effectiveness. Did we address the right problem? Did the information packet cause any *different* action on the part of the recipient? Was the information even used in subsequent action steps by the recipient? What was the ultimate *value* of the decision to which the information contributed? Was its contribution to the decision substantive or supportive? Was it filed away for possible future use or tossed into the round file?

We first have to do the right thing and *then* learn to do it well. Packets of information (large or small), even delivered on time in neat array, with great eye appeal, have no value unless they are used to solve problems or support productive decisions. Hence, we do need an integrated approach with our users. We need candid feedback regarding the usefulness (*value* if you will) of the data/information/knowledge transferred.

SUMMARY

An overview of the environment (from my perspective) related to information resource management in pharmaceutical R&D has been presented. Some notions of organizational preference (functional), employee selection (chemist turned information scientist), automation (user friendly, cost effective), and the value of project teams (information transfer) have been noted. Difficulties associated with keeping our innovative tools sharp were observed. Finally, we noted that our bottom line—productivity—should first consider what is useful (effectiveness) and then learn how to do it well—efficiency. Success in the management of information resources depends on the proactive delivery of information packets which find their way into problem solving and decision support for scientists or line managers.

REFERENCES AND NOTES

(1) Laubach, Gerald D. "Impact of Industrial Innovation in the 1980's—The Pharmaceutical/Health Industry". *Res. Management* Mar **1981**, 9-11.
(2) Foster, Richard N. "Effective R&D Operations in the 1980's—Boosting the Payoff from R&D". *Res. Management* Jan **1982**, 22-3.
(3) Sarett, Lewis H. "Stimulating Technological Innovation—The Innovative Spirit in an Industrial Setting". *Res. Management* Nov **1979**, 15-6.
(4) Gerstenfeld, Arthur; Sumiyoshi, Keyi "The Management of Innovation in Japan—Seven Forces that Make the Difference". *Res. Management* Jan **1980**, 30-4.

Computer Representation of Generic Chemical Structures by an Extended Block-Cutpoint Tree

TAKASHI NAKAYAMA

Foundation for Advancement of International Science, 3-9-1, Amakubo, Sakura-mura, Niihari-gun, Ibaraki 305, Japan

YUZURU FUJIWARA*

Institute of Information Sciences and Electronics, University of Tsukuba, Sakura-mura, Niihari-gun, Ibaraki 305, Japan

Received September 2, 1982

A representation scheme for generic chemical structures (Markush formulas) is presented. This method is an extension of a BCT (block-cutpoint tree) representation for specific chemical structures, and is called an EBCT (extended BCT) representation of generic chemical structures. A general Markush formula is dissolved into several simple Markush formulas by expanding conditions and nested substitutions. Variable parts of simple Markush formulas are represented in terms of a simple type of substituent group called a generic unit. An inverted file for homologous series and a fragment screening facilitate (sub)structure searches.

INTRODUCTION

One of the important problems in dealing with chemical structure information by computers is how to implement the representation of chemical structures in computer storage. So far, many representation methods have been proposed and implemented for specific chemical structures.[1] Those methods are categorized to two major approaches: the linear representation and the topological representation. One major reason why so many representation methods have been and will be devised is that we do not have such a complete representation method yet to support various types of substructure searches. We have presented the BCT representation of chemical structures as a method for representing specific chemical structures.[2] The features of this method are that it gives a hierarchical view of chemical structures in terms of blocks (ring assembly) and that those blocks become common structural descriptors for both the topological relation and the fragmentation of each chemical structure. As a result, the BCT representation facilitates substructure searches with flexibility and speed.

On the other hand, there have been few reports about the representation of generic chemical structures, although several retrieval systems, such as IDC[3] and Derwent,[4] are in use for generic or patent information on compounds. In practice, however, it is not always satisfactory to retrieve generic chemical structures by those current systems, with respect to the speed and the flexibility in search or the cost of retrieval.[5]

Recently Lynch et al. have reported a descriptive language (a chemical grammar) for generic chemical structures (in particular, Markush formulas in patents), and it is well suited for the description of Markush formulas, supplanting the description by natural language.[6-8] However, it is not so clear how efficient a search their chemical grammar gives, as compared with other representations of generic chemical structures, partly because the whole chemical grammar and the screening system[9] are still under development. Apparently it is advan-

0095-2338/83/1623-0080$01.50/0

Table I. Connectivity Matrix

	B_1	B_2	B_3	B_4	B_5	B_6	B_7
B_1	0	1	0	2	0	2	0
B_2	1	0	6	0	0	0	0
B_3	0	1	0	0	0	0	0
B_4	1	0	0	0	2	1	0
B_5	0	0	0	1	0	0	0
B_6	1	0	0	1	0	0	2
B_7	0	0	0	0	0	1	0

tageous to organize a chemical structure data base as a comprehensive system which deals with generic structures (Markush formulas) as well as with specific structures.[5] As is well-known,[6] there are three types of searches performed on a generic chemical structure data base: (1) searches for specific structures within a generic structure, (2) searches for specific substructures within a generic structure, and (3) detection of common specific structures between two or more generic structures. The guiding principle of structure searches is to reduce the number of matches between the query structure and object structures, in other words, to detect the unmatching as fast as possible and to discard the unmatched object structures. The difficulty of this problem is due to the combinatorial explosion of the number of atom-by-atom matchings. Fragment screening is the method ordinarily employed to implement the guiding principle. That is, the problem is to construct a good screening system.

Extended BCT representation of generic chemical structures, presented in this paper, gives an approach to organizing a chemical data base system which provides various views of the structures. That is, the consistent representation is implemented for both specific and generic chemical structures by the concept of extended BCTs, and it gives the facilities used for various facets of substructure searches.

EXTENDED BCT REPRESENTATION

A chemical structure is regarded as a graph whose vertices and edges correspond to atoms and bonds, respectively. The BCT (block-cutpoint tree) representation of chemical structures is a hierarchical representation method where an intermediate descriptive unit is introduced to facilitate substructure search. This descriptive unit is called a block or biconnected component of a graph and corresponds to what is called a ring system or a ring assembly, except for blocks that have only two vertices. A cutpoint is defined as a vertex whose removal increases the number of connected components of a graph, so vertices which constitute an acyclic part of a graph or spiro connection are cutpoints. Blocks and cutpoints can be extracted from a graph in a strictly algorithmic way.[2,10] A BCT is constructed as a bigraph from those blocks and cutpoints, where the set of vertices is composed of two classes of vertices: blocks and cutpoints. (The term "node" is used instead of "vertex" hereafter to mean vertices of a BCT, to distinguish it from those of an original graph.) An edge is defined between a cutpoint node c and a block node B, if c is a member of the set of constituent vertices of block B. The graph constructed in this manner is a tree, so it is called a block-cutpoint tree (BCT).[11]

The BCT representation of chemical structures employs the block-cutpoint tree to express a rough shape of a structure, and the exact structure is given by a connectivity matrix between blocks whose structures are defined in the block dictionary. Figure 1 shows a graph, its blocks, and the derived block-cutpoint tree, and Table I shows the corresponding connectivity matrix. The constituent blocks are entered in the fragment record of the compound. The details of file organization of these records were presented in our previous paper.[12]

In this paper, it is shown that the generic chemical structures (Markush formulas) can be registered in terms of the BCT

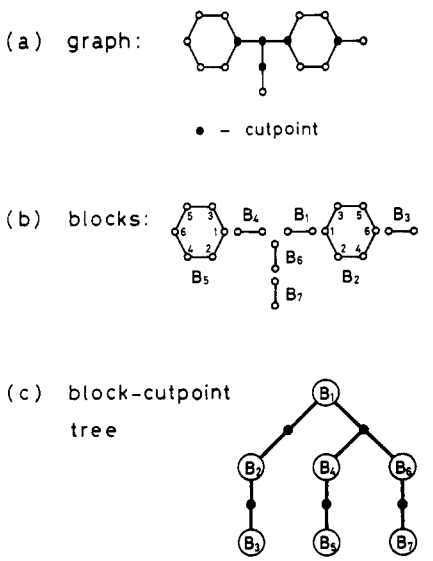


Figure 1. Graph and its blocks and block-cutpoint tree.

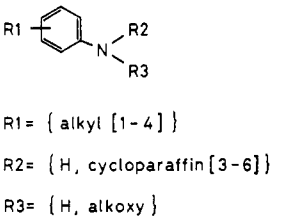
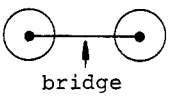


Figure 2. Simple generic structure expressed by a Markush formula.

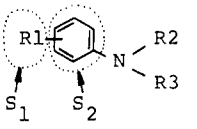
concept by modifying or extending its several specifications. Figure 2 shows a simple generic structure expressed by a Markush formula. The term "simple" is defined precisely later.

A Markush formula is regarded as a structure consisting of two different parts: a constant part and variable parts called substituents. A Markush *group* is a set of alternative substituents, one of which is exclusively attached to a constant part or another variable part of the generic structure. In Figure 2, R1, R2, and R3 are Markush groups, and the remainder is a constant part. The extended BCT (EBCT) representation for a simple Markush formula is roughly sketched as follows: a constant part (a specific structure) is encoded as a BCT, and each Markush group which is a component of a variable part is treated as a node corresponding to a block, called a *generic node*. Thus, in an EBCT, a set of block nodes consists of specific block nodes and generic nodes. The connectivity between these nodes is defined on the basis of the type of connector, which is a shared substructure between the two nodes.

(a) Bridge Type. Two nodes are connected by a bridge, so



there are no atoms shared between them. Assuming that S_1 and S_2 represent sets of vertices of the substructures corresponding to the nodes, this type of connectivity means $S_1 \cap S_2 = \phi$. Connections between specific/generic nodes expressed by an explicit bond in a Markush formula are included in this type of connectivity. In Figure 2, the connection between R1 and the benzene ring is a bridge type, and so are the connections for R2 and R3.



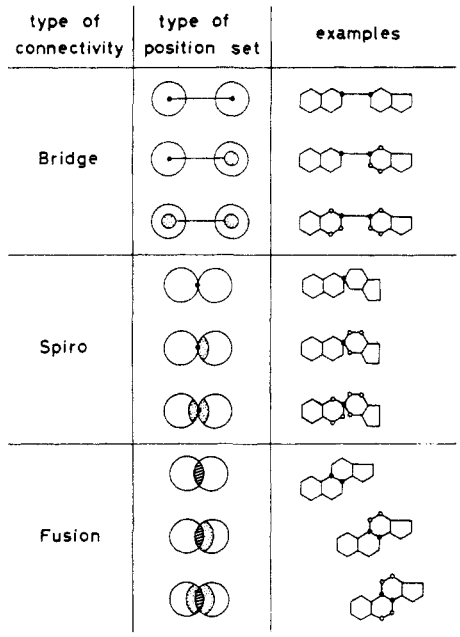
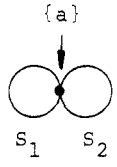
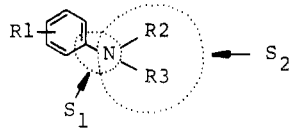


Figure 3. Classification of the connectivity between specific/generic nodes. Black vertices and hatched areas represent atoms shared between two nodes. Dotted areas and white vertices indicate that they are the alternatives of a shared atom.

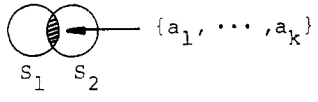
(b) Spiro Type. Two nodes share a single atom (vertex); $S_1 \cap S_2 = \{a\}$. If two generic nodes are connected to the same



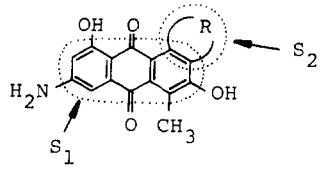
atom and a condition is given to make them a single atom, this type of connection appears. In Figure 2, if a condition R2 + R3 = {pyrrolidyl, piperidyl} is given, N is the shared atom between node S_1 (= C–N) and node S_2 (= R2 + R3).



(c) Fusion Type. Two nodes share more than one atom (vertex); $S_1 \cap S_2 = \{a_1, ..., a_k\}$ $(k > 1)$.



The following example shows this type of connection.



The variability of generic chemical structures derives from two kinds of alternative sets: a substituent set and a position set. A substituent set corresponds to a Markush group and is regarded as a generic node. A position set is a set of vertices (atoms) whose members are selected for a connector to the other node. Three types of connectivity were described above,

Table II. Four Types of Expression of Elements of ECM (c_{ij})

type	expression
specific	d_1
bridge	$b(d_1, ..., d_k)$
spiro	$s(d_1, ..., d_l)$
fused	$f((d_{11}, ..., d_{1n}), ..., (d_{m1}, ..., d_{mn}))$

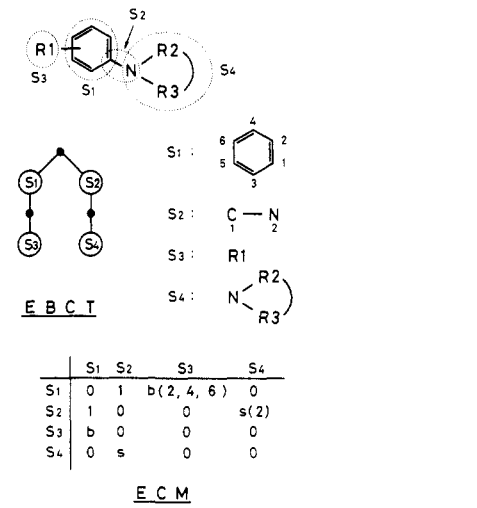


Figure 4. EBCT representation of a generic structure.

and the position of attachment for them is variable. Classification of the connectivity is illustrated in Figure 3, where dotted areas indicate that there are multiple alternatives for connection of the node.

The connectivity between these nodes (specific/generic nodes) is represented by an extended connectivity matrix (ECM), where elements indicate types of connectivity and positions of attachment. The ith row of an ECM represents how the ith node is connected to the other nodes. The element c_{ij} of an ECM includes information about the connectivity type and the attachment position of node i to node j. The position of attachment is identified by a number or a tuple of numbers assigned to the atoms (vertices) of node i. There are four types of expressions of the element c_{ij}, shown in Table II. The notations b, s, and f indicate the connectivity types *b*ridge, *s*piro, and *f*usion, respectively, and d_i $(i = 1, 2, ...)$ indicates a position number. $c_{ij} = b(d_1, ..., d_k)$ means that node i is connected to node j at one of the positions $d_1, ..., d_k$, which are terminals of the bridge connector. $c_{ij} = s(d_1, ..., d_l)$ means that node i is connected to node j at one of the positions $d_1, ..., d_l$, which are shared atoms (cutpoints). $c_{ij} = f((d_{11}, ..., d_{1n}), ..., (d_{m1}, ..., d_{mn}))$ means that node i is connected to node j at one of the groups of $(d_{11}, ..., d_{1n}), ..., (d_{m1}, ..., d_{mn})$, which are groups of shared atoms. These positions may not be specified in an ECM. Figure 4 shows an example of an EBCT representation of a generic structure, where block node S_1 is connected to S_2 specifically at vertex 1 and to generic node S_3 by a bridge at vertex 2, 4, or 6. It shows further that S_2 is connected to S_1 specifically at vertex 2 and to generic node S_4 by a spiro junction at vertex 1. Generic nodes S_3 and S_4 are connected to S_1 and S_2 by a bridge and spiro junction, respectively.

There is one additional modification of the definitions in an EBCT: The appropriate acyclic part of a graph which has three or more vertices is often regarded as a single superblock node of an EBCT rather than as two or more nodes. This treatment gives a more macroscopic view of chemical structures. For example, in Figure 1 we can regard blocks B_6 and B_7 as a single superblock, say B_8, instead of two separate simple blocks. Then we have another reduced representation of graph a in Figure 1, and this is very useful when the acyclic part is

a popular one or consists of many atoms (although only one block was reduced in this case, B_8 instead of B_6 and B_7).

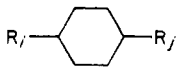
GENERIC UNIT DESCRIPTION

Complex Markush formulas contain various sorts of descriptors which describe their variable parts. In general, a variable part has two major generic factors: substituents and positions of attachment. The position of attachment is defined between a constant part and a variable part or between variable parts. This was discussed in the previous section. In this section, the classification of substituents is discussed, and the generic unit description of Markush formulas is presented.

Substituents contain two types of entities: specifically defined substructures and generically defined substructures which are homologous series, nested substituents, verbal expressions, or sets of substructures indicated by multipliers. These entities are grouped as a set of substituents called a Markush group (denoted by R_i) which gives alternatives to be attached:

$$R_i = \{r_{i1}, r_{i2}, \ldots, r_{in}\}$$

Only one of the elements $r_{i1}, \ldots, r_{in}$ is selected exclusively to be attached. There are n ways of selection of alternatives for R_i, so there are mn ways of combination of an alternative selection for R_i and R_j in a Markush formula. However, not



$$R_i = \{r_{i1}, \ldots, r_{im}\}$$
$$R_j = \{r_{j1}, \ldots, r_{jn}\}$$

all combinations of elements of R_i and R_j are always allowed to appear, and an element r_{ik} ($k = 1, \ldots, m$) itself may be a Markush group. Conditions are often given to restrict their occurrence of attachment positions.

The member r_{ik} of Markush group R_i is classified in one of those categories described above. A summary of the categories is given below.

Specifically Defined Substructure. This is a member whose substructure is given by a connection table or a nomenclatural term, i.e., block/superblock. Nomenclatural terms are associated with connection tables internally by a dictionary.

Homologous Series. In general, a homologous series can be regarded as an infinite set of homologues. It is specified in a Markush formula with various restrictions, so that in most cases a subset of a homologous series is employed to represent a substituent. The handling of homologous series is described later in more detail.

Nested Substitution. A Markush group itself is allowed to appear in another Markush group as its member; i.e., the expression of nested substitution is allowed to be specified as a substituent. In other words, a Markush group can be defined recursively.

Verbal Expression. The verbal expression is usually used to describe some kind of attribute which specifies a certain group of chemical structures. In order to answer any query using verbal expression, substituents should be structured to include all sorts of chemical attributes corresponding to verbal expressions; this will not be discussed here.

Multiplier. Some part of a chemical structure may be considered as a repetition of its smaller substructure, as seen typically in polymers. If the number of repetitions is variable, that part of the structure is variable. The number of repetitions is called a multiplier, and the set of substructures indicated by a single multiplier gives alternative substituents. Multipliers can be applied to any of the specific or generic structures. If a multiplier is applied to a specific structure, it gives a set of specifically defined substructures as explained above.

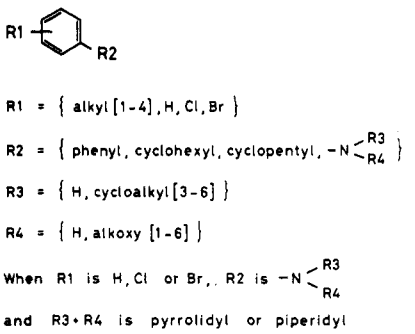


Figure 5. Markush formula which contains various sorts of descriptors.

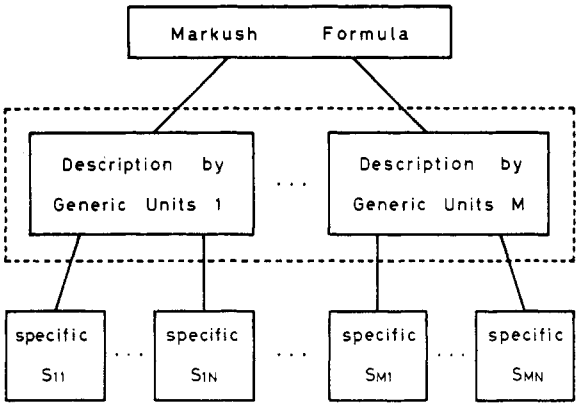
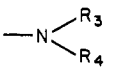


Figure 6. Hierarchy of structure representation from a generic unit view.

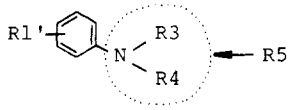
Figure 5 shows the occurrence of these categories. It is difficult and inefficient to deal with these complicated expression directly by computers. It is desirable for a simple descriptive unit to be introduced to describe these various generic concepts. The complexity is mostly due to the use of the nested substitution and the condition for the coexistence of substituents. The handling of homologous series and nomenclatural terms is facilitated by providing a proper dictionary, as described in the following section.

The simple descriptive unit, called a *generic unit*, is introduced to represent Markush formulas systematically. A generic unit is defined as a set of substituents (a Markush group) whose members are restricted to specifically defined substructures (blocks/superblocks) and/or homologous series. Consequently, the nested substitution has to be dissolved, and the condition for the coexistence of substituents has to be applied to each Markush group or to its members. Then a Markush formula which contain nested substitutions or conditions is expanded to several Markush formulas, which are called *simple* Markush formulas and contain only the generic units. Figure 6 shows the hierarchy of structure representation from a generic unit view. It would be futile to resolve a generic expression into all its specific expressions, but it is necessary and useful to resolve it into intermediate, tractable generic expressions. The generic unit does this. Basically this suggests that the descriptors of Markush formulas should be limited to sets of specific substructures and/or homologous series identifiers.

For example, in Figure 5, a condition is imposed for R1 and R2: When R1 is H, Cl, or Br, R2 is -NR3R4, and R3 + R4 makes pyrrolidyl or piperidyl. And Markush group R2 contains a nested substitution:

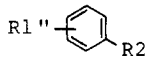


The formula is then expanded by the condition:

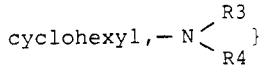
(1) 

R1' = {H,Cl,Br}

R5 = {pyrrolidyl,piperidyl}

(2) 

R1" = {alkyl[1-4]}

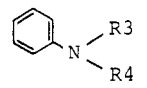
R2 = {phenyl,cyclopentyl, cyclohexyl,  }

Case 2 is further expanded with respect to the nested substitution:

(2-1) 

R1" = {alkyl[1-4]}

R2' = {phenyl,cyclopentyl,cyclohexyl}

(2-2) 

R1" = {alkyl[1-4]}

R3 = {H,cycloalkyl[3-6]}

R4 = {H,alkoxy[1-6]}

Expressions of cases 1, 2-1, and 2-2 are simple Markush formulas, and this form of Markush formula is the object of an EBCT representation, which is described by generic units R1′, R1″, R2′, R3, R4, and R5.

FILE ORGANIZATION

As discussed in the previous section, a general Markush formula is reduced to several simple Markush formulas by dissolving nests of Markush groups or applying conditions. These simple Markush formulas are then coded to EBCTs, and the corresponding fragment records are created. EBCTs give an intermediate view of topological structures of simple Markush formulas, whereas corresponding fragment records give existential information of substructures, i.e., a part of screening system. The substructures which are the descriptors of EBCTs and the fragment records consist of blocks and superblocks. A part of homologous series also appears in some fragment records. These relations are implemented as a file constitution as shown in Figure 7. The lines connecting the files indicate referential relationships between them, which form a network structure internally. The MASTER file contains the miscellaneous nonstructural record items of Markush formulas. IF is an inverted file for homologous series, FRAG is a fragment file for screening, and EBCT is a file for extended connectivity matrices. BD and SBD are dictionaries of blocks and superblocks, respectively. They contain the topological structures of blocks/superblocks, and their identifiers appear in the records of EBCT and FRAG as foreign keys. ND is a dictionary of nomenclatural terms of homologous series and extended homologous series which are explained later (the name of each homologue is also included). In this section, we discuss how homologous series are represented internally as part of generic units to describe Markush formulas and fragments for screening.

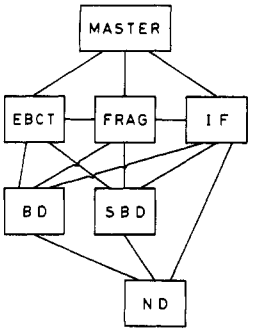


Figure 7. Diagram of file constitution.

Chart I

(1) paraffin hydrocarbon (C_nH_{2n+2})

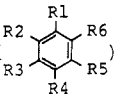
$CH_3 + C_nH_{2n+1}$

(2) olefin hydrocarbon (RCH=CHR')

$CH{=}CH + (C_nH_{2n+1}, C_{n'}H_{2n'+1})$

(3) hydrocarbons of acetylene series (RC≡RC')

$C{\equiv}C + (C_nH_{2n+1}, C_{n'}H_{2n'+1})$

(4) aromatic compounds with one benzene ring 

 $+ (C_{n_1}H_{2n_1+1}, \cdots, C_{n_6}H_{2n_6+1})$

(5) carboxylic acid (RCOOH)

$COOH + C_nH_{2n+1}$

(6) alkoxy (RO-)

$CH_3O + C_nH_{2n}$

Representation of Homologous Series. Homologous series can be generally expressed as follows:

kernel structure + variable parts

where the kernel structure corresponds to a functional group or a starting structure of the homologous series, and the operator "+" denotes that variable parts are connected to a kernel structure. The variable part can be regarded as alkyl. Shown in Chart I are examples of the expression of homologous series. Each homologous series is determined by specifying the scope of the carbon count *n*, and it is regarded as a subset of an infinite set of a homologous series corresponding to its general expression.

On the other hand, it is useful to group a certain type of compounds which have a particular structural feature in common but are not members of a homologous series in a strict sense of the term; that is, the structural difference (the variable parts of a general expression) is not restricted to the difference of alkyl parts. For example, "amine" includes primary amine (R_1NH_2), secondary amine (R_2NH), and tertiary amine (R_3N), and R_1, R_2, and R_3 need not be alkyls. Thus the term "amine" is used as a generic expression for those three types of compound groups. These terms are entered in the nomenclature dictionary (ND) in constructing a thesaurus. This type of compound group is called an extended homologous series. The general expression for extended homologous series is also described as kernel structure + variable parts, where the kernel structure may be a sum of disconnected substructures, and the variable parts may not be alkyl.

Both homologous series and extended homologous series are represented in the same format of records and files. (So we shall use the term "homologous series" to include "extended homologous series".) A homologous series is represented by

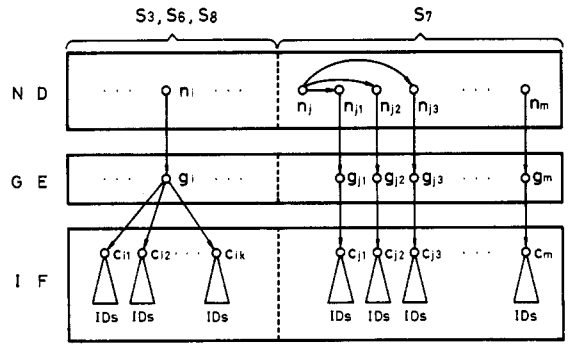


Figure 8. Record relation between a nomenclature dictionary (ND) and an inverted file (IF). S_3, S_6, S_8, and S_7 correspond to those subsets in Figure 9 (set relation).

Table III. Meaning of Bit Position of *c* Quantifier

bit	meaning	bit	meaning
0	$c=0$	9	$c=9$
1	$c=1$	10	$c=10$
2	$c=2$	11	$c=11$
3	$c=3$	12	$c=12$
4	$c=4$	13	$c=13$
5	$c=5$	14	$c \geq 14$
6	$c=6$	15	$c=[0,\infty]$
7	$c=7$		
8	$c=8$		

three records from files ND, GE, and IF. ND is the nomenclature dictionary, GE is a file for storing general expressions of homologous series, and IF is an inverted file whose entries are specific subsets of general expressions of homologous series. The relationships between these files are shown in Figure 8, where the nodes n_i, g_i, c_{i1}, etc. denote records of the files, and the lines between the nodes denote referential relationships. IDs indicated by a triangle in the IF means a set of identifiers of generic chemical structures. Each entry of the IF branching from the same general expression is discriminated by specifying the distribution of the carbon count; thus the GE acts as a kind of directory to the IF.

This discriminating parameter can be regarded as one of the quantifiers which characterize homologous series and is denoted by the *c* quantifier. The resolving power for homologous series is limited to the level of the *c* quantifier at present, but it can be augmented simply by catenating other quantifiers after the *c* quantifier. The *c* quantifier consists of 16 bits (from bit 0 to bit 15). The meaning of each bit position is shown in Table III. Membership examination of blocks/superblocks for homologous series is performed by using these files. For example, the *c* quantifier c_{i1} = 0101110000000000 represents a homologous series whose carbon count is 1, 3, 4, or 5. A total set of a homologous series can be specified by setting bit 15 of the *c* quantifier. The range of the carbon count which can be specified in a *c* quantifier is from 0 to 13; that is to say, the *c* quantifier cannot represent those homologous series unambiguously that have more than 13 carbon atoms in their variable parts. However, the typical range of the carbon count is adequately covered by this handling of homologous series.

On the other hand, the extended homologous series is characterized by the kernel structure alone. In other words, the variable parts of the extended homologous series cannot be characterized simply by the carbon count. So only one entry in IF is derived from a general expression in GE (e.g., c_{j1} from g_{j1}). Instead, a thesaurus is constructed in ND corresponding to an extended homologous series (e.g., n_{j1} [primary amine] can be accessed from n_j [amine]).

Use of Homologous Series. Homologous series are allowed to appear in the elements of generic units. This is because it

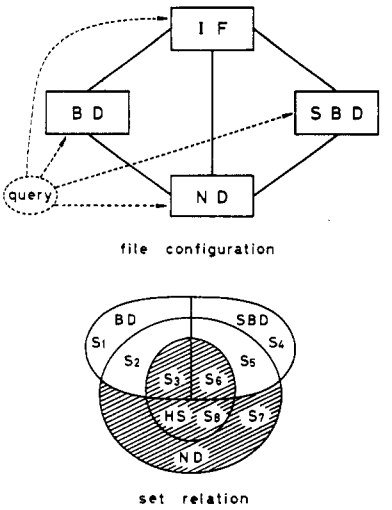
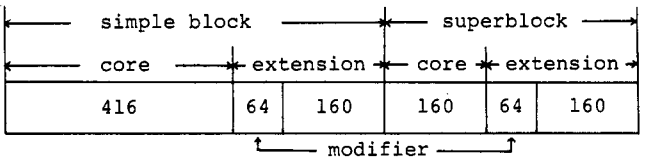


Figure 9. Four files which are used for query analysis indicated by dotted lines, and the set relation between those files. Files BD and SBD have no entity in common, but there are some common entities among these four files, and they are classified in eight subsets according to the type of overlapping.

Chart II



is difficult and even impossible in some cases to expand a homologous series into extensive homologous members. The generic expression has to be used for representing a concept that has a wide range of entities. In practice, "alkyl", for example, implies an infinite set of compounds. A nomenclatural term, "methyl", or a structural representation, CH_3, is a member of "alkyl". Such an occurrence in query structures should be examined to see if it is a member of any homologous series, as well as if it is a block, a superblock, or a nomenclatural term. Screening by fragments would then be performed.

Three files, BD, SBD, and ND, are used for identifying blocks, superblocks, and nomenclatural terms. BD and SBD are mutually exclusive, and ND overlaps them. The relationship between these files is shown in Figure 9, where HS means the set of homologous series in a narrow sense (i.e., variable parts are "alkyl"). Preprocessing for query structures involves resolving them into fragments (blocks/superblocks) and coding them to the EBCT. The fragments are identified by these files, and the screening pattern for FRAG is made from their identifiers if they belong to BD or SBD. If a fragment belongs to HS ($=S_3 \cap S_6 \cap S_8$) or to subset S_7 in Figure 9 (hatched area), the inverted file (IF) is used.

Fragment Record. Fragment records are created for Markush formulas as mentioned above. The *i*th records for a Markush formula in MASTER, EBCT, and FRAG are related to each other as shown in Figure 10. This shows that a Markush formula (MF) containing conditions or nested substitutions is expanded into several simple Markush formulas (SMF_{i1}, ..., SMF_{in}) and that FRAG contains several kinds of fragment records as well as lists of homologous series. Those fragment records (TFR, SFR_{i1}, ..., SFR_{in}, CFR_1, ..., CFR_p, UFR_1, ..., UFR_q) have the same record format (fixed length of 1024 bits) illustrated in Chart II.[12]

FRAG(*i*) in Figure 10 is explained as follows.

(1) TFR (total fragment record): This is a record of total fragments (blocks/superblocks) occurring in the original

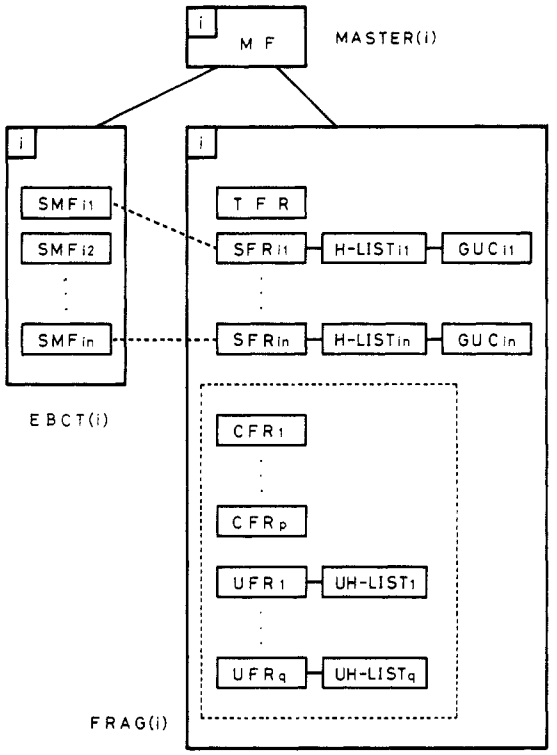


Figure 10. Relation between the three files MASTER, EBCT, and FRAG for the *i*th record.

Table IV. Extended Connectivity Matrices

ECM_1

	S_1	S_2	S_5	S_6
S_1	0	1	0	*b* (2, 3, 4, 5, 6)
S_2	1	0	*s* (2)	0
S_5	0	*s*	0	0
S_6	*b*	0	0	0

ECM_2

	S_1	S_7	S_8
S_1	0	*b* (2, 3, 4, 5, 6)	*b* (1)
S_7	*b*	0	0
S_8	*b*	0	0

ECM_3

	S_1	S_2	S_3	S_4	S_7
S_1	0	1	0	0	*b* (2, 3, 4, 5, 6)
S_2	1	0	*b* (2)	*b* (2)	0
S_3	0	*b*	0	0	0
S_4	0	*b*	0	0	0
S_7	*b*	0	0	0	0

Markush formula (MF). TFR is a logical sum of SFR_{i1}, ..., SFR_{in} and is explained below.

(2) SFR_{ik} (simple fragment record; $k = 1, ..., n$): This indicates a fragment record corresponding to a simple Markush formula (SMF_{ik}) derived from MF. The logical sum of SFR_{i1}, ..., SFR_{ik} gives the TFR mentioned above.

(3) CFR_j (constant part fragment record; $j = 1, ..., p$): This indicates a fragment record corresponding to a constant part of a simple Markush formula (SMF_{ik}). A constant part may be shared by two or more simple Markush formulas, so only the different constant parts are actually represented in FRAG(*i*).

(4) $H\text{-}LIST_{ik}$ (homologous series list; $k = 1, ..., n$): This is a list of identifiers of homologous series occurring in a simple Markush formula (SMF_{ik}). The extensive representation of each homologous series is stored in the inverted file (IF), and

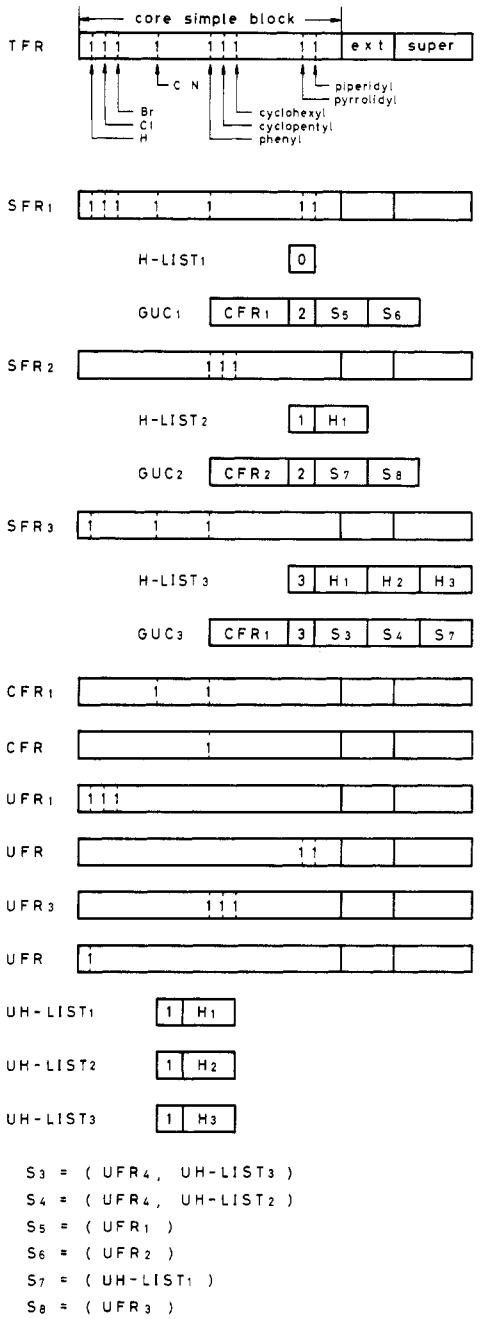


Figure 11. Fragment records generated from the generic structure in Figure 5.

the screening for homologous series is usually performed on IF in the first place. $H\text{-}LIST_{ik}$ is then used for identifying which simple Markush formula it belongs to.

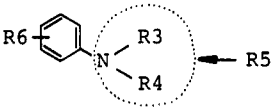
(5) GUC_{ik} (generic unit constitution; $k = 1, ..., n$): This gives a generic unit constitution of a simple Markush formula (SMF_{ik}); that is, it contains the identifiers of generic units and a constant part. Each generic unit is a set of specific blocks/superblocks and homologous series and is represented in terms of UFR and UH-LIST (explained below), just as a simple Markush formula is represented in terms of SFR and H-LIST.

(6) UFR_j (generic unit fragment record; $j = 1, ..., q$): This indicates a fragment record corresponding to blocks/superblocks which are the members of a generic unit. UFR_j together with some of $UH\text{-}LIST_k$ forms the generic unit.

(7) $UH\text{-}LIST_j$ (generic unit homologous series list; $j = 1, ..., q$): This is a list of identifiers of homologous series occurring in a generic unit. $UH\text{-}LIST_j$ together with some of UFR_k forms the generic unit. The catenated record (UFR_j, UH-

Chart III

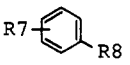
SMF_1:



R5 = {H,Cl,Br}

R6 = {pyrrolidyl,piperidyl}

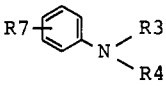
SMF_2:



R7 = {alkyl[1-4]}

R8 = {phenyl,cyclopentyl,cyclohexyl}

SMF_3:



R3 = {H,cycloalkyl[3-6]}

R4 = {H,alokxy[1-6]}

R7 = {alkyl[1-4]}

$LIST_j$) in Figure 10 denotes a generic unit, say S_j.

Thus FRAG provides a various types of fragment records and gives a flexible screening system.

Example. The generic structure in Figure 5 is expanded to the three simple Markush formulas (Chart III) as shown in the previous section. Extended connectivity matrices for these SMFs are shown in Table IV, where each node is defined as shown in Chart IV. S_1 and S_2 are specific block nodes, and S_3, ..., S_8 are generic nodes and generic units.

Figure 11 shows fragment records corresponding to this generic structure. In this case, only the core simple blocks appear in the fragment records, so that the fields for extended simple blocks and superblocks are omitted in Figure 11. This is explained as follows. TFR gives all the specific simple blocks (i.e., H, Cl, ..., piperidyl), which is the logical sum of SFR_1, SFR_2, and SFR_3. SFR_1 shows that the blocks corresponding to on-bit positions are contained in the simple Markush formula SMF_1. SFR_2 and SFR_3 are explained similarly. H-LIST is prefixed by the numbers of identifiers of homologous series. Thus, H-$LIST_1$ means that SMF_1 contains no homologous series, H-$LIST_2$ means that SMF_2 contains a homologous series H_1 (alkyl[1-4]), and H-$LIST_3$ means that SMF_3 contains three homologous series H_1, H_2 (alkoxy[1-6]), and H_3 (cycloalkyl[3-6]). GUC has the same format as H-LIST except for the prefixed field indicating the relevant constant part. For example, GUC_1 indicates that the simple Markush formula (SMF_1) contains a constant part, CFR_1, and two generic units S_5 (=R_5 = {H, Cl, Br}) and S_6 (=R6 = {pyrrolidyl, piperidyl}). GUC_2 and GUC_3 are explained similarly. CFR_1 gives fragments of the constant part of SMF_1 and SMF_3, and CFR_2 corresponds to SMF_2. UFR_1 is a fragment record representing blocks in generic unit S_5, which is the set {H, Cl, Br}. Generic unit S_5 contains no homologous series, so that it is expressed symbolically as $S_5 = (UFR_1, \phi) = (UFR_1)$. Other generic units are explained similarly.

CONCLUSIONS

The internal representation of a Markush formula is presented for computer manipulation by extending the block-cutpoint tree technique. A fragment file (FRAG), an inverted file for homologous series (IF), an extended block-cutpoint tree for connectivity information (EBCT), and dictionaries for blocks, superblocks, and nomenclatural terms (BD, SBD, and ND) are provided for (sub)structure search. The key idea of the EBCT representation of generic chemical structures is to employ the appropriate portions of substructures as structural descriptors and to introduce a hierarchical view into generic chemical structures systematically. This is a natural extension of the BCT representation for specific chemical structures; that is, both specific and generic chemical structures are represented comprehensively by the block-cutpoint tree technique, which facilitates rapid and flexible (sub)structure search.

REFERENCES AND NOTES

(1) Rush, J. E. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 202.
(2) Nakayama, T.; Fujiwara, Y. *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 23.
(3) Ash, J. E.; Hyde, E. "Chemical Information Systems"; Ellis Harwood: Chichester, England, 1975.
(4) Kaback, S. M. *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 1.
(5) Silk, J. A. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 195.
(6) Lynch, M. F.; Barnard, J. M.; Welford, S. M. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 148.
(7) Barnard, J. M.; Lynch, M. F.; Welford, S. M. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 151.
(8) Welford, S. M.; Lynch, M. F.; Barnard, J. M. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 161.
(9) Kitchen, L.; Krishnamurthy, E. V. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 44.
(10) Aho, A. V.; Hopcroft, J. E.; Ullman, J. D. "The Design and Analysis of Computer Algorithm"; Addison-Wesley: Reading, MA, 1974.
(11) Harary, F. "Graph Theory"; Addison Wesley: Reading, MA, 1969.
(12) Fujiwara, Y.; Nakayama, T. *Anal. Chim. Acta* **1981**, *133*, 647.