Computer Storage and Retrieval of Generic Chemical Structures in Patents. 14. Fragment Generation from Generic Structures

J. D. Holliday, G. M. Downs, V. J. Gillet, and M. F. Lynch*

Department of Information Studies, University of Sheffield, Sheffield S10 2TN, U.K.

Received February 18, 1992

The generation of topological fragments from generic structures for full structure and substructure searching is described; these include fragments from components described either in specific or in generic terms, and those which overlap them. Fragments derived wholly from within partial structures (PS's) are termed intra-PS fragments, while those which span partial structures are termed inter-PS fragments. Where the generation of fragments from generic radicals is involved, the methods depend on searching the members of the fragment set against the intensional description of generic radical terms (or Homologous Series Identifiers, HSI), represented as default or explicit parameter values, to determine which fragments contribute to the description of the potential specific members of the homologous radical described by the HSI. High efficiency in screen generation is necessary because of the much higher number of screens generated from generic structures in comparison with specific structures and because of the complexity of the data structures involved. To achieve this efficiency in processing, a tree-structured dictionary of fragment types (augmented atoms and atom sequences) is created, against which potential fragments are matched, so that the appropriate level of description is correctly selected. Both of these fragment types are considered in this and the following paper. The interrelation of the fragments is facilitated by the Extended Connection Table Representation (ECTR) and by the bubble-up process which makes use of a logical matrix to represent the inter-PS relationships. Fragment screens are organized as a two-part vector in which one part indicates the presence of a fragment in an invariant feature of the structure (MUST screens), and the second (POSS screens) is formed from the union of the MUST fragments with those which are optional in the structure (the MAY screens).

1. INTRODUCTION

Substructure searching of chemical structure databases has, from the first, involved a two-stage search process, and hence some kind of screen searching, since the iterative search stage is inherently a greedy process.[1,2] Screening has had a central role in our work on generic structures, having been addressed first in the description of the overall strategy,[3] and later in the contexts both of fragment-based and of reduced graph methods for screening.[4] Indeed, it was recognized that, in view of the difficulties faced, not least the potentially infinite number of structures included by many generic expressions, it was important to make provision for orthogonal screening possibilities, i.e., screens which incorporate a range of characterizations of structural features, and include separate and distinct characteristics in alternative and complementary methods. Initial approaches to fragment screening were first described by Welford et al.;[5] the grammar-based approach identified there did not prove to be sufficiently general for application. Welford also described the logic for the storage and search of fragment-based screens for non-h-variant generic structure[6] (for terminology see Dethlefsen et al.[7,8]). An implementation of this approach for purposes of screening non-h-variant ring systems in generic structures has already been discussed by Downs, et al.[9]

The central problem of storing and searching generic structures is the need to establish equivalences between components of database structures and queries such that those which are described in intensional terms (h-variation, as described by Dethlefsen et al.,[8] such as 'alkyl', 'aryl', or 'carbocyclyl') can be matched against corresponding components described in terms of atoms and bonds. These equivalences need to fulfill or to be excluded by the necessary matching requirements.[8] In the extended connection table representation, the components are represented by 'partial structures' (PSs).

Fragment searching in the Chemical Abstracts Service MARPAT system has been described by Fisanick.[10,11] The approach which we have taken involves generating a limited variety of fragment screens which are common to the two types of expression (at the greatest level of detail practicable), so that the necessary matching relations between queries and file structures are accurately reflected in the screens and in the matching algorithms.[7,8] Here, queries comprise the following types:

1. specific full structure
2. specific substructure
3. generic full structure
4. generic substructure

Databases to which queries are addressed may, in principle, comprise either specific or generic structures. Here, only databases of generic structures are discussed.

The generation of topological fragments from generic structures, including those which exhibit h-variation, is described in this paper and that which follows.

Fragments which are explicit in the specifically described parts of the structures are generated by methods that are extensions of those used for specific structures, though with added complexities in consequence of the much more complex data structures, while those within generically expressed components are generated by using methods based on graph theory. The latter is the subject of the following paper in this series. Fragments which span two or more components are completed by linking part-fragments generated from their constituent components. The fragments are of the types first described by Adamson et al.[12] and extended by Graf et al. within the BASIC system,[13] by Feldmann et al.,[14] and by Dittmar et al.[15]

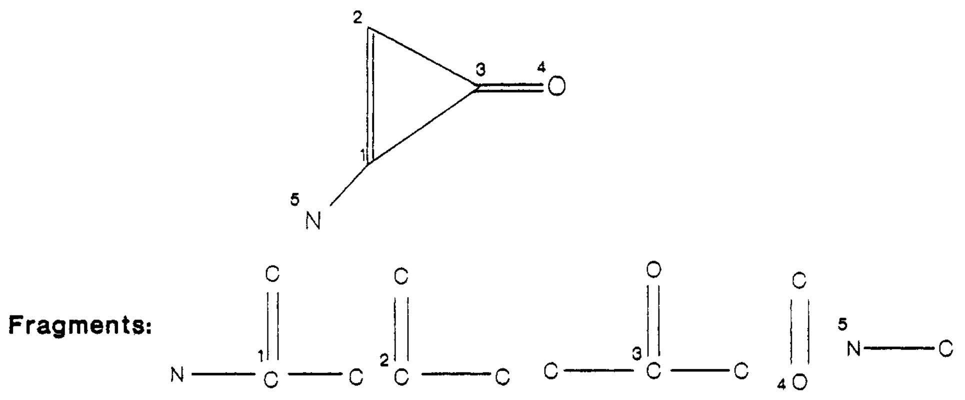


Figure 1. Augmented atom fragments.

The bubble-up method,[9] a means of accumulating fragment information on the components of a generic structure, is used during the accumulation of information to determine those fragments which are essential to a structure and those which are optional, on the basis of the distinction identified by Welford et al.[6] into **essential** (or MUST), i.e., fragments wholly occurring within invariant parts of the structures, and **optional** (or MAY) fragments, depending on whether they occur wholly or partly within optional components. A further distinction needs to be made about the nature of the searches, depending on whether full structure or substructure matching processes are involved.

The complexities to be overcome in any approach to fragment screening thus include dealing with the distinction between invariant and variable components, with optional substitution, as when one of the alternatives includes hydrogen as a value, and especially with intensional description of components, i.e., as generic radicals with or without explicit parameter values. In this case, we have chosen to regard fragment generation as a search for the possible occurrence of each of the fragments as a descriptor of any of the possible actual values which may be assumed by the generic variable. The approach adopted is based on the identification of default or actual parameter values for each generic radical term, as described by Dethlefsen et al.[8]

The fragment types investigated here are a subset of those used in CAS Online, the system operated by Chemical Abstracts Service through STN[16] but could equally well be derived automatically, according to well-established principles.[12,17,18] They are organized physically as a linear bit string in two components, the first containing the MUST or essential fragments and the second containing the POSS fragments, the logical union of the MUST and MAY fragments. As noted above, provision is also made for incorporation of the ring screens already described by Downs et al.[9] An implementation as an inverted file search system has also been undertaken by inversion of the bit screens.

There are obvious hazards in invoking fragment screens for generic structure searching, in particular, the fact that certain group designations such as 'radical', without further qualification, may result in the assignment of most or all possible fragments to a single generic structure, known to us as the **black** bit screen. This underlines the need for orthogonal screening techniques, where the weaknesses of one method may well be compensated for by the strengths of the others. In addition, the flexible methods underlying the representation of structures (in particular, the ECTR or Extended Connection Table Representation)[19] and their manipulation through the bubble-up method support the assignment of screens not merely to complete structures but also to components within the structures, so that limits can be placed on the hazard outlined above by limiting the black bit screen to the description of single components within a generic structure.

2. FRAGMENT SPECIES AND THEIR SPECIFICITY

The fragment types considered here comprise augmented atoms and sequence fragments, the latter including multiple types, lengths, and levels of description, according in part to the frequencies with which basic, fully explicit fragments are found in an initial analysis of a sample database. Other fragment types included in the CAS Online screen dictionary, in particular, connectivity sequences, were not considered likely to add discrimination in search in consequence of the wide latitude in substitution encountered in generic structures. The augmented atom fragment types considered here include the following:

1. Augmented atoms (AA), i.e., a central atom with its non-hydrogen congener atoms and the bonds to them

Linear sequences comprise the following:

1. Atom sequences (AS), paths of connected atoms of length 4–6 with bond information confined to the distinction between ring and chain bonds
2. Bond sequences (BS), paths of connected bonds, 3–5 bonds in length, without definition of the atoms which they connect

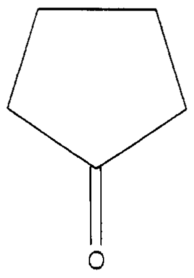
First, the fragment types are described, together with the need for an alternative representation of the screen dictionary which is regarded as being more appropriate for direct assignment than the more conventional list structure, in view of the greatly increased numbers of fragments to be assigned.

The descriptor types used in the bit screening search stage are thus augmented atoms, linear sequences, and ring descriptors, as noted above. The ring descriptors used, and their generation, have already been described by Downs et al.[9] Fragment descriptors form a closed set of 3300 elements, each of which has an associated bit position in a fixed-length representation, the bit string or the bit vector. In many cases the same bit represents more than one fragment; as a result the bit string contains only 1123 bits.

Figures 1 and 2 show a simplified representation of the types of fragment used. The true representations describe several levels of atom and bond definition, and as a result, a single path or central atom can produce a variety of different fragments, all described at different levels of specificity.

Atom definitions are of two types:

Specific definition uses the element symbols C, N, O, S, and X (X is a generic representation for any halogen).



Fragments:

O ═ C — C — C

O ═ C — C — C — C

O ═ C — C — C — C — C

C — C — C — C

C — C — C — C — C

Figure 2. Linear sequence fragments.

Table I. Bond Descriptor Codes

bond definition	code
nonspecific	
any	1
intermediate	
chain	2
ring	3
specific	
chain single	7
chain double	8
chain triple	9
ring single	11
ring double	12
ring triple	13
ring aromatic	14

Table II. Fragment Types in Screen Set

	bond definition		
	nonspecific	intermediate	specific
	Augmented Atoms		
nonspecific		bai	bas
specific	aan	aai	aas
	Sequences		
nonspecific		bsi	bss
specific	asn	asi	

Nonspecified definition uses the symbol *A* for all atoms (any atom).

Bond definitions are of three types:

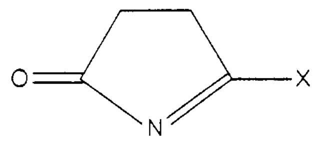
Specific definition uses a specific bond descriptor which shows the unsaturation and ring/chain nature of the bond.

Intermediate definition distinguishes between the ring and chain nature of the bond.

Nonspecified definition does not define the bond type (any bond).

The descriptor codes used to represent these bond types are given in Table I.

Levels of specificity in atom and bond definition are not mixed in fragments. If a fragment contains an intermediate level of bond definition and a specific atom definition, then descriptors from other levels of definition do not occur. The different fragment types which result from possible choices of bond and atom definitions are shown in Table II. Associated with each of these is a three-character code which describes the fragment type. The code is a standard notation for the



contains the sequence fragment (amongst others)

O ═ C — N ═ C — X

represented as **O 8 C 11 N 12 C 7 X**

reducing to **A 8 A 11 A 12 A 7 A**
O 2 C 3 N 3 C 2 X
A 2 A 3 A 3 A 2 A
O 1 C 1 N 1 C 1 X

Figure 3. Multiple levels of fragment definition.

remainder of this paper and that which follows, and each character is defined as follows:

1st character (atom definition)
- **a**, specific atom definition
- **b**, nonspecified atom definition

2nd character (linear or augmented fragment)
- **a**, augmented atom fragment
- **s**, linear sequence fragment

3rd character (bond definition)
- **s**, specific bond definition
- **i**, intermediate bond definition
- **n**, nonspecified bond definition

Note that no wholly specifically defined sequence fragments occur.

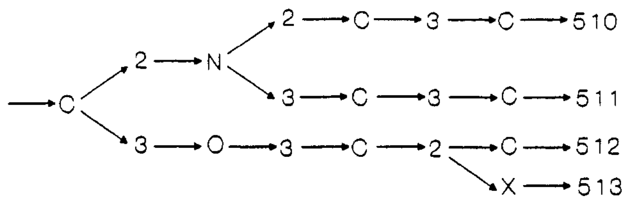
As a result of multiple levels of definition, a single fragment may produce further fragments by the process of specificity reduction. Thus, the sequence fragment in Figure 3 produces further fragments, illustrated, by this process.

3. TREE REPRESENTATION OF SCREEN DICTIONARY

As noted already, the multiplicity of fragments generated from generic structures as well as the complexity of the structures and of the generation routines demand high efficiency. Accordingly, an alternative to the conventional list-matching approach to screen generation was replaced by a tree-structured dictionary, the root of the tree pointing to the first atom in the fragment and the branches pointing to the atom types to which it is connected. The tree shown in Figure 4 is an example, representing the sequence fragments shown. The last atom of a complete fragment points to its bit position in the vector. The reasons for such a representation are

1. Once a candidate fragment has been generated, it must be located in the screen dictionary before being assigned in the bit screen. Using a list would require a sequential search of each record in order to locate the respective bit position. A tree structure is more readily searchable.
2. The readily searchable tree representation also means that, for each new atom and bond being added to the candidate fragment, examination of the next branch of the tree will instruct the generation algorithm to continue or to terminate.

The fragment generation process, described in detail in later sections, uses a PS-by-PS algorithm. All the fragments



510: C 2 N 2 C 3 C
511: C 2 N 3 C 3 C
512: C 3 O 3 C 2 C
513: C 3 O 3 C 2 X

Figure 4. Tree representation of sequence fragments of length 4 atoms.

contained within each PS which are found in the screen dictionary tree are generated for that PS. A **complete fragment** is a fragment which is found in the screen dictionary and has an associated bit position in the bit screen. If a complete fragment is contained wholly within one PS, then it is an **intra-PS** fragment. Fragments emanating from a PS may be incomplete, i.e., they form only a part of a complete fragment, the remainder of which may be contained in one or more adjoining PSs. A complete fragment which spans more than one PS is called an **inter-PS** fragment. Since all possible intra-PS fragments and inter-PS fragments must be generated, then all complete and incomplete fragments must be generated from each PS. For this purpose, the screen dictionary tree not only must represent the complete fragments but also must represent all possible incomplete fragments.

The screen dictionary tree comprises 49 different subtrees, each representing fragments (complete and incomplete) of varying lengths and of different fragment types. Augmented atoms are of five fragment types (see Table II) and of five possible sizes (0–4 congeners); there is a subtree representation for each of these 25 combinations. Linear sequence are of four different fragment types and six possible sizes (1–6 atoms); there is a subtree to represent each of these 24 combinations.

4. INTERNAL REPRESENTATION OF FRAGMENTS

The representation of complete fragments, once assigned, is simple; there is a correspondence, possibly many-to-one, between the fragment and its position in the bit screen.

The generation of incomplete fragments calls for development of the processes and representations detailed below, since the position is much less straightforward.

Each PS has an associated series of lists of incomplete fragments of varying sizes and types. Each record in the list contains the fragment descriptor, together with information describing its logical status and its location in the PS. The locations of the first and last atom of a sequence fragment or the central atom of an augmented atom fragment are described, since these atoms will be involved in further expansions of the fragment into neighboring PSs. The location of these atoms is described in terms of the atom numbers, the type of external connection (to child or parent), and the logical status of their position in the ECTR. Figure 5 illustrates the incomplete fragments, together with their location descriptors for the partial structure given. This information is used during the linkage of incomplete fragments.

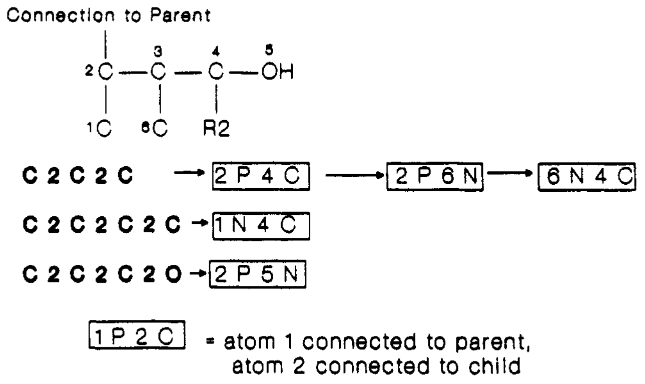

Connection to Parent
2C — 3C — 4C — 5OH
1C 6C R2
C 2 C 2 C → 2 P 4 C → 2 P 6 N → 6 N 4 C
C 2 C 2 C 2 C → 1 N 4 C
C 2 C 2 C 2 O → 2 P 5 N
1 P 2 C = atom 1 connected to parent, atom 2 connected to child

Figure 5. Fragment location descriptors.

5. GENERATION OF FRAGMENTS FROM SPECIFIC PARTIAL STRUCTURES

The process of fragment generation from specifically described PSs depends on the type of fragment being generated. For augmented atom fragments, each atom of the partial connection table is designated as a central atom; the congeners are then directly assigned from the associated connection table row. All possible reduced forms of the augmented atom are generated, so that the same central atom will produce all possible fragments of lesser connectivity which are contained within it. This is important for later expansions and for substructure search.

Sequence fragments begin with a connection table atom as the first atom, and further atoms are then added using a path-tracing technique.

5.1. Enumeration of Multiple Levels of Description. Each fragment generated usually contains specific atom and bond descriptors. These must be translated to produce the multiple levels of description found in the screen dictionary (see Table II).

Furthermore, some fragments may contain descriptors of lower specificity, resulting from variable bond definitions in the connection table. In such cases, all possible fragment descriptors of higher specificity must be enumerated by replacement of the bond with the equivalent specific bond descriptors. One *chain* bond in a fragment, for example, will produce the specific bond types *chaising, chaidoub, and chaitrip* and, therefore, three fragment descriptors at the higher specificity level. These fragments are automatically assigned the status POSS.

5.2. Sorting and Bit Assignment. Fragments are now sorted into the respective local lists of the PS being examined. The first process of sorting is the canonicalization of augmented atoms as it is the canonical form which is found in the dictionary tree. The respective screen dictionary subtree is examined to see whether the fragments occur in the dictionary. If so, and if they are complete fragments, then the associated bit is set in a local bit string. If they are found to occur as incomplete fragments in the subtree and they are extendable (i.e., augmented atoms with an externally connected focal atom or sequence fragments with at least one externally connected terminal atom), then they are sorted into a local list. There are 40 such lists, one for each level of description for each extendable size of fragment (9 of the 49 levels of description found in the screen dictionary tree (see section 3) are for complete, unextendable fragments).

6. GLOBAL FRAGMENT GENERATION PROCESS

The aim of the fragment generation program FRAGGENB is to process an entire generic structure represented as an ECTR

```
Procedure TreeTrace(PS);
var Info (information to be gathered)

      Procedure CbTrace(CBI; var Info);
      var CbInfo;
      Begin
      Initialise Info;
      for each CBI do
         begin
         case BottomBar of
            True : PsTrace(CBI∧.ChildPs, CbInfo);
            False: AbTrace(CBI∧.Alternatives, CbInfo);
         MergeFragments(Info, CbInfo);
         end
      End;

      Procedure AbTrace(ABI; var Info);
      var AbInfo;
      Begin
      Initialise Info;
      for each ABI do
         begin
         CbTrace(ABI∧.Combination, AbInfo);
         MergeFragments(Info, AbInfo);
         end
      End;

      Procedure PsTrace(PS; var Info);
      var Cginfo; Cg;
      Begin
      Process the PS; Initialise CgInfo;
      PS∧.ChildGate → Cg;
      for each Cg do
         begin
         case BottomBar of
            True : PsTrace(Cg∧.ChildPS, CgInfo);
            False: AbTrace(Cg∧.Alternatives, CgInfo);
         LinkFrags(Info, CgInfo)
         end
      End;

Begin
Initialise Info;
if PS ≠ nil then PsTrace(PS, Info)
End;
```

Figure 6. TreeTrace algorithm.

and to assign values to the required bits in the two bit screens, MUST and POSS. The ECTR is an AND/OR tree structure containing partial structures (PSs) at its leaf nodes as well as at some of its nonleaf nodes. 'AND' nodes of the tree are represented by *combination bars*, lists of Combination Bar Items (CBIs) each of which is in 'and' relation with other CBIs in the list. 'OR' nodes of the tree are represented by *alternative bars*, lists of Alternative Bar Items (ABIs) each of which is an alternative to other ABIs in the list. In order to accumulate the desired information for the whole structure, every branch of the tree must be examined, and the information gathered together to complete the picture. To do this, a depth-first tracing algorithm, *Tree Trace*, has been designed[6,20] to explore the tree using a 'bottom-up' methodology.

6.1. TreeTrace Algorithm. The TreeTrace algorithm, detailed in Figure 6, explores each PS, calling procedures which generate the complete and incomplete fragments as it goes. The process is recursive, each parent-PS/child-PS level of abstraction being dealt with in the same way. On accessing a PS, starting with the root PS, the process *PsProcess* is called which invokes the fragment generation procedures for either specific or generic PSs. The generation procedures, described above and in the following paper respectively, produce the local bit screens and fragment lists for fragments within that PS.

The *ChildGate* of the PS (which points to all child PSs) is then accessed, and each CBI is processed in turn. Since each CBI in the list headed by the ChildGate leads to PSs which are in combination with all other PSs headed by CBIs in that list, then each CBI in that list, called a *Top CBI*, is processed independently. The processing of the Top CBI begins with the generation of a logical matrix (see Section 6.2), which represents the logic between every child PS immediately below that CBI. Only immediate children are represented in the matrix, not 'grandchildren', since the information gathered in that child PS represents every PS below it.

Next, the child PSs are examined by tracing every path from the Top CBI in a depth-first manner, i.e., the *Next* field of an ABI or CBI is the last field accessed in all cases. When each child PS is accessed, the TreeTrace algorithm is called again, recursively, to examine the children of that child PS (i.e., the child PS now becomes the parent).

Once all child PSs have been visited, the information gathered and returned to the Top CBI is in three forms:

1. a list of all incomplete fragments generated from PSs below that CBI
2. an array of bit screens, each element containing a MUST and a POSS bit screen representing one of the child PSs
3. a logical matrix representing the logic between the child PSs

The next process is to assign logical values to each incomplete fragment in the list to describe its MUST or POSS status, since the logical relationships between incomplete fragments can only be determined once all child PSs have been visited. This process, described in Section 6.4, deals only with the incomplete fragments; the logical relationships between complete fragments, represented by the bit screens, must also be clarified. This is done using the logical matrix to merge all child PS bit screens into that of the parent PS so that the parent PS bit screen represents its own intra-PS fragments and those of its child PSs. Section 6.3 describes this operation.

Finally, incomplete fragments from the child PSs are linked, in accordance with the information which describes the inter-PS relationships in the ECTR, to the incomplete fragments of the parent PS to make inter-PS fragments, the logical status of which must also be deduced. These inter-PS fragments may be complete fragments, in which case the respective bits are assigned in the parent PS bit screen. They may also be incomplete fragments, in which case they are written to the respective parent PS list of incomplete fragments.

All Top CBIs are dealt with in the same manner, and the result of all recursions of the TreeTrace algorithm is the final bit screens of the root PS which represent the full generic structure. As noted earlier, this 'bottom-up' methodology for gathering information is called the **bubble-up.**[9]

6.2. AND/OR Trees and Logical Matrices. Generic structures contain a complex logical framework which describes the combinations of possible specific and generic partial structures which may make up a desired full structure. Some PSs may be in combination with each other, and some may be alternative to each other. The AND/OR tree can therefore be used to represent graphically the combinations of PSs which

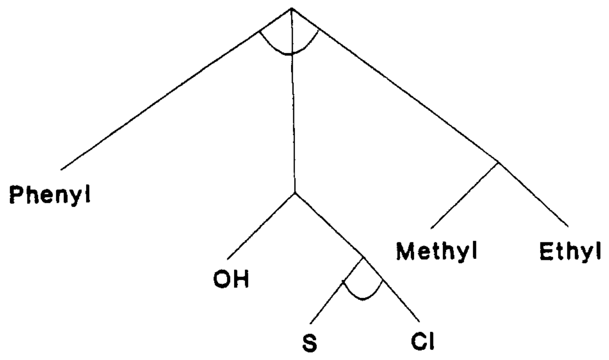


Figure 7. AND/OR tree representation of combinations.

go to make the generic structure. For example, the definition of two substituents on a phenyl group might be, in GENSAL:

R1 = OH/S SB Cl

R2 = methyl/ethyl

This definition describes the four combinations:

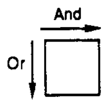
1. phenyl, hydroxyl, and methyl OR
2. phenyl, hydroxyl, and ethyl OR
3. phenyl, sulfur, chlorine, and methyl OR
4. phenyl, sulfur, chlorine, and ethyl

A more graphical representation in the form of an AND/OR tree is shown in Figure 7. *AND branches* are distinguished from *OR branches* by arcs.

The four combinations defined above can be expressed by a Boolean matrix denoting the presence or the absence of that structure in the combination. The table of these Boolean values is then:

choice	phenyl	hydroxyl	sulfur	chlorine	methyl	ethyl
1	1	1	0	0	1	0
2	1	1	0	0	0	1
3	1	0	1	1	1	0
4	1	0	1	1	0	1

In this table, 1 = present; 0 = absent. Each row represents an alternative combination of PSs to the other rows. Each column representing a PS is in combination with each other column containing the Boolean value of 'true'. Schematically



6.3. Merging of Bit Screens. The process of merging the bit screens, each one representing the complete fragments of a child PS, is carried out using the logical matrix.

Each row in the matrix contains the references to the bit screen representing a child PS, and each bit screen so referenced by that row is in combination with every other bit screen so referenced. A procedure *CombMatrices* combines all of these bit screens into one bit screen which represents the row. The MUST screen of this bit screen is the union of all the MUST screens of the bit screens referenced by the row, and the POSS screen is the union of all the POSS screens of the bit screens referenced by the row.

Each row is an alternative to every other row, and the operation on these new bit screens is therefore different. A procedure *AddMatrices* combines all of these new bit screens representing the rows into one bit screen representing the whole matrix. This time the MUST screen of the final bit screen

```
Function AndMatrices(M1, M2 : tBitScreen): tBitScreen;

Begin
M1.Poss ∪ M2.Poss → AndMatrices.Poss;
M1.Must ∪ M2.Must → AndMatrices.Must
End;

Function OrMatrices(M1, M2 : tBitScreen): tBitScreen;

Begin
M1.Poss ∪ M2.Poss → AndMatrices.Poss;
M1.Must ∩ M2.Must → AndMatrices.Must
End;

Procedure LinkMatrices(var BitScreen : tBitScreen);

    Function CombMatrices: tBitScreen;

    Begin
    Initialise CombMatrices;
    For each column where PS found in row do
       AndMatrices(CombMatrices, Bitscreen for PS) → CombMatrices
    End;

    Function AddMatrices: tBitScreen;

    begin
    Initialise AddMatrices;
    For each Matrix row do
       OrMatrices(AddMatrices, CombMatrices(Row)) → AddMatrices
    End;

Begin
   AndMatrices(BitScreen, AddMatrices(Matrix)) → BitScreen
End;
```

Figure 8. Algorithm for merging bit screens.

is the intersection of the MUST screens representing each row since only those bits common to all rows may be assigned in the MUST screen. The POSS screen of the final bit screen is the union of the POSS screens representing the rows. An algorithm for the processing of the bit screens is given in Figure 8.

The result is one bit screen, containing a MUST screen and a POSS screen, which represents all of the bit screens below the respective Top CBI. Since the PSs represented by the Top CBI are in combination with the parent PS, then the bit screens for the Top CBI are combined with those of the parent PS, union of MUST and union of POSS, to complete the processing of the Top CBI.

6.4. Assignment of Incomplete Fragments. Two types of flags must be assigned to describe the status of the fragment. The first, the *Status* field of the fragment descriptor itself, describes the overall status of all occurrences of the fragment. The other type of flag is local to each individual occurrence of the fragment and describes the status according to the position in the ECTR (see Section 4).

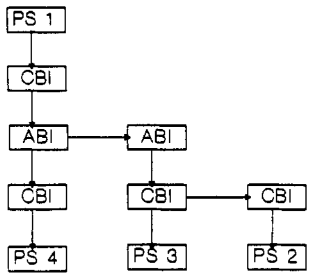
The local status fields are assigned according to the variability of the connections to the fragment and according to the nature of the fragment's derivation. In the first case, the connections to the fragment (to the focal atom of an augmented atom or to the front or back atoms, i.e., terminal atoms, of a sequence fragment) have the respective local status value assigned **false** if they are derived from extensions through inter-PS connections of a variable nature, i.e., if there is a choice of atoms by which the PSs are connected or a choice of bonds by which they are connected (see Section 6.6). This is determined by examination of the positional information fields of the inter-PS links in the ECTR.

The assignment of the local status field is also dependent on the type of PS from which the fragment is derived, the second case above. This is **false** for a generic PS and usually **true** for a specific PS. Speeific PSs may generate POSS

GENSAL

```
PIPERAZINYL SB ( [2] ADAMAMANTYL /
                 ( [3-4] T-BUTYL & [2] CYCLOHEXYL ) );
```

ECTR



Logical Matrix

2	3
4	

Figure 9. Assignment of status to fragments emanating from part of the ECTR.

fragments as a result of enumeration of fragments of high specificity from descriptions of low specificity (see Section 5.1).

The local status fields are assigned **true** only if they are *true* in both cases, constituent connections and original derivation.

The assignment of the Status field of the fragment descriptor is carried out after fragments have been generated from all child PSs below a Top CBI. The assignment is done using the logical matrix in a manner similar to bit screen merging, using the local status fields of all occurrences of the fragment. In order that a fragment descriptor is assigned the value **true**, it must be found at least once as a MUST fragment in every possible combination of PSs. In other words, at least one occurrence record must represent that fragment in MUST local status in a PS of each row of the logical matrix.

Considering the section of the ECTR shown in Figure 9, where PS 3 has a variable connection to PS 1, and the logical matrix which describes the relationship, the generation of the augmented atom fragments might be as follows (local status given):

(i) C 3 C 3 C	generated as MUST from PSs 2 and 4
(ii) C 1 C 1 C 1 C	generated as MUST from PS 4 and POSS from PS 3
(iii) C 1 C 1 C	generated as MUST from PSs 2 and 4 and POSS from PS 3

Local MUST occurrences appear for fragment i in both rows of the logical matrix, and its Status field is therefore set to **true**. Local MUST occurrences appear for fragment iii in both rows of the logical matrix, and its Status field is therefore set to **true**. Local MUST occurrences only appear in one row of the logical matrix for fragment ii; its occurrence in PS 3 is POSS. The Status field is therefore set to **false**. This is because the extension of this incomplete fragment into the parent may proceed through any atom in the parent to produce a variety of possible fragments, none of which can be regarded as MUST.

6.5. Linking of Incomplete Fragments. Incomplete fragments must now be linked together, as appropriate, to produce inter-PS fragments. Linkage is performed by using the two lists of incomplete fragments: one of which, *PsInfo*, contains all incomplete fragments generated from the parent PS, and the other, *CgInfo*, contains all incomplete fragments generated from all child PSs below the Top CBI.

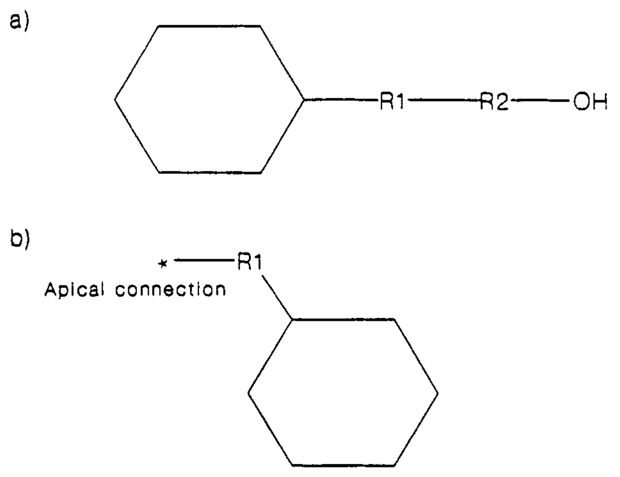
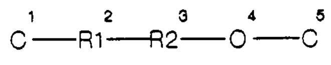


Figure 10. Externally connected substituent identifiers.



$R1 = N;^{1}$

$R2 = CH_2.^{1}$

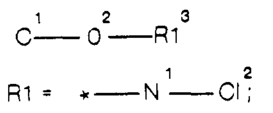
PsInfo:	C	Ps1 Row1, Ps1 Row3
	O	Ps1 Row4
	C-O	Ps1 Row4-5
	N	Ps1 Row2
CgInfo:	N	Ps2 Row1
	C	Ps3 Row1

Figure 11. Sorting part-fragments into the parent.

Two incomplete fragments or part-fragments, one from the PsInfo list and one from the CgInfo list, are examined to see if they are adjacent anywhere in the ECTR and, if so, then they are linked together. In some cases, two part-fragments emanating from two child PSs may be adjacent in the ECTR. In such cases, where two nodes of a parent PS represent two adjacent substituent identifiers (as in Figure 10a), then the fragments from the child PSs must be sorted into the respective PsInfo lists so that the linkage can be carried out. These fragments are also sorted in this way if they emanate from substituent identifiers which are connected apically (i.e., externally to the parent PS as in Figure 10b). The connection table row, in the parent PS, of the substituent identifier is examined to determine if any of its neighbors are either substituent identifiers or apical connections. If so, then the fragments generated from the child PSs represented by the substituent identifier are sorted into the PsInfo list.

A new record is held in the PsInfo list for the fragment, describing its position of occurrence as the substituent identifier's row in the parent PS, as in Figure 11. In effect, the part-fragment is represented by a 'superatom' in the parent. The neighboring part-fragments from the child PS can then be linked to this superatom as if it were a part-fragment from the parent PS.

Part-fragments are systematically linked to create new fragments of varying size. The process is the enumeration of all possible combinations of fragments in all possible positions of occurrence. Every occurrence of every fragment from the PsInfo lists is tested against every occurrence of every fragment from the CgInfo lists, provided that the maximum size is not exceeded and that all fragments are of the same description type. If the fragments are found to be adjacent in the ECTR,



```
PsInfo: O    Ps1 Row2
        C-O  Ps1 Row1-2
CgInfo: N    Ps2 Row1
        N-Cl Ps2 Row1-2

New fragments:
        O-N    Ps1 Row2-3
      C-O-N    Ps1 Row1-3
      O-N-Cl   Ps1 Row2-3
    C-O-N-Cl   Ps1 Row1-3
```

Figure 12. Linking of fragments and assignment of occurrence descriptions.

then they combine to form a new fragment which is sorted into the respective PsInfo list. When such a fragment is sorted, its position is described as if it came from the parent PS.

Figure 12 shows two part-fragments (sequences) producing new fragments together with their new occurrence description. (Note that the Cl and C atoms alone do not produce part-fragments as they are not externally connected.) The sorting of the new fragment is dependent on whether it is found in the screen dictionary tree. If it is not found, then it is not sorted; if it is found as an incomplete fragment, then it is sorted into the respective PsInfo list, and if it is found as a complete fragment, then its associated bit position is set in the bit screen of the parent PS.

Once all fragments from the CgInfo lists emanating from the Top CBI have been linked, the CgInfo lists are deleted.

6.5.1. Linkage of Augmented Atom Fragments. The linkage of augmented atom fragments is a process of adding congener atoms where possible to either the part-fragments of the PsInfo lists or the part-fragments of the CgInfo lists. The congener atoms to be added to the PsInfo lists are the equivalent-level central atoms of the CgInfo lists, and vice versa, as these are the complete list of single atoms (focal atoms) which may be used as congeners.

Processing starts with each fragment in the CgInfo lists as a focal fragment. Each single atom of the PsInfo list is tested, using the inter-PS positional information of the ECTR, to see if it may be a congener to this focal fragment and then tested, using the logical matrix, to see if it is in AND relation to other PSs which may have contributed to the derivation of the fragment. Where two part-fragments are found to be neighbors, then a new fragment is generated and sorted into the PsInfo list accordingly (and/or its bit is set). This process is repeated using each fragment in the PsInfo lists as a focal fragment and each single atom of the CgInfo lists as the congener.

6.5.2. Linkage of Sequence Fragments. Due to the nature of sequence fragments, it is often necessary to link together three part-fragments at the same time to produce a new fragment. In most cases only two part-fragments are used, but in certain cases, particularly with doubly-connected substituents, a fragment may span more than one boundary between PSs. Doubly-connected substituents are processed separately from singly-connected substituents in order to link three part-fragments together. This is because the double-connection inter-PS information in the ECTR describes both connections between the parent PS and the child PS at the same time.

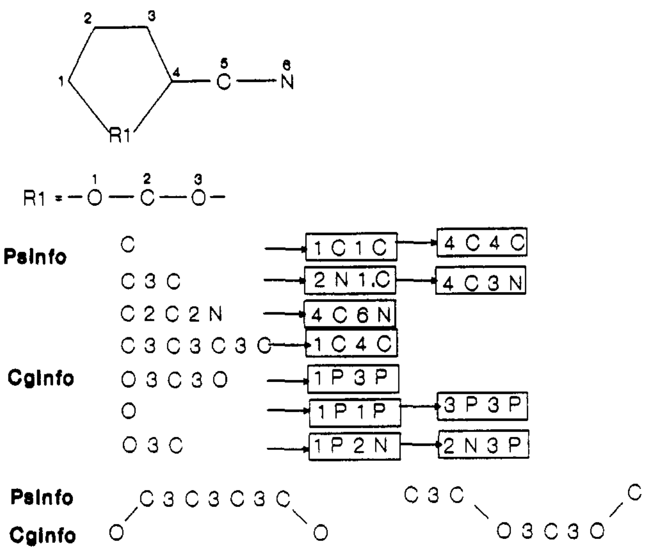


Figure 13. Linkage of doubly-connected sequence fragments.

Processing starts by linking a CgInfo part-fragment to two PsInfo part-fragments; the CgInfo part-fragment is the central part-fragment in the new fragment being generated. In the case of a doubly-connected linkage, the inter-PS positional information describes both connections. In the case of a singly-connected linkage, where one connection only is described, only two part-fragments are effectively linked by using a dummy part-fragment of length zero for the second PsInfo part-fragment. Again, every occurrence of these fragments is tested, and any new fragments created are sorted into the respective PsInfo lists and/or their bits are set accordingly. This process is repeated using a PsInfo part-fragment as the central part-fragment linking to two CgInfo part-fragments. Only doubly-connected linkage is carried out in this second stage; the dummy fragment is not used, since singly-connected linkage has been carried out in the first stage. The two CgInfo part-fragments will both originate from the same child PS.

Figure 13 illustrates the occurrence records and linkage of part-fragments from a doubly-connected substituent to its parent PS.

6.6. Logical Assignment of Inter-PS Fragments. Inter-PS fragments result from the cumulation of part-fragments emanating from more than one PS. The logical status of the inter-PS fragment is dependent on the local status fields of its constituent part-fragments. Each local status field is examined together with the Status field of the fragment descriptor. A local Boolean value, *Sts*, is used which reflects the status of the bond (or bonds) connecting the constituent part-fragments. This value may be set to **false** if the bond is variable or if there are optional positions of attachment in the parent or child. The final status of the inter-PS fragment is **true** only if all constituent local status fields and constituent fragment descriptor Status fields and Sts are **true**. This does not include the local status fields representing the atoms which are not involved in the connection. The concatenation of two incomplete sequence fragments uses one atom from each constituent fragment; logical assignment therefore requires only the examination of the two respective local status fields.

Those local status fields not involved in the above assignment are used to assign the respective local status fields of the inter-PS fragments. Figure 14 illustrates the values resulting from the concatenation of part-fragments.

In the first example, the bond connecting the two part-fragments is nonvariable, the Sts field is therefore **true** (MUST); the local status fields involved in the connection are

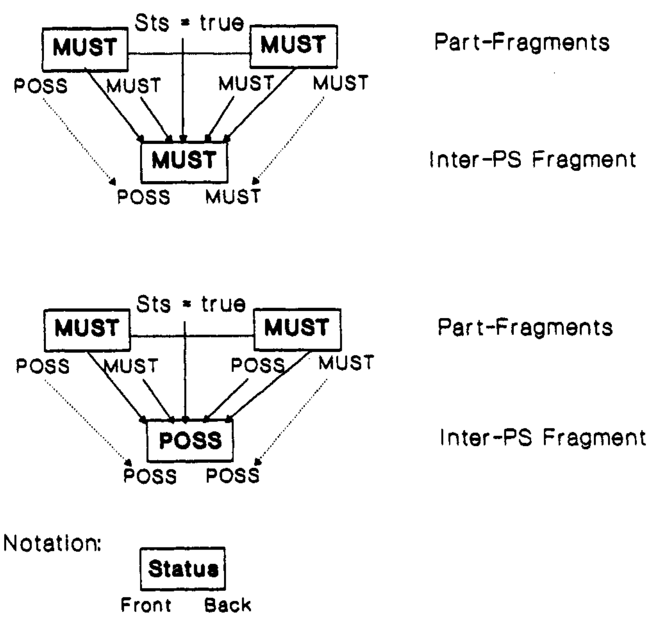


Figure 14. Status assignment due to concatenation of part-fragments.

also both **true** (MUST). The resulting inter-PS fragment, therefore, has its Status field set to **true** (MUST). The two local status fields of the inter-PS fragment are then assigned according to the respective local status fields not involved in the connection: *Front* becomes **false** and *Back* becomes **true**. The second example produces a POSS inter-PS fragment due to the **false** local status field involved in the connection. Since the fragment produced has a POSS Status, then both local status fields are assigned **false**.

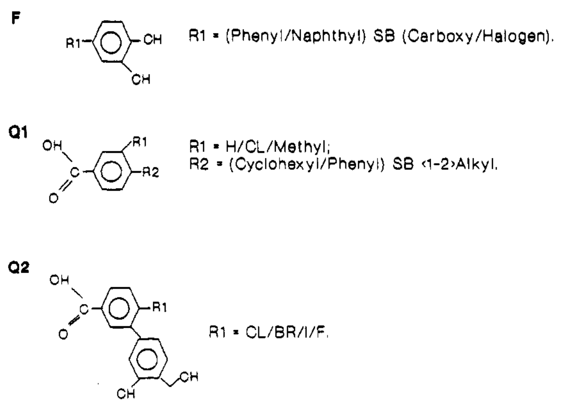
6.7. Bit Screen Assignment. The relevant screen dictionary tree is examined to determine the existence of the inter-PS or intra-PS fragment in the screen dictionary. If the fragment exists, then it may be a complete fragment, in which case it has an associated bit screen position (or vector). If so, then the fragment descriptor Status field is examined in order to determine whether the fragment is MUST, **true**, or POSS, **false**. If the fragment is MUST, then the respective bits in both the MUST and POSS screens of the PS are set to **true**, if not, then only the respective bit in the POSS screen of the PS is set to **true**. The initial values of these bits does not affect this operation.

7. MATCHING

The criteria for matching a generic query structure with a generic file structure using the two-component bit screens have been described by Welford et al.[6] For a full structure match, the MUST screen of the **query** structure must be a subset of the POSS screen of the **file** structure *AND* the MUST screen of the **file** structure must be a subset of the POSS screen of the **query** structure.

Figure 15 illustrates a query match (Q1) and a query mismatch (Q2) for a full structure search against the same file structure (F) using just five sequence fragments. (The structures are given in GENSAL.) Note the 'bubble-up' of some fragments into the MUST screen; fragment i of structure F and fragment ii of the structure Q2.

For substructure search, the first condition only, that the MUST screen of the **query** structure must be a subset of the POSS screen of the **file** structure, applies to identify a match. The query Q2 does not, therefore, give a substructure match against the file structure F.


F
R1
CH
CH
R1 = (Phenyl/Naphthyl) SB (Carboxy/Halogen).
Q1
OH
C
O
R1
R2
R1 = H/CL/Methyl;
R2 = (Cyclohexyl/Phenyl) SB <1-2>Alkyl.
Q2
OH
C
O
R1
R1 = CL/BR/I/F.
CH
CH

Sequence Fragments	F		Q1		Q2	
	MUST	POSS	MUST	POSS	MUST	POSS
(i) A 14 A 7 A 14 A	1	1	0	1	1	1
(ii) C 3 C 3 C 2 X	0	1	0	1	1	1
(iii) C 3 C 2 C 2 O	0	1	1	1	1	1
(iv) C 3 C 2 C 2 C	0	0	0	1	1	1
(v) C 2 C 3 C 2 C	1	1	0	1	0	0
where A = anyatom, X = halogen and bond codes are given in Table 1						

Figure 15. Example searches against file structure F.

8. CONCLUSION

The generation of topological fragments from generic structures has been described. The generation uses a PS-by-PS methodology, the incomplete fragments of neighboring PSs being linked together using the 'bubble-up' technique. The generation of fragments from the individual PSs is dependent on the type of PS description: specific or generic. The generation from specifically described PSs has been described; generation from generically described PSs is the subject of the following paper.

The result is a two-part bit screen, representing the possible and essential features attributable to the generic structure. These bit screens are used as a preliminary screening stage in the overall search strategy. The performance of such a search will be the subject of a future publication.

ACKNOWLEDGMENT

We gratefully acknowledge funding from International Documentation in Chemistry mbH, Derwent Publications Ltd, and Questel SA in support of the research described here and thank Dr. W. Dethlefsen of BASF, Professor P. Willett, and Dr. J. M. Barnard of Barnard Chemical Information Ltd. for generous advice. We also thank Chemical Abstracts Service for provision of documentation on the screen sets.

REFERENCES AND NOTES

(1) Ash, J. E.; Chubb, P. A.; Ward, S. E.; Welford, S. M.; Willett, P. *Communication, Storage and Retrieval of Chemical Information*. Ellis Horwood Ltd.: Chichester, 1985.
(2) Ash, J. E.; Warr, W. A.; Willett, P. *Chemical Structure Systems*. Ellis Horwood Ltd.: Chichester, 1991.
(3) 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.
(4) Gillet, V. J.; Downs, G. M.; Holliday, J. D.; Lynch, M. F.; Dethlefsen, W. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 13. Reduced Graph Generation. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 260–270.
(5) Welford, S. M.; Barnard, J. M.; Lynch, M. F. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 3. Chemical Grammars and Their Role in the Manipulation of Generic Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 161–168.
(6) Welford, S. M.; Barnard, J. M.; Lynch, M. F. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 5. Algorithmic Generation of Fragment Descriptors for Generic Structure Screening. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 57–66.

(7) Dethlefsen, W.; Lynch, M. F.; Gillet, V. J.; Downs, G. M.; Holliday, J. D.; Barnard, J. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 11. Theoretical Aspects of the Use of Structure Languages in a Retrieval System. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 233–253.

(8) Dethlefsen, W.; Lynch, M. F.; Gillet, V. J.; Downs, G. M.; Holliday, J. D.; Barnard, J. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 12. Principles of Search Operations Involving Parameter Lists: Matching-Relations, User-Defined Match Levels, and Transition from the Reduced Graph Search to the Refined Search. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 253–260.

(9) Downs, G. M.; Gillet, V. J.; Holliday, J. D.; Lynch, M. F. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 10. The Generation and Logical Bubble-up of Ring-screens for Structurally Explicit Generics. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 215–224.

(10) Fisanick, W. Storage and Retrieval of Generic Chemical Structures in Patents. U.S. Patent 4642762, Feb 1987.

(11) Fisanick, W. The Chemical Abstracts Service Generic Chemical (Markush) Structure Storage and Retrieval Capability. 1. Basic Concepts. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 145–155.

(12) Adamson, G. W.; Cowell, J.; Lynch, M. F.; McLure, A. H. W.; Town, W. G.; Yapp, A. M. Strategic Considerations in the Design of a Screening System for Substructure Searches of Chemical Structure Files. *J. Chem. Doc.* **1973**, *13*, 153–157.

(13) Graf, W.; Kaindl, H. K.; Warszawski, R. The Third BASIC Fragment Search Dictionary. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 177–181.

(14) Feldmann, R. J.; Milne, G. W. A.; Heller, S. R.; Fein, A.; Miller, J. A.; Koch, B. An Interactive Substructure Search System. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 157–163.

(15) Dittmar, P. G.; Farmer, N. A.; Fisanick, W.; Haines, R. C.; Mockus, J. The CAS ONLINE Search System. 1. General System Design and Selection, Generation, and Use of Search Screens. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 93–102.

(16) *CAS online screen dictionary for substructure search*, 2nd ed.; American Chemical Society: Washington, DC, 1981.

(17) Lynch, M. F.; Harrison, J. M.; Town, W. G.; Ash, J. E. *Computer Handling of Chemical Structure Information*; Elsevier: London, 1971.

(18) Lynch, M. F. Screening Large Chemical Files. In J. E. Ash and E. Hyde, *Chemical Information Systems*; Ash, J. E., Hyde, E., Eds.; Ellis Horwood Ltd.: Chichester, 1975; pp 177–194.

(19) 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.

(20) Barnard, J. M. Extended Connection Table Representation, (Version2.0). Technical Description. Technical Report, Barnard Chemical Information Ltd.: Sheffield, 1987.