O)*	3(C)	5(C)	failed					
Vector 4	2(N)	1(C)	5(C)*	3(C)	4(O)	failed					
Vector 5	2(N)	1(C)*	3(C)	4(O)	5(C)	failed					
Vector 6	2(N)	3(C)	1(C)*	4(O)	5(C)	failed					
Vector 7	2(N)	3(C)	4(O)	1(C)	5(C)	matched					

* Point of permutation

Figure 8. Matching of chemical structures by the connectivity stack method.

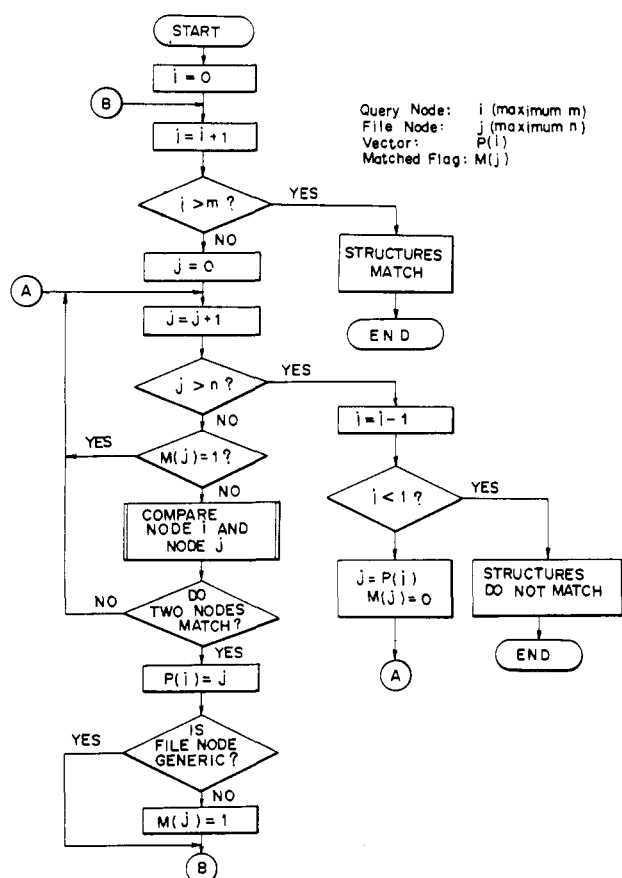


Figure 9. Flow chart of matching query and file structures.

multiple-atom groups rather than single atoms.

FLOW CHART OF SEARCH PROCESS

A specific search process conducted on the basis of the modified connectivity stack method is illustrated in Figure 9. Here nodes of the query structure (hereafter called query nodes) are numbered by i . By definition, $0 < i \leq m$, where m is the count of the query nodes. Similarly, nodes of a candidate structure in the file (hereafter called file nodes) are numbered by j . Also by definition, $0 < j \leq n$, where n is the count of the file nodes. A vector $P(i)$ is defined to store the

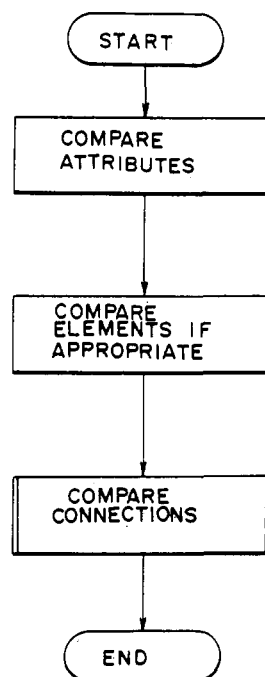


Figure 10. Flow chart of matching query and file nodes.

number of a file node that matched with the query node i . Another vector $M(j)$ is defined to mark whether the file node j is already matched with a query node. If it is, its value is 1. Otherwise it is 0. Since a generic file node may be matched with multiple query nodes, the $M(j)$ value of a generic file node is always kept 0. An examination of a file structure against a query structure is conducted by comparing every query node ($0 < i \leq m$) with every file node ($0 < j \leq n$), except those file nodes already matched with any query nodes, for which $M(j)$ equals 1.

Thus at a certain column point i , j is incremented until a match is found between the query node i and the file node j . If a match is found, $P(i)$ is now j , and i is incremented by one. If a match is failed for all possible j values, i is $i - 1$ and j is $P(i)$, which is the last j value where there was a match for the new i value. The examination will end either when complete match is found or when i becomes minus.

The match of a query node and a file node is examined as in Figure 10 by comparing their attributes, elements where no generic nodes are involved, connections with nodes already matched, and other conditions if necessary.

It should be noted that a match can be examined between a generic node and a specific (single) node only by using attributes. Attributes of a query structure and a stored structure mentioned in Figure 10 may be compared by the step illustrated in Figure 11. A series of logical operations are applied as described earlier. The match is determined when a bit is one for all the columns of the file vector for which a bit is one in the query vector. This match is called an inclusive match.

The connections of a query node and those of a file node with those nodes already matched may be compared as illustrated in Figure 12 by reading the connection data of the query connection table for a particular query node and the file connection table for a particular file node. It is important here that only those connections with query nodes to the left of the current column position (nodes of younger node numbers) are examined, i.e., those query nodes that are already matched with one or more of the file nodes.

Several cases illustrating when matching of connection is determined are given in Table VI. If the query node A has a connection with a query node B already matched with a file

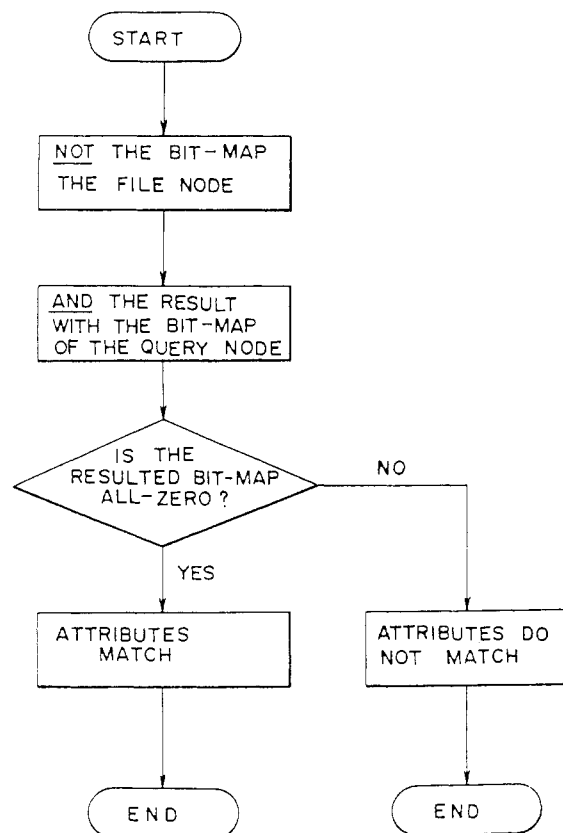


Figure 11. Flow chart of matching query and file attributes.

Table VI. Various Conditions Where Connection Match Is Made

examined node	connecting node	condition of connection
query node A (specific)	B (specific)	connected
file node X (specific)	Y (specific)	connected
query node A (specific)	B (specific)	connected
file node X (generic)	X (generic)	A and B must be in the same ring or chain
query node A (specific)	B (generic)	connected
file node X (specific)	Z (specific)	connected
	Y (specific)	Y and Z must be in the same ring or chain if B and Y match and B and Z do not

node Y, then the file node X, which is now compared with the query node A, must have a connection with the file node Y. In addition, the bond type between X and Y must be the same as that of A and B. If the file node X is generic, i.e., it can be matched with multiple query nodes, there is a chance both A and B match with X. Then the connection between A and B need not be examined as long as both A and B belong to the same ring or chain. If A matches X and B, which is connected to A, matches Y and B is generic, X and Y need not be necessarily connected with each other. If a node Z,

Table VII. Connection Table of File Structure I

node no.	element	attribute	connecting node		connecting node		connecting node		connecting node	
			no.	bond	no.	bond	no.	bond	no.	bond
1	C	6-membered ring with nitrogen	2	4	6	4				
2	C	6-membered ring with nitrogen	1	4	3	4				
3	C	6-membered ring with nitrogen	2	4	4	4				
4	C	6-membered ring with nitrogen	3	4	5	4				
5	N	6-membered ring with nitrogen	4	4	6	4				
6	C	6-membered ring with nitrogen	1	4	5	4	7	1		
7	XX	5- or 6-membered ring with oxygen	6	1	8	1				
8	Cl	halogen	7	1						

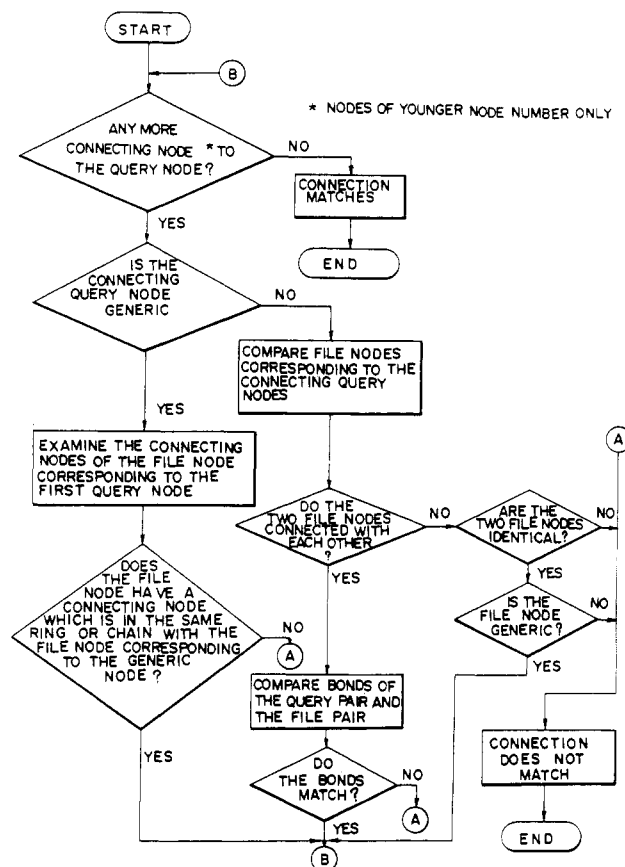
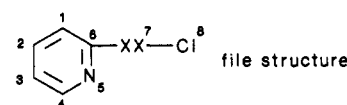


Figure 12. Flow chart of matching query and file connections.

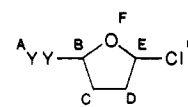
which is in the same ring or chain with Y, is connected to X, then it is determined that the connection is matched.

EXAMPLE

Sample Generic Chemical Structures. The stored structure (structure I) is represented by Table VII (the connection table) and Table VIII (the attribute table) and the query structure (structure II) is represented by (the connection table) and



(XX is an oxygen-containing ring system consisting of one or two component rings with the ring size of five or six)



(YY is a nitrogen-containing heterocycle; the number of component rings is unlimited)

Table VIII. Ring Attributes of File Structure I

bit position	description	1	2	3	4	5	6	7
	element type (may be present)							
1	heteroatom	1	1	1	1	1	1	1
2	oxygen	0	0	0	0	0	0	1
3	sulfur	0	0	0	0	0	0	0
4	nitrogen	1	1	1	1	1	1	0
5	other element	0	0	0	0	0	0	0
	element type (may not be present)							
6	heteroatom	0	0	0	0	0	0	0
7	oxygen	1	1	1	1	1	1	0
8	sulfur	1	1	1	1	1	1	1
9	nitrogen	0	0	0	0	0	0	1
10	other element	1	1	1	1	1	1	1
	no. of component rings (minimum)							
11	1	1	1	1	1	1	1	1
12	2	0	0	0	0	0	0	1
13	3	0	0	0	0	0	0	0
14	4	0	0	0	0	0	0	0
	no. of component rings (maximum)							
15	1	1	1	1	1	1	1	1
16	2	1	1	1	1	1	1	1
17	3	1	1	1	1	1	1	1
18	4	1	1	1	1	1	1	1
	size of rings (may be present)							
19	3	0	0	0	0	0	0	0
20	4	0	0	0	0	0	0	0
21	5	0	0	0	0	0	0	1
22	5 (2 or more)	0	0	0	0	0	0	1
23	6	1	1	1	1	1	1	1
24	6 (2 or more)	0	0	0	0	0	0	1
25	6 (3 or more)	0	0	0	0	0	0	0
26	7	0	0	0	0	0	0	0
27	8 or 9	0	0	0	0	0	0	0
28	10 or more	0	0	0	0	0	0	0
	size of rings (may not be present)							
29	3	1	1	1	1	1	1	1
30	4	1	1	1	1	1	1	1
31	5	1	1	1	1	1	1	1
32	5 (2 or more)	1	1	1	1	1	1	1
33	6	0	0	0	0	0	0	1
34	6 (2 or more)	1	1	1	1	1	1	1
35	6 (3 or more)	1	1	1	1	1	1	1
36	7	1	1	1	1	1	1	1
37	8 or 9	1	1	1	1	1	1	1
38	10 or more	1	1	1	1	1	1	1
	one or more aromatic rings							
39	yes	1	1	1	1	1	1	1
40	no	0	0	0	0	0	0	1

Table X (the attribute table). The search process is described step by step in Table XI.

First, all the atoms (including generic atoms) except hydrogens, which will be called nodes hereafter, are numbered at will. In this example, these nodes correspond to the chemical unit described earlier.

A table is created with each row representing each node of the structure and the first column representing the node number. The second column shows the element type of the node, the third column shows the attribute code of the node, the fourth column shows the number of nodes to which the current node is connected, and the fifth column shows the bond

value of the connection above. The bond value is coded such that a single bond is represented by 1, double bond, 2, triple bond, 3, and aromatic bond, 4. In structure I, a single bond is represented by a straight line and an aromatic bond by a broken line. It is important that the columns four and five form a pair to describe connection information. Similarly, the pairs of columns six and seven, eight and nine, and ten and eleven each describe connections of the current node to the other nodes. Although there are only four pairs of the connection information in Tables VII and IX, one may add more columns if more than four connections are expected. Although attribute data are expressed by text in Tables VII and VIII to enhance the comprehension, they are in fact represented by bit-maps as in Tables VIII and X.

The search is conducted by comparing one by one the nodes of the query structure (structure II) with the nodes of the stored structure. The nodes of the query structure are assigned alphabet values rather than numerals here. The generic node A of the query structure is assigned a dummy element value of YY. In ordinary chemical structure search systems, it is convenient to examine the match of element type first and then the matches of connection, which consist of the node number of the connecting atom and the bond value. In this method the matches of attributes are most important. Although the examination steps described in Table XI show that the element matches precede the attribute match, this order is not critical. It is also possible to eliminate the element matches entirely.

The attributes of the query structure are shown in Table X. Since the number of rings of the node A is undefined (more than one), only the column of the minimum value of 1 is assigned the number 1. Also, no attribute is assigned for the size of the ring.

A match is determined when every column of the attribute bit-map of a stored structure node fulfills the corresponding column of the bit-map of a query structure node; i.e., if a column of the query node is one, then the corresponding column of the stored structure node MUST be one. On the other hand, if the column is zero, the corresponding column of the stored structure node can be either one or zero.

Step-by-Step Illustration. A step-by-step illustration is given in Table XI, using the sample structures I and II, and in Tables VII, VIII, IX, and X. As described in Table XI, this method develops a vector with the node numbers of the query structure as components and assigns the nodes of the stored structure one by one, comparing the connection with other nodes.

First, node 1 of the stored structure matches with node A of the query structure by their attributes. The element value is not examined since the node A is generic. No connection is examined for the first node. Then by comparing node 2 and node B, one finds that their attributes do not match (node B has a five-membered ring while node 2 has a six-membered ring), although the element is the same. Exchanging node 2 with the rest of the nodes of the stored structure, one finds no matches through the last step, step 8. Then one concludes that the initial assignment of node 1 and node A is irrelevant. Thus, one begins comparing node 2 with node A instead. The successful match is found when node 6 is assigned to node A

Table IX. Connection Table of Query Structure II

node no.	element	attribute	connecting node		connecting node		connecting node		connecting node	
			no.	bond	no.	bond	no.	bond	no.	bond
A	YY	heterocycle with nitrogen	B	1						
B	C	5-membered ring with oxygen	A	1	C	1	F	1		
C	C	5-membered ring with oxygen	B	1	D	1				
D	C	5-membered ring with oxygen	C	1	E	1				
E	C	5-membered ring with oxygen	D	1	F	1	G	1		
F	O	5-membered ring with oxygen	B	1	E	1				
G	Cl	halogen	E	1						

Table X. Ring Attributes of Query Structure II

bit position	description	A	B	C	D	E	F
element type (may be present)							
1	heteroatom	1	1	1	1	1	1
2	oxygen	0	1	1	1	1	1
3	sulfur	0	0	0	0	0	0
4	nitrogen	1	0	0	0	0	0
5	other element	0	0	0	0	0	0
element type (may not be present)							
6	heteroatom	0	0	0	0	0	0
7	oxygen	1	0	0	0	0	0
8	sulfur	1	1	1	1	1	1
9	nitrogen	0	1	1	1	1	1
10	other element	1	1	1	1	1	1
no. of component rings (minimum)							
11	1	1	1	1	1	1	1
12	2	0	0	0	0	0	0
13	3	0	0	0	0	0	0
14	4	0	0	0	0	0	0
no. of component rings (maximum)							
15	1	0	1	1	1	1	1
16	2	0	1	1	1	1	1
17	3	0	1	1	1	1	1
18	4	0	1	1	1	1	1
size of rings (may be present)							
19	3	0	0	0	0	0	0
20	4	0	0	0	0	0	0
21	5	0	1	1	1	1	1
22	5 (2 or more)	0	0	0	0	0	0
23	6	0	0	0	0	0	0
24	6 (2 or more)	0	0	0	0	0	0
25	6 (3 or more)	0	0	0	0	0	0
26	7	0	0	0	0	0	0
27	8 or 9	0	0	0	0	0	0
28	10 or more	0	0	0	0	0	0
size of rings (may not be present)							
29	3	0	1	1	1	1	1
30	4	0	1	1	1	1	1
31	5	0	0	0	0	0	0
32	5 (2 or more)	0	1	1	1	1	1
33	6	0	1	1	1	1	1
34	6 (2 or more)	0	1	1	1	1	1
35	6 (3 or more)	0	1	1	1	1	1
36	7	0	1	1	1	1	1
37	8 or 9	0	1	1	1	1	1
38	10 or more	0	1	1	1	1	1
one or more aromatic rings							
39	yes	0	0	0	0	0	0
40	no	0	1	1	1	1	1

and node 7 is compared with node B. Since node 7 is generic, only the attribute is examined. The attribute of node 7 tells it has a five- or six-membered ring with oxygen atoms, which fulfills the condition of node B. Since node B is connected to node A, it is required that node 7 is connected to the corresponding node 6. This requirement is also fulfilled, fortunately. Since the bond values of these two connections are the same,

the single bond, the match of nodes 7 and B, is confirmed.

Then nodes C, D, and E of the query structure all match with node 7. The match is confirmed by the fact that nodes C, D, and E are in the same ring system, because the generic node 7 must be a single ring system. Duplicate matches are possible, since the generic node 7 is considered to consist of multiple nodes. Finally, node 8 matches with node G in every respect, and it is concluded that the stored structure I matches with the query structure II.

DISCUSSION

This method breaks a generic expression down to chemically significant units and represents them by a finite number of attribute positions on a bit-map. This allows one to find a match between nonidentical expression such as a C₃₋₅ alkyl and a C₄₋₇ alkyl or a nitrogen-containing ring and a six-membered heterocycle. Since these attributes are generated for both the specifically defined nodes and generic nodes, the comparison between those two types of nodes is easy.

Exact structure searching is not discussed in detail in this paper. In substructure searching, all the query nodes are examined but not all the file nodes. In exact match searching, all the query nodes as well as all the file nodes should be examined for node matching. This will be achieved by conducting reverse searching, i.e., reversing a query structure and a file structure, after the substructure match is found. Thus, exact match searching is a combination of substructure searching and superstructure searching.

Since it is usually easy to generate the attributes algorithmically from the normal generic expression as well as the specific expression, the method is easily implemented on a computer. It is important, too, that the method is compatible with the currently available specific structure search system using the connection tables, so that the mixed handling of both generic structure and specific structure data is possible.

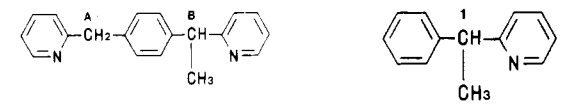
There are many complex expressions in patent claims other than simple variable nodes and generic nodes as defined here. But it is not wise to design a system that will be able to handle all varieties of generic structure expressions. As stated earlier, some generic expressions may be divided into more manageable ones. A conditional expression of variables (when R1 is Cl, then R2 is Ph, etc.) may also be defined as a set of structures that as a whole satisfies all the cases defined by the conditional expression.

It should be also noted that attributes are very powerful in matching specific structures and thus can be used to improve the performance of a specific structure search system. In Figure 13, node a of the file structure will be easily eliminated from the candidates of node 1 in the query structure because the chain attributes do not match by a simple comparison of the number of atoms in the chain fragment.

Table XI. Step-by-Step Matching of File and Query Structures

step	node of query							connect	match	element type	attribute	bond
	A	B	C	D	E	F	G					
1	1							1-A	Y	a	Y	
2	1	2						2-B	N	Y	N	
8	1	8						8-B	N	N		
9	2							2-A	Y	a	Y	
10	2	1						1-B	N	Y	N	
39	6							6-A	Y	a	Y	
40	6	1						1-B	N	Y	N	
46	6	7						7-B	Y	b	Y	Y
47	6	7	1					1-C	N	Y	N	
52	6	7	7					7-C	Y	b	Y	Y ^c
72	6	7	7	7	7	7	7	7-G	N	b	N	
73	6	7	7	7	7	7	8	8-G	Y	Y	Y	Y

^aElement type is not examined for generic node A. ^bElement type is not examined for generic node 7. ^cPresence of connection assumed between nodes belonging to the same generic node.



Bit Position	Type	Attribute	File		Query
			Node a	Node b	Node 1
Number of Atoms					
6		1 or more	1	1	1
7		2 or more	0	1	1
8		3 or more	0	0	0
9		4 or more	0	0	0
Number of Atoms					
6		1	1	0	0
7		2 or less	1	1	1
8		3 or less	1	1	1
9		4 or less	1	1	1

Figure 13. Matching of specific structures using attributes. (Top left) File structure. (Top right) Query structure.

As a conclusion, this method is particularly useful in storing and retrieving complex chemical structures containing both specific and generic expressions.

REFERENCES AND NOTES

- (1) *Computer Handling of Generic Chemical Structures*; Barnard, J. M., Ed.; Gower: Aldershot, U.K., 1984.
- (2) Lynch, M. F.; Barnard, J. M.; Welford, S. M. "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 1. Introduction and General Strategy". *J. Chem. Inf. Comput. Sci.* **1981**, 21, 148-150.
- (3) Barnard, J. M.; Lynch, M. F.; Welford, S. M. "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.* **1981**, 21, 151-161.
- (4) Welford, S. M.; Lynch, M. F.; Barnard, J. M. "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.* **1981**, 21, 161-168.
- (5) Barnard, J. M.; Lynch, M. F.; Welford, S. M. "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 4. An Extended Connection Table Representation for Generic Structures". *J. Chem. Inf. Comput. Sci.* **1982**, 22, 160-164.
- (6) Welford, S. M.; Lynch, M. F.; Barnard, J. M. "Towards Simplified Access to Chemical Structure Information in the Patent Literature". *J. Inf. Sci.* **1983**, 6, 3-10.
- (7) Welford, S. M.; Lynch, M. F.; Barnard, J. M. "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 5. Algorithmic Generation of Fragment Descriptors for Generic Structures". *J. Chem. Inf. Comput. Sci.* **1984**, 24, 57-66.
- (8) Barnard, J. M.; Lynch, M. F.; Welford, S. M. "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 6. An Interpreter Program for the Generic Structure Description Language GENSAL". *J. Chem. Inf. Comput. Sci.* **1984**, 24, 66-71.
- (9) Scholley, A. "A Relaxation Algorithm for Generic Structure Screening". *J. Chem. Inf. Comput. Sci.* **1984**, 24, 235-241.
- (10) Lynch, M. F.; Barnard, J. M.; Welford, S. M. "Generic Structure Storage and Retrieval". *J. Chem. Inf. Comput. Sci.* **1985**, 25, 264-270.
- (11) Gillet, V. J.; Welford, S. M.; Lynch, M. F.; Willett, P.; Barnard, J. M.; Downs, G. M.; Manson, G.; Thompson, J. "Computer Storage and Retrieval of Generic Chemical Structures in Patents. 7. Parallel Simulation of a Relaxation Algorithm for Chemical Substructure Search". *J. Chem. Inf. Comput. Sci.* **1986**, 26, 118-126.
- (12) Kudo, Y.; Chihara, H. "Chemical Substance Retrieval System for Searching Generic Representations. 1. A Prototype System for the Gazetted List of Existing Chemical Substances of Japan". *J. Chem. Inf. Comput. Sci.* **1983**, 23, 109-117.
- (13) Fisanick, W. "Requirements for a System for Storage and Search of Markush Structures". In *Computer Handling of Generic Chemical Structures*; Barnard, J. M., Ed.; Gower: Aldershot, U.K., 1984; pp 106-129.
- (14) Zamora, A. "An Algorithm for Finding the Smallest Set of Smallest Rings". *J. Chem. Inf. Comput. Sci.* **1976**, 16, 40-43.
- (15) Dittmar, P. G.; Stobaugh, R. E.; Watson, C. E. "The Chemical Abstracts Service Chemical Registry System. 1. General Design". *J. Chem. Inf. Comput. Sci.* **1976**, 16, 111-121.
- (16) Sussenguth, E. H., Jr. "A Graph-Theoretic Algorithm for Matching Chemical Structures". *J. Chem. Doc.* **1965**, 5, 36-43.
- (17) Figueras, J. "Substructure Search by Set Reduction". *J. Chem. Doc.* **1972**, 12, 237-244.
- (18) Kudo, Y.; Sasaki, S. "The Connectivity Stack, a New Format for Representation of Organic Chemical Structures". *J. Chem. Doc.* **1974**, 14, 200-202.
- (19) Kudo, Y.; Sasaki, S. "Principle of Exhaustive Enumeration of Unique Structures Consistent with Structural Information". *J. Chem. Inf. Comput. Sci.* **1976**, 16, 43-49.

BOOK REVIEWS

Computer Software Applications in Chemistry. By Peter C. Jurs. John Wiley & Sons, Inc., New York. 1986. xiv + 253 pp.

This book's stated aim is to provide an overview of computer applications in chemistry at an introductory level. It is specifically restricted to computer software that does not include laboratory usage. Thus, there is no discussion of software algorithms specific to that type of application. The book is divided into four parts, which are titled "Introduction", "Numerical Methods", "Nonnumerical Methods", and "Graphics". Where appropriate, literature references are included for the reader interested in more technical details. Most citations are from the 1970s or earlier, with a few as recent as 1985. The Introduction contains a very brief discussion of digital computers and program design (14 pages) that is almost too lacking in detail (e.g., pp 16-18 discuss assembly and machine language code, but define neither). Similar criticism can be made of the six page (!) "Graphics Display" section, which could have been omitted without serious compromise of the book's main goals. The two central sections are well-done and represent reasonable surveys of

a variety of numeric and nonnumeric software applications in chemistry including simplex optimization, graph theory, pattern recognition, library searching, numerical integration, matrix algebra, etc. All in all, the book does a fine job of defining the scope of computer software for chemistry at a level suitable for undergraduates or beginning graduate students. Experts in any of the areas covered will learn little new here—as intended by the author. From a pedagogical standpoint, *Computer Software Applications in Chemistry* can be highly recommended for use in a survey course. It will fill a unique role in this regard.

Errors are few, but they do exist (e.g., test for task on p 12, a heading "The Simlex Method" on p 129, alternate usage of Jordan and Jordon as spellings for Jordan on p 89, and, chemically most serious, the use of the word "absorption" in several places on p 109 where adsorption is intended). However, minor typographical errors aside, this is a high-quality production and is recommended for anyone seeking a quick overview of the book's important topic.

Charles L. Wilkins, University of California, Riverside