Irredundant Generation of Isomeric Molecular Structures with Some Known Fragments

Marina S. Molchanova and Nikolai S. Zefirov*

N.D. Zelinsky Institute of Organic Chemistry, Leninsky Prospekt 47, Moscow, 117913 Russia

Received February 27, 1997⊗

The problem of exhaustive and irredundant generation was solved for the case of organic structural isomers sharing the same molecular formula and containing some predefined fragments—a set of arbitrary "core" fragments that are known in advance and may be used as ready construction units for generation along with separate atoms. The algorithm was implemented in the SMOG software, which had been described in an earlier publication in this Journal.[1] Some examples are presented to illustrate the correctness and performance of the algorithm.

1. REDUNDANCY PROBLEM IN GENERATION

1.1. Generators of Structural Isomers. During the past 3 decades, since 1965−1969,[2] an increasing number of computer programs for generation of structural isomers on the basis of the given molecular (gross) formula have been designed and created for various purposes: CONGEN[3] and GENOA[4] within the DENDRAL project,[5] ASSEMBLE,[6] COCOA,[7] ISOGEN,[8] MOLGEN,[9] CHEMICS,[10] IGOR and RAIN (in the structure-generating mode),[11] GEN,[12] GENM,[13] AEGIS,[14] CAMGEC,[15] ESOSOC,[16] programs by Faulon[17,18] and Bangov,[19] *etc*. The majority of these generators, except for COCOA,[7] are fully or partially based on the principle of *structure assembly*:[6a] that is, complete molecules are constructed from atoms and/or multiatomic structural units by successive consideration of different ways of joining these atoms and units together by chemical bonds.

In addition to the molecular formula, the overwhelming majority of isomer generators provide the possibility of using structural fragments (substructures) as constraints: that is, the user may specify the lists of fragments that should be present in the target structures or, conversely, should be absent from them (GOODLIST and BADLIST, respectively[2,3a]). This feature makes such generators applicable for the solution of various practical problems in organic chemistry. One of the most common tasks favoring the use of generators is the problem of computer-assisted structure elucidation.[20,21] As well as some other problems (such as design of new compounds with the desired properties), it may frequently be reduced to the same task: generation of molecular structures that conform to some known gross formula and contain (or do not contain) several definite fragments.

1.2. Consideration of GOODLIST: Core Fragments. We consider generation of isomers in the case when GOODLIST contains some nonoverlapping multiatomic structural fragments. Two approaches are possible. One of them (the straightforward one) is to generate all isomers for the given molecular formula, assembling them from separate atoms, and then to exclude those which do not contain any of the fragments from GOODLIST. Evidently, this approach is usually totally inefficient: the majority of the resultant structures probably will have to be rejected after they are

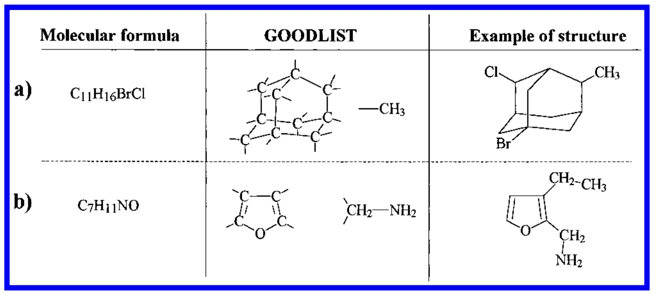


Figure 1. Generation of isomers with known GOODLIST. Hereafter, bonds with an unspecified terminus denote free valences.

generated. Another approach (presumably it was first developed in the CONGEN generator within the DENDRAL project[3]) appears to be much more promising: we may use some or all fragments from GOODLIST as ready "building blocks" for assembly (in our terminology, such fragments are called *core fragments*[22]) along with the remaining atoms. Then, all of the resultant structures will automatically contain the fragments that were used as building blocks (construction units).

To illustrate this statement, we consider two simple examples. Suppose we have to generate all bromochloromethyladamantanes. The straightforward solution would be to generate all of the 5 758 744 structures (our result) that conform to the molecular formula $C_{11}H_{16}BrCl$ and then exclude those which do not contain the adamantane skeleton or the methyl group (Figure 1a). The much more satisfactory approach, however, is to use the adamantane fragment as the ready construction unit and then join the remaining atoms and groups (Br, Cl, CH_3) to it in all possible ways. Then, only 48 isomers (in complete agreement with Pólya's theorem[23]) are generated, and the vast diversity of unsatisfactory structures that do not represent substituted adamantanes are automatically omitted. Another example is generation of all $C_7H_{11}NO$ isomers that contain the furan ring and the CH_2-NH_2 fragment. Using the core fragments shown in Figure 1b, we may combine them with each other and with the remaining atoms (2 carbon atoms and 7 hydrogens) in different ways and obtain a few as 22 isomers, whereas the total number of $C_7H_{11}NO$ isomers is 174 763.[9a]

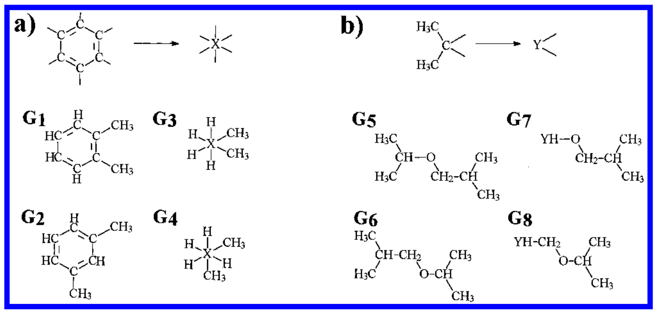


Figure 2. Complications associated with substitution of fragments by imaginary atoms during generation.

However, assembly of molecules from both atoms and fragments, efficient as it is, entails some additional combinatorial difficulties. We comment on this statement.

As a rule, a structural generator is oriented at exhaustive and irredundant generation (the only exceptions are different kinds of stochastic generators,[18] which are not considered further in this paper), and avoiding the appearance of duplicates among the structures produced by the program is an especially difficult task. This problem is due to graph isomorphism:[24,25] for each molecule, the program may produce the whole family of isomorphic molecular graphs representing it. Since, according to the definition of isomorphism, these graphs differ only by the order of vertex numbering, exactly one graph out of each isomorphic family should be left among the results, whereas all others should be discarded. Some efficient approaches have been designed to find and reject isomorphic graphs as early during generation as possible. However, one may note that some of these approaches[13b] completely exclude the formation of duplicates only when a molecule is assembled from separate atoms: the use of fragments as construction units often leads to the appearance of duplicates[15] or is not considered at all.[8,26] In a number of other generators, only specific kinds of fragments are rigorously considered during assembly: for example, the CHEMICS generator[10] makes explicit use only of fragments that contain exactly one atom with free valences,[27] and the deterministic version of Faulon's program SIGNATURE[17] ensures strictly duplicate-free generation only in the case of "ideal" graphs, where chemical bonds between the fragments in question must be topologically nonequivalent to bonds inside these fragments. However, comparatively few authors of structure-generating software attempt to provide a solution to the more general problem—that is, rigorous irredundant generation making use of *arbitrary* multiatomic fragments.

To illustrate the complications associated with the problem of fragment usage, we analyze one seemingly evident approach to its solution. It might seem that an appropriate way of incorporating fragments directly into target molecules is treatment of these fragments as a kind of "large atoms". For example, a benzene ring, having six free valences, might be regarded as an imaginary hexavalent atom X (Figure 2a). However, even brief analysis shows that this representation is not fully acceptable for determination of isomorphic structures, if only because the sets of free valences possessed by the fragment in question and by a hexavalent atom have different automorphism groups.[28] Indeed, graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ in Figure 2a (generated for the molecular formula C_8H_{10}) are nonisomorphic, whereas the corresponding graphs $\mathbf{G}_3$ and $\mathbf{G}_4$ (generated for the molecular formula $C_2H_{10}X$, where the benzene ring C_6 is substituted by an X atom) are isomorphic.

In some fragments, the automorphism group of the set of free valences may be equivalent to that of a single atom. The most common case of such symmetry is when only one atom in the fragment has free valences, whereas all valences of all other atoms are saturated (a *simple fragment*, according to the terminology of ref 13c). For example, as is shown in Figure 2b, the set of free valences of the $(CH_3)_2C$ fragment has the same symmetry as that of an imaginary bivalent atom Y. However, representation of this fragment by the Y atom may prove unsatisfactory as well, if another copy of the same fragment may be assembled during generation from the remaining C and H atoms. For example, we compare generation for the molecular formula $C_7H_{16}O$ with $(CH_3)_2C$ in GOODLIST and generation for the formula $C_4H_{10}OY$, where the $(CH_3)_2C$ fragment is substituted by the bivalent atom Y. Graphs $\mathbf{G}_5$ and $\mathbf{G}_6$ in Figure 2b, generated for the former problem, are isomorphic, in contrast to the corresponding graphs $\mathbf{G}_7$ and $\mathbf{G}_8$ generated for the latter one.

That is, if a core fragment is simply substituted by an imaginary atom, either some correct structures may be lost (when different graphs are taken for isomorphic ones) or, vice versa, some duplicates may actually occur in the set of results.

One of the possible ways out is to supplement such "imaginary-atom" substitution with additional procedures for symmetry perception, as in the frequently used "generate—embed" approach (for example, this approach is implemented in the CONGEN[3] and GENOA[4] generators within the DENDRAL[5,21] project or in the MOLGEN/MOLGEN+ software,[9] where some algorithmic features of DENDRAL are used). That is, core fragments (referred to as macroatoms in MOLGEN) are denoted by imaginary atoms only at intermediate stages of generation; after generation of each molecular structure containing such atoms is completed, they are expanded—that is, the actual structures of fragments are substituted instead of the imaginary atoms, with explicit regard made for the symmetry of the fragments. This consideration of symmetry makes it possible to avoid the problem represented in Figure 2a. However, another difficulty (that is, assembly of other copies of core fragments from the remaining atoms and/or other core fragments; see Figure 2b) is not so easy to avoid: the structure of the complete molecule after expansion of fragments has to be rigorously analyzed if we aim at total exclusion of duplicates.[29]

Although the "generate—embed" approach appears to enhance the efficiency of generation (note that MOLGEN is presumably the most efficient generator among those used at present!), this technique is also characterized by some shortcomings, even if we disregard the aforementioned additional procedures at the final stage. For example, problems that are encountered in chemical practice usually necessitate the use of other constraints (in addition to core fragments) during generation: BADLIST, hybridization states and local environments of atoms, structure of the ring system, local and global topological symmetry, *etc*.[1] Then, it may be more difficult to ensure efficient "interplay" of these constraints[7] (and thus to avoid unsatisfactory intermediate structures at the earliest possible stages of generation) if core fragments are represented by some imaginary atoms rather than directly included into molecules as multiatomic units of known actual structure. So, development of some techniques that would use direct inclusion of fragments into target structures during generation (not only after it!) is a problem of both theoretical and practical interest.

An important step in this direction was undertaken by Molodtsov, who developed a highly efficient generator GENM.[13b,c] The algorithm of GENM, which inherited a number of features from Faradjev's algorithm of graph generation,[30] ensures fast and exhaustive construction of isomers for any molecular formula.[31] Core fragments, if any, are indeed directly included into target structures as a whole at the very beginning of generation (accordingly, this approach may be referred to as "embed−generate" rather than "generate−embed"), and their symmetry and free valences are explicitly considered. However, the problem of irredundant generation in the presence of core fragments is only partially solved in GENM through the use of semiheuristic procedures:[13c] in some cases, duplicates are indeed removed completely; in other cases, the fraction of duplicates among the results may even exceed 50%. That is, some retrospective check for the presence of remaining duplicates is still necessary, exactly as in many other generators that use fragments for structure assembly (for example, the necessity of such checking is admitted by the authors of GEN[12]).

Actually, a retrospective check for the presence of some remaining duplicates is not a fatal disadvantage in itself—of course, if the number of these duplicates is not very large. Moreover, this check is sometimes regarded as more convenient than the total exclusion of duplicates at intermediate stages of generation (as the authors of GEN put it, a retrospective check may be more efficient if only because "there are more intermediate fragments than final structures"[12b]). However, it also has some evident shortcomings (for example, if we miss only one pair of duplicates at some early stage of generation, many duplicates may appear at the last stage as a result), and therefore many authors state that generation algorithms should better be designed in such a way that the retrospective check would be completely avoided.[21] Meanwhile, as far as we are aware (judging by the literature data available), none of the presently used generators ensure absolutely rigorous exclusion of duplicates at intermediate stages of generation (that is, without any retrospective check) in the presence of an arbitrary number of arbitrary core fragments.

Our software SMOG (**s**tructural **mo**lecular **g**eneration)[1] has much in common with the aforementioned generator GENM: both are based on different original modifications of Faradjev's algorithm.[30b,c] The approach to consideration of core fragments, although independently developed, also resembles the one used in GENM (the embed−generate technique): that is, core fragments are similarly incorporated into the required structures as multiatomic "construction units", with their composition and symmetry implicitly taken into account (see section 2.3 and below). However, in contrast to GENM, the rigorous algorithm of SMOG makes it possible to exclude 100% of redundant structures at the earliest possible stages of generation, using a special duplicate-detection procedure that is independent of the number, size, composition, or symmetry of the initial core fragments.

In our previous paper on SMOG,[1] we concentrated on other key aspects of this software, planning to describe and discuss the actual algorithm for detection and removal of duplicates in a separate paper. This communication is an attempt to cover this important topic. Some notions and statements from the said paper, which are necessary for understanding and illustrating the algorithm, are briefly repeated here. In other aspects, the terminology of the present communication is almost fully consistent with the one used by Molodtsov in his description of GENM,[13b,c] except for the difference between the notion of GOODLIST and our term "set of core fragments".[22]

2. FOUNDATIONS OF GENERATION ALGORITHM

2.1. Adjacency Matrices of Molecular Graphs. One of the accepted representations of a molecular graph is its adjacency matrix $\mathbf{A} = (a_{ij})$. According to definition,[32] the element a_{ij} of the adjacency matrix is equal to the multiplicity of the edge (bond) between the vertices (atoms) i and j in the molecular graph, if such an edge exists, or 0 in the opposite case. The size N of this square ($N \times N$) and symmetrical ($a_{ij} = a_{ji}$) matrix is equal to the number of vertices in the molecular graph, that is, to the number of non-hydrogen atoms in the molecular formula.[33] The numbers of hydrogens adjacent to vertices of the molecular graph are specified within SMOG in an additional array, hereafter referred to as the **H** array. The degree of each vertex i (the sum of the multiplicities of all edges incident to it, $D_i = \sum_{j=1}^{N}(a_{ij})$) plus the number of hydrogen atoms at this vertex (the corresponding element of the **H** array, H_i) is equal to the valence of the corresponding atom: $D_i + H_i = V_i$.

Since N vertices of the graph may be numbered in $N!$ ways (among these $N!$ numberings, $N!/|\mathrm{Aut}(\mathbf{G})|$ are nonequivalent, where $\mathrm{Aut}(\mathbf{G})$ is the automorphism group of graph $\mathbf{G}$), the family $\{\mathbf{G}_i\}$ of isomorphic N-vertex graphs is described by $N!$ adjacency matrices $\mathbf{A}_i$ ($N!/|\mathrm{Aut}(\mathbf{G})|$ of them are different), which are interconvertible by permutations of their lines/columns: graphs $\mathbf{G}_i$ and $\mathbf{G}_j$ are isomorphic if and only if the symmetrical permutation group S_N contains such a permutation g that $\mathbf{A}_j = g\mathbf{A}_i g^{-1}$.

For "colored" graphs (graphs where vertices are classified by *types*), this definition is also valid. However, if we state that the permutation g must preserve the types of vertices that correspond to the given lines/columns of the adjacency matrix, a subgroup of S_N should be considered instead of the whole group. In this paper, vertices (atoms) of a molecular graph are said to belong to one and the same type if and only if the corresponding atoms have identical characteristics (element symbol, valence, charge, isotopic number). Accordingly, the lines/columns of the adjacency matrix may be ordered so that the first n_1 lines correspond to vertices of type k_1; ...; and the last n_T lines correspond to type k_T, where T is the number of different vertex types. Permutations $g \in S_N$ that preserve this distribution by types form a group: $S(T) = \oplus_t S(k_t)$, where $S(k_t)$ is the symmetrical group of permutations among vertices of type k_t. Such permutations and adjacency matrices produced by them are called *admissible*.[13b] As an example, see Figure 3a,b: the two graphs are isomorphic (differ only by the order of vertex numbering), and the first adjacency matrix may be converted

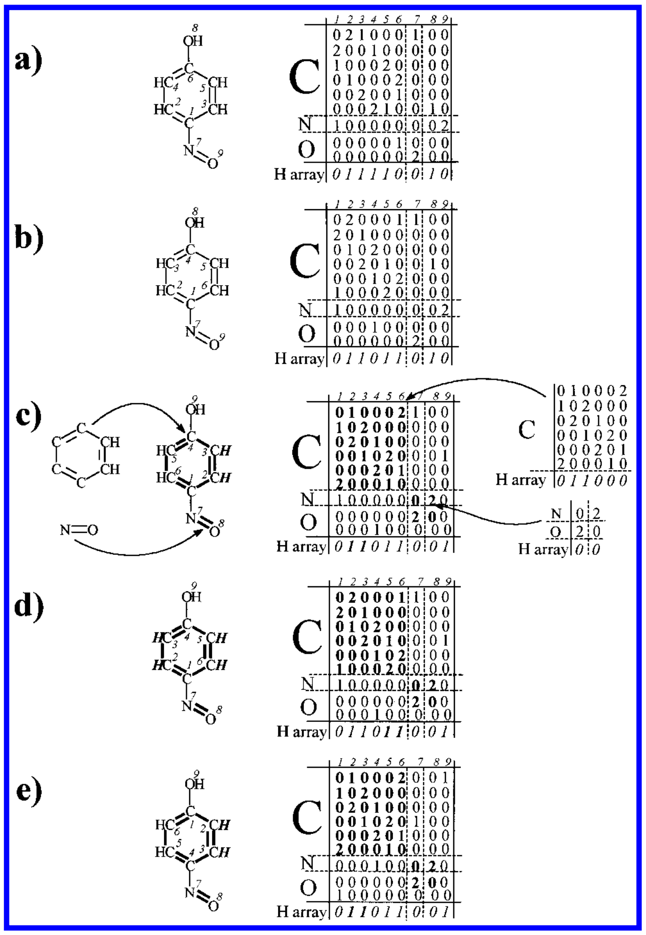


Figure 3. Admissible adjacency matrices of molecular graphs: (a, b) ordinary and (c−e) partly invariant; (a, c, d) canonical and (b, e) noncanonical.

into the second one by the following admissible permutation g:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 6 & 3 & 5 & 4 & 7 & 8 & 9 \end{pmatrix}$$

Apparently, all other matrices in Figure 3 are also admissible: the first six lines/columns belong to the C type; the seventh line/column, to the N type; the rest, to the O type (as is denoted by dashed lines and atom symbols in Figure 3). Hereafter, only admissible matrices and admissible permutations are considered.

Now we imagine a graph that was generated with the use of some nonoverlapping core fragments. As an example, see Figure 3c, where the core fragments directly incorporated in the molecule are marked by thick lines in the representation of the molecular graph, whereas the remaining part of the molecule—the one actually constructed during generation—is represented by lines of ordinary thickness. Vertices of this molecular graph, as well as of any other graph, may be numbered in different ways; therefore, the molecule may be described by several adjacency matrices. Considering one of these matrices (Figure 3c), we somehow mark (for example, by boldface) the elements that were incorporated from the core fragments: both nonzero elements, which correspond to thick-lined bonds in the graph, and zero elements, which denote the absence of bonds between some vertices. (If the core fragments contain any hydrogen atoms, we also mark the corresponding elements of the **H** array.)

Hereafter, such elements of the adjacency matrix that were incorporated from core fragments are referred to as *fixed*, or *invariant*. A matrix containing such elements is called *partly invariant*, whereas a matrix that was constructed without the use of core fragments and contains no invariant elements is an *ordinary* adjacency matrix.

If we renumber vertices of the molecular graph in a different order—that is, permute the lines/columns of its adjacency matrix (cf. Figure 3c−e)—the values of the invariant elements may either change (for example, a_{12} is equal to 1 in Figure 3c and to 2 in Figure 3d) or remain the same (cf. Figure 3c,e). The permutations $g \in S_N$ that preserve the fixed elements of the matrix are hereafter referred to as *core-invariant*. That is, g is a core-invariant permutation if and only if $a_{g(i),g(j)} = a_{ij}$ for any fixed element a_{ij} of the matrix and $H_{g(i)} \geq H_i$ for any fixed element H_i of the **H** array.

As is easily shown, core-invariant permutations form a group: $S(C)$. The same refers to admissible core-invariant permutations: $S(T,C) = S(T) \cap S(C)$ is also a group. In the absence of invariant parts, $S(T,C)$ coincides with $S(T)$.

2.2. Canonicity Predicate for Adjacency Matrices. As was stated above, a generator should produce strictly one graph (one adjacency matrix) out of each family of isomorphic graphs (their matrices). Actually, a certain predicate should be formulated for the target graphs so that exactly one graph out of each family of isomorphic graphs would satisfy this predicate and be included in the set of results, whereas all others would be rejected. Such a predicate is called the *canonicity predicate*,[30a,b] and a graph that satisfies it is called *canonical*, just as its adjacency matrix.

SMOG uses the well-known predicate of the "maximum matrix"[13,24,30b,34] with respect to admissible partly invariant matrices: a canonical matrix is the one that is **maximum** among all admissible adjacency matrices that represent isomorphic graphs and have the given invariant parts: $\mathbf{A} \geq g\mathbf{A}g^{-1}$ for all $g \in S(T,C)$. In the absence of invariant parts, the canonical ordinary matrix is simply the maximum admissible matrix. Matrices are compared as their concatenated numerical vectors: $\mathbf{A} > \mathbf{A}'$ if and only if $(a_{11}a_{12}...a_{1N}a_{21}...a_{2N}...a_{NN})$ is lexicographically greater than $(a'_{11}a'_{12}...a'_{1N}a'_{21}...a'_{2N}...a'_{NN})$. The **H** arrays are not compared: evidently, if the admissible adjacency matrices are identical for two isomorphic structures, their **H** arrays are also identical.

According to this definition, the ordinary matrix in Figure 3a is canonical and the one in Figure 3b is not: the latter is smaller than the former and may be converted into it by an admissible permutation of lines/columns, as was noted in the previous section. As to partly invariant matrices, the one in Figure 3c is canonical (although it is apparently smaller than the matrix in Figure 3a), because it cannot be transformed into a greater matrix by an admissible **core-invariant** permutation, as may be proved by successive consideration of all such permutations. At the same time, the matrix in Figure 3e is not canonical, because an admissible core-invariant permutation transforms it into a greater matrix—the one in Figure 3c.

2.3. Preliminary Steps before the Beginning of Generation. A simplified flow chart of the generation process within the SMOG program is shown in Figure 4; for more details, see Figure 6 in ref 1.

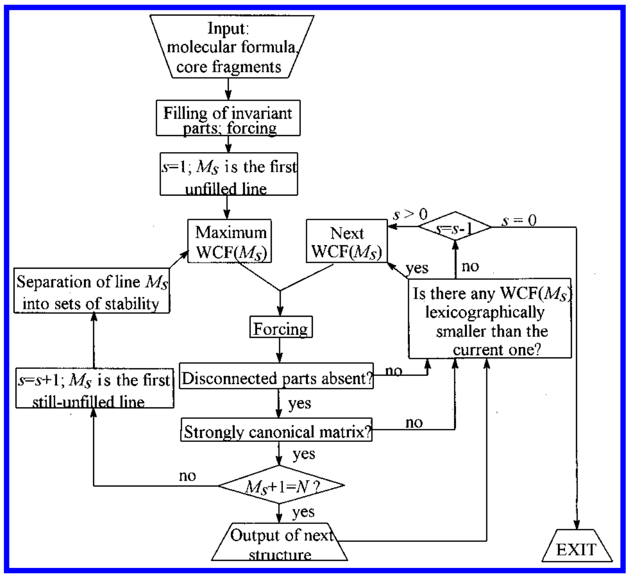

Input: molecular formula, core fragments
Filling of invariant parts; forcing
s=1; M_S is the first unfilled line
Maximum WCF(M_S)
Next WCF(M_S)
$s > 0$
s=s-1
s = 0
Separation of line M_S into sets of stability
yes
no
Is there any WCF(M_S) lexicographically smaller than the current one?
Forcing
s=s+1; M_S is the first still-unfilled line
Disconnected parts absent?
no
yes
Strongly canonical matrix?
no
yes
no
M_S+1=N ?
yes
Output of next structure
EXIT

Figure 4. Simplified flow chart of the SMOG program: s is the number of the current generation step, M_s is the line to be filled at step s, N is the size of adjacency matrix, and WCF(M_s) is a weakly canonical filling of line M_s (see section 2.5.2.1).

As an example, we consider construction of isomers that share the molecular formula $C_4H_3N_3O_2$ and contain a core fragment.

The number of non-hydrogen atoms in this molecular formula is $N = 9$. We consider an "empty" $N \times N$ matrix, which will be used as the basis for producing adjacency matrices of the desired isomers. Initially, its elements are regarded as unknown, or *unfilled*, as is denoted by the special symbol δ;[30b] only the diagonal elements are zero by definition (Figure 5a). Following refs 13b and 30b, we denote a matrix that contains any unfilled elements as $\mathbf{B} = (b_{ij})$, in contrast to a completely filled matrix $\mathbf{A} = (a_{ij})$.

Lines/columns of the **B** matrix—that is, vertices of target graphs—are ordered according to atom types: C (1−4), N (5−7), and O (8−9). For each vertex i, the maximum possible degree is equal to the valence of the corresponding atom: $D_{1-4} \leq 4$, $D_{5-7} \leq 3$, and $D_{8-9} \leq 2$.

Now we consider the core fragment—the pyridazine ring in our example (Figure 5b).

Before the generation process, SMOG directly *embeds* the core fragment(s) into the unfilled adjacency matrix **B**. This embedding is accomplished in the following way. Each vertex of each core fragment is mapped into some vertex of the target molecular graphs—that is, put into correspondence with some line/column of **B**. If the set of core fragments contains more than one copy of a certain fragment, these copies are regarded as different fragments and embedded separately. The mapping should be *nonoverlapping* (that is, different vertices of one and the same fragment or vertices that belong to different core fragments should be mapped into different vertices of the target structures) and *consistent* (each vertex should be mapped into a vertex with the same atom type); in all other respects, it is arbitrary. After some way of mapping is selected, the further embedding procedure is obvious: if vertices i_0 and j_0 of one and the same core fragment were put into correspondence with the lines/columns i and j of the **B** matrix, respectively, and the multiplicity of the (i_0,j_0) edge in the fragment is equal to K, then we set $b_{ij} = b_{ji} = K$ in **B**. Evidently, if there is no bond between i_0 and j_0 in the core fragment, then $b_{ij} = b_{ji} = 0$ (Figure 5b, left-hand matrix). If vertices i_0 and j_0 belong to different core fragments, the b_{ij} value remains unknown, as is denoted by the δ symbol.

SMOG conducts generation in such a way that the initial mapping, once chosen for the vertices of core fragments, is preserved in all target structures and, accordingly, in all partially generated structures that appear at intermediate stages of the generation process. Thus, the embedded (fixed) elements of the adjacency matrix indeed form its invariant part.

Statement. For any nonoverlapping and consistent mapping and embedding of core fragments on **B**, generation of all canonical admissible core-invariant matrices represents exhaustive and irredundant generation of isomers containing these nonoverlapping core fragments.

Proof. First, we prove that generation will be *exhaustive*—that is, *all* different isomers that share the given molecular formula and contain the given nonoverlapping core fragments will be found. Indeed, a molecule will be generated if and only if its atoms (vertices) may be numbered in such a way that an admissible matrix with the given invariant parts would result (if several ways of such numbering are possible, the canonical one will be selected). So, we have to show that at least one numbering of this kind may be actually found for any molecule containing the desired set of nonoverlapping core fragments. To find this numbering, we should first find the required set of nonoverlapping core fragments (subgraphs) in the molecular graph. (The number of such sets may be more than one—for example, if some core fragments are identical or if additional copies of some fragments were assembled during generation.) Then, we should associate each of these subgraphs with the corresponding set of invariant elements in the matrix (this is possible because of the condition of nonoverlapping mapping) and number the vertices in this subgraph in accordance with the numbers of matrix lines/columns that correspond to these invariant elements (there may be more than one possible way of numbering, depending on the symmetry of the fragment). Finally, we should number the remaining vertices (if any) in such a way that the resultant adjacency matrix would be admissible; the possibility of such numbering is ensured by the condition of consistent mapping.

On the other hand, *irredundancy* of generation is achieved because of the canonicity requirement: among all different core-invariant admissible matrices that represent isomorphic graphs, we can unambiguously select the maximum one. Irredundancy is preserved even in cases where an additional copy (or copies) of some core fragment(s) has been assembled from the "remaining" atoms and/or from other core fragments during generation: indeed, as was mentioned in the previous paragraph, such a situation simply means that we may find several sets of nonoverlapping subgraphs corresponding to the given invariant parts of the matrix. Considering all ways of numbering for all these sets, we may select the one that results in the largest (canonical) core-invariant admissible matrix. The proof is completed.

After the embedding, SMOG uses the procedure of *forcing*—automatic unambiguous filling of some of the yet-unfilled elements in the adjacency matrix.[1,13b,30b] For example, the unfilled elements of the matrix lines/columns 5−6 may be filled with zeros (Figure 5b, right-hand matrix),

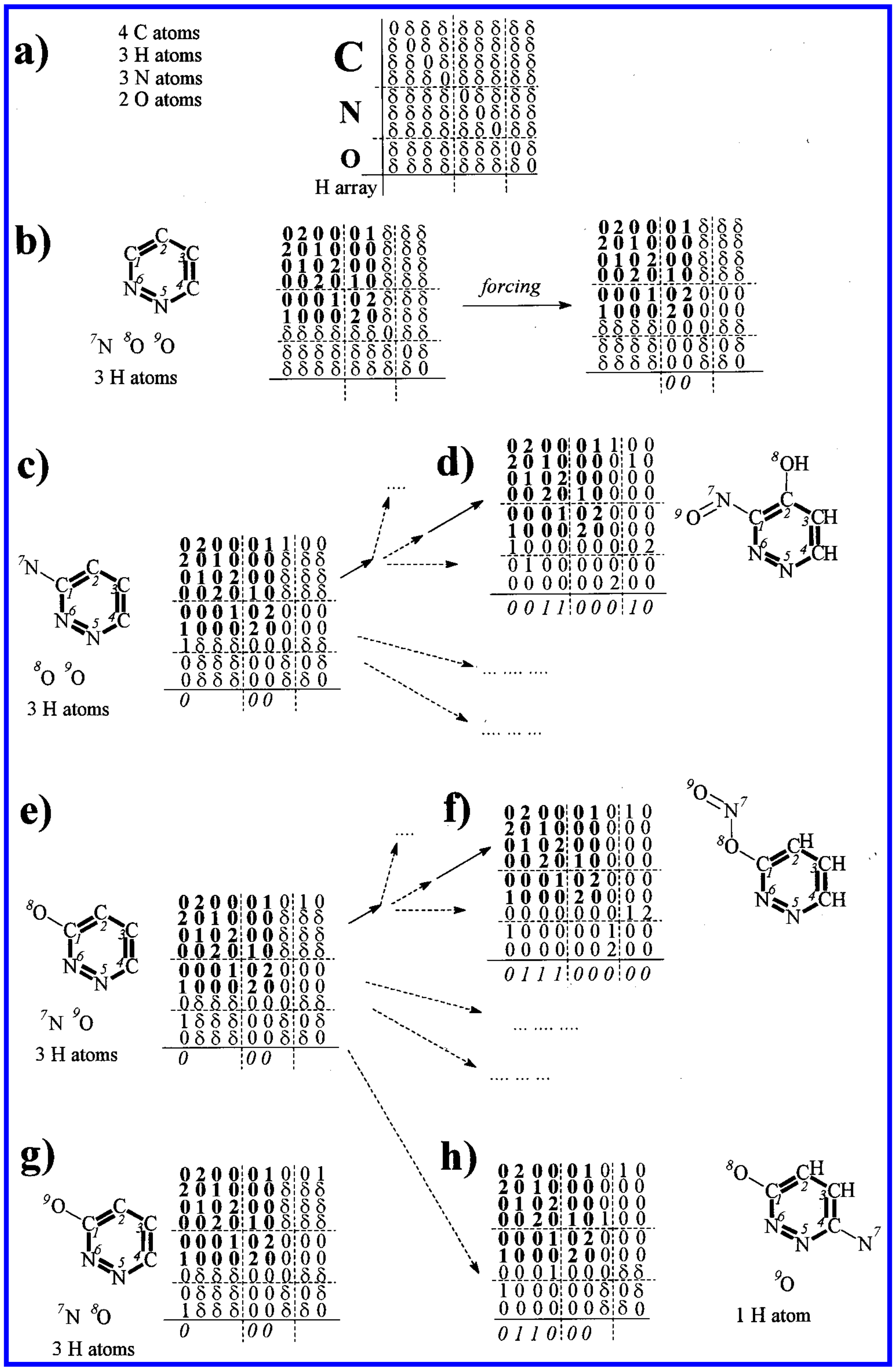


Figure 5. Partly invariant adjacency matrices at different stages of filling: (a) initial matrix, (b) after setting of invariant parts, (c) partially filled, strongly canonical, (d) one of its full complements, canonical, (e) partially filled in another way, strongly canonical, (f) one of its full complements, canonical, and (g, h) partially filled, noncanonical.

because the corresponding vertices have no free valences left—indeed, nitrogen is regarded as trivalent in this problem, and the sum of elements in the corresponding lines (or columns) 5 and 6 is already equal to 3.

2.4. Extension of the Canonicity Predicate. Each generation step consists of *filling* of a new line/column of the adjacency matrix, that is, assignment of some definite nonnegative values to its yet-unfilled elements. Lines are filled in the order of their identity numbers, starting from the top. The corresponding columns are filled automatically after the filling of each line: $b_{ji} = b_{ij}$ ($j > i$); thus, if the size of the adjacency matrix is N and the number of the line to be filled at the current step is M, we have to fill only elements from $b_{M,M+1}$ to b_{MN} inclusive at this step.

Apparently, each partially filled matrix at intermediate stages of generation corresponds to some partially assembled intermediate structure. Evidently, to avoid the appearance of duplicate structures at the final output of the program, duplicate intermediate structures should be found and excluded at each stage of generation.

We introduce some terms. A partially filled matrix $\mathbf{B}^+$ that is produced by any filling of some yet-unfilled elements in a partially filled matrix **B** (and coincides with **B** on the filled elements of **B**) is called a *complement* of **B**;[13b,30b] by analogy, considering a completely filled matrix **A** produced by some filling of all unfilled elements in **B**, we refer to it as a *full complement* of **B**.

According to principles of depth-first search, after each successive line/column of the adjacency matrix is filled (*i.e.*, a new partially assembled structure is generated), the program should in some way determine if the current matrix **B** may be complemented by some filling of its yet-unfilled elements so that a canonical adjacency matrix **A** (a full complement of **B**) would result. If the **B** matrix *a priori* cannot be filled in this way, the program must consider some way of modifying its filled elements: choose another way of filling for its last filled line M or (if all different ways of filling have already been considered for M) for the lines filled earlier (Figure 4). On the contrary, if a canonical full complement is possible for the current partially filled matrix, the program passes to the next stage—filling of its next unfilled line. As may be shown,[30a,b] this approach indeed ensures that exactly one (canonical) structure out of each family of isomorphic molecular graphs is found and, moreover, noncanonical structures are excluded at the earliest possible stages.

The only remaining problem is to determine which partially filled matrices may or may not have a canonical full complement. In other words, we should **extend** the canonicity predicate to partially filled adjacency matrices.

We consider the sets I and J of the lines/columns in the adjacency matrix that are already filled and still unfilled, respectively, at some generation stage. For each vertex type k_t, we may also consider the sets of filled and unfilled lines/columns: I_t and J_t. The group $S(I,J) = \oplus_t(S(I_t) + S(J_t))$ includes all admissible permutations that independently permute filled and unfilled lines/columns within each type: any permutation g from it may be represented as $g_{IJ} = g_I \bigcirc g_J$, where $g_I \in S(I) = \oplus_t(S(I_t))$ and $g_J \in S(J) = \oplus_t(S(J_t))$. The $S(I,J,C)$ group is the subgroup of $S(I,J)$ that includes all core-invariant permutations from it. Now we may introduce a definition: a partially filled matrix is called *strongly canonical* if no permutation $g \in S(I,J,C)$ increases this matrix—that is, $\mathbf{B}' = g\mathbf{B}g^{-1} \leq \mathbf{B}$ for all g from this group.[13c,30c] Note that partially filled matrices, like completely filled ones, are compared as their numerical vectors; however, if some element b_{ij} is unfilled (δ) in any of the numerical vectors (or both of them) and all the preceding elements in both vectors are filled and equal ($b_{kl} = b'_{kl}$ for $k < i$ or for $k = i$ and $l < j$), then the relations "greater than" and "smaller than" are undefined for such matrices.

Evidently, the requirement of strong canonicity is an extension of the canonicity predicate:[13b,30b,c] if a partially filled matrix is not strongly canonical, none of its full complements may be canonical. Indeed, suppose that some admissible core-invariant permutation of filled lines/columns increases a partially filled matrix: $g\mathbf{B}g^{-1} > \mathbf{B}$. Then, if **A** is a full complement of **B**, one can show that $g\mathbf{A}g^{-1} > \mathbf{A}$ by direct consideration of the corresponding numerical vectors. The same is true for permutations of unfilled lines/columns.

2.5. Overview and Example of Stepwise Generation Process. After mapping of core fragments in the adjacency matrix and forcing (Figure 5b), as was stated above, the remaining elements must be filled line after line: at each step, the uppermost unfilled line of the matrix is filled, and then the corresponding column is filled according to symmetry. For each line/column of the matrix, the sum of its elements (corresponding to the degree of the vertex) must be less than or equal to the valence of the corresponding atom: the difference between the valence and the vertex degree corresponds to the number of hydrogens attached to the current atom, and the sum of these differences (the *remaining valences*) over all vertices must be equal to the total number of hydrogens in the molecular formula (3 in our case).

For example, the topmost unfilled line of the right-hand matrix in Figure 5b is simply its first line. One possible way of filling it is shown in Figure 5c: $b_{17} = 1$; $b_{18} = b_{19} = 0$. In this case, the sum of elements in the first line is 4; since the valence of carbon is also 4, the condition that the sum of elements in a line must be less than or equal to the valence of the corresponding atom is satisfied. The remaining valence H_1 is zero. The first column is filled according to symmetry.

After forcing (if it is possible, which is not the case here) and connectivity testing[35] (see Figure 4), we proceed to the determination of strong canonicity for the partially filled matrix obtained at the current step. Direct search within the $S(I,J,C)$ group shows that the adjacency matrix in Figure 5c is strongly canonical (see next sections for the implementation of the search procedure): no permutation from this group produces a greater matrix. That is, we may pass to the next step, which consists of filling the next unfilled line/column (the second one in our case), and so on. If all lines are filled so that strongly canonical matrices are obtained at each step, and at the same time we take care that the resultant matrix should correspond to a connected structure and the total number of remaining valences should be equal to the total number of hydrogens in the molecular formula, we arrive at a ready canonical structure (for example, Figure 5d).

After all lines are filled, we return back to lines filled at previous steps (at the last step, then at the last but one, *etc.*) and try to fill them in some other way (Figure 4). For example, a strongly canonical matrix obtained by another filling of the first line is shown in Figure 5e, and one of its full canonical complements generated by subsequent filling is given in Figure 5f.

However, we also often arrive at noncanonical partially filled matrices during generation. For example, it is easily seen that the matrix in Figure 5g is increased by a permutation from $S(I,J,C)$:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 9 & 8 \end{pmatrix}$$

and the matrix in Figure 5h, by another permutation from the same group:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 4 & 3 & 2 & 1 & 6 & 5 & 7 & 8 & 9 \end{pmatrix}$$

As was stated above, whenever we arrive at such noncanonical matrices during generation, we should make a step back (Figure 4), because they cannot give rise to any canonical full complement. That is, we should find some convenient way of determining if the current partially filled matrix is strongly canonical.

2.5.1. Test for Strong Canonicity. The test for strong canonicity, applied to the partially filled adjacency matrix **B** after filling of each successive line, is performed in a rather straightforward way, which may be schematically described as follows.

(1) Within the $S(I) = \oplus_t(S(I_t))$ group, we select some permutation g_I.

(2) A new matrix is obtained: $\mathbf{B}^* = g_I\mathbf{B}g_I^{-1}$.

(3) If $\mathbf{B}^* > \mathbf{B}$ and g_I is core-invariant, **B** is not strongly canonical. The test is completed.

(4) We try to choose a permutation g_J from the $S(J) = \oplus_t(S(J_t))$ group in such a way that $g_{IJ} = g_I \bigcirc g_J$ would produce (for the given g_I) a matrix larger than **B** and preserve the invariant parts of the matrix. (As was noted above, any desired permutation $g \in S(I,J,C)$ may be represented in the form $g_{IJ} = g_I \bigcirc g_J$.) If there is no such permutation g_J that $g_{IJ} = g_I \bigcirc g_J$ would be core-invariant and at the same time satisfy the condition $g_{IJ}\mathbf{B}g_{IJ}^{-1} = g_J\mathbf{B}^*g_J^{-1} = \mathbf{B}^{**} > \mathbf{B}$, we select some new g_I from $S(I)$ and return to step 2 again. In the other case, **B** is noncanonical, and the test is completed.

Steps 2−4 are repeated until all possible core-invariant permutations in $S(I,J)$—that is, all permutations in $S(I,J,C)$—are considered or until we encounter a core-invariant permutation that produces a matrix larger than **B**.

2.5.2. Simplification of the Search among Permutations. Since exhaustive search within the $S(I,J,C)$ group is usually computationally intensive and should be performed at each step of generation, a number of ways to reduce it were implemented in SMOG.

(1) We may establish some comparatively simple criterion, such that adjacency matrices that do not satisfy this criterion may in no case be strongly canonical. Therefore, no additional search procedure would be required for such matrices.

(2) After the invariant parts of the matrix are specified but before any other elements are filled (Figure 5b), we may find some classes of admissible permutations that will not be core-invariant regardless of the filling of all other elements. Therefore, these permutations should be omitted during the search for core-invariant permutations in the $S(I,J)$ group at any stage.

(3) We may try to *construct* an admissible core-invariant permutation that increases the matrix, rather than just find it in the set of possible permutations. That is, ordinary search for the target permutation is substituted by its *stepwise formation*.

The corresponding procedures are outlined in the next three subsections.

2.5.2.1. Separation of Lines into Sets of Stability. We consider filling of the Mth line in the partially filled matrix $\mathbf{B}(N,N)$ during generation. According to our algorithm, filling of the Mth line implies that all lines/columns up to $M - 1$ inclusive have already been filled at previous stages. Since elements of the Mth line from b_{M1} to $b_{M,M-1}$ were automatically filled during filling of previous lines ($b_{M1} = b_{1M}$, ..., $b_{M,M-1} = b_{M-1,M}$), and $b_{MM} = 0$ by definition, we have to fill only the elements from $b_{M,M+1}$ to b_{MN} inclusive (section 2.4) that were not fixed from the beginning of generation or filled at some previous step as a result of forcing. We consider two elements that are to be filled at the current step, b_{Mi} and b_{Mj} ($M < i < j$), and analyze the possibility of their filling in such a way that $b_{Mi} < b_{Mj}$.

Statement. A partially filled matrix **B** is not strongly canonical if the following conditions are simultaneously satisfied for vertices i and j ($M < i < j$) and the corresponding elements b_{Mi} and b_{Mj} ($b_{Mi} < b_{Mj}$) that have been filled at the current step of generation: (1) vertices i and j are of the same type; (2) either $M = 1$ or $b_{ki} = b_{kj}$ for all $k < M$; (3) the swapping of i and j—that is, the permutation in which the line/column i is substituted by j and vice versa but all other lines/columns of the matrix are left intact—is core-invariant.

The **proof** of this statement is evident: it is sufficient to consider the action of the permutation g_J described in condition 3:

$$\begin{pmatrix} 1...i...j...N \\ 1...j...i...N \end{pmatrix}$$

According to condition 1, this permutation is admissible; according to condition 2, it does not change any of the elements in the lines above the currently filled one. Therefore, the initial matrix and the matrix produced by the permutation have numerical vectors that are identical up to the $(M, i - 1)$th element inclusive but have different (M, i)th elements: equal to b_{Mi} in the former matrix and b_{Mj} in the latter one. Since $b_{Mi} < b_{Mj}$, we obtain $\mathbf{B} < g_J\mathbf{B}g_J^{-1}$. That is, **B** is noncanonical. The proof is completed.

As an example, we consider the matrix in Figure 5g. Evidently, all conditions of the above statement are fulfilled for $M = 1$, $i = 8$, and $j = 9$. And, indeed, swapping of the lines/columns 8 and 9 produces a greater matrix—that is, the matrix in Figure 5g is noncanonical.

That is, all unfilled elements of line M may be divided into equivalence classes (referred to as *sets of stability*[36]) in such a way that conditions 1−3 of the above statement are true for any pair of elements b_{Mi} and b_{Mj} that belong to one and the same class. Of course, the sets of stability are different for different lines and depend on the earlier filled lines. For admissible matrices in the absence of core fragments, each set is represented by a single numerical range (i and j belong to the same set of stability if $i_0 \leqslant i < j \leqslant j_0$, where i_0 and j_0 are the boundaries of the range);[13b] for partly invariant matrices, a set of stability may consist of several disjoint ranges as well.[1,37]

If line M is filled so that the general rule $(i < j) \Rightarrow (b_{Mi} \geq b_{Mj})$ is true within each set of stability in this line, such filling is referred to as *weakly canonical*.[30b] Evidently, a partially filled matrix may be strongly canonical only if each successive line was filled in the weakly canonical manner. Therefore, the procedure of filling in the generation algorithm is implemented so that only weakly canonical fillings are considered for a line/column at each step (Figure 4): other fillings, such as the one in Figure 5g, are automatically omitted.

However, weakly canonical filling of all lines is a necessary but insufficient condition for complete exclusion of noncanonical matrices:[30b,c] as an example, see Figure 5h. A special test for strong canonicity is still necessary.

Separation into sets of stability is also useful for reducing the search in the $S(I,J,C)$ group (see section 2.5.1): permutations g_J that lead to violation of the weak canonicity rule in some line need not be considered during this search at all.[38]

2.5.2.2. *A Priori* Exclusion of Some Permutations from Consideration. Some classes of admissible permutations of lines/columns may be excluded from consideration even before the beginning of generation, because all permutations from these classes *a priori* may not be core-invariant.

For example, we refer to the graph in Figure 5c again and consider some permutation g in which the image of vertex 3 is vertex 4: that is, the third line/column in the adjacency matrix **B** becomes the fourth one in $\mathbf{B}' = g\mathbf{B}g^{-1}$:

$$\begin{pmatrix} 1 & \ldots & 3 & \ldots & 9 \\ g(1) & \ldots & 4 & \ldots & g(9) \end{pmatrix}$$

It is easily shown that such a permutation may not be core-invariant, regardless of the images of other vertices and regardless of the filling of unfilled elements. We prove this statement *ex adverso*, assuming that g is core-invariant. Since $g(3) = 4$, vertex 4 in the resultant graph **G′** (after the permutation) must form all edges that were incident to vertex 3 in the initial graph **G** (before the permutation). That is, vertex 4 in **G′** must be incident to one double C═C edge and one single C−C edge (set 1). However, since the permutation is core-invariant, vertex 4 in **G′** also must form all kinds of edges that were incident to the vertex with the same number 4 in **G** and correspond to fixed elements of the adjacency matrix—otherwise, the invariant values will not be preserved. That is, vertex 4 in **G′** must be incident to one double C═C edge and one single C−N edge (set 2). Combining set 1 and set 2, we obtain an environment that contains at least one C═C edge, at least one C−C edge, and at least one C−N edge.

However, no such environment may actually exist for vertex 4 in **G′**, as is easily seen if we consider not only edges of the graph but also "nonedges"—that is, zero elements of the adjacency matrix. Indeed, as is apparent from the matrix in Figure 5b, vertex 4 in **G′** may form strictly one carbon−carbon edge, and this edge is double: all other elements of the adjacency matrix that might correspond to carbon−carbon edges of vertex 4 were set to zero during the mapping of the core fragment. (All four carbons that are present in the molecular formula are also present in the core fragment, and, therefore, the number and multiplicity of carbon−carbon edges is known in advance for each vertex.) However, if vertex 4 in **G′** is the image of vertex 3 produced by a core-invariant permutation, then, as was stated in the previous paragraph, it must also form one single C−C edge. We have arrived at an inconsistency. Therefore, vertex 4 may not be the image of vertex 3 in a core-invariant permutation; in other words, vertex 3 is regarded as *nonrepresentable* by vertex 4.

Applying the same reasoning to the general case, we may formulate a criterion of nonrepresentability for vertices. We consider two vertices of a molecular graph: i and j. By $F_{t,m,i}$, we denote the number of matrix elements b_{ik} (k is a line/column corresponding to a vertex of type t) that are fixed and equal to m. The $F_{t,m,j}$ value is defined in a similar way, as the number of fixed elements $b_{jk} = m$. By $W_{t,m,j}$, we denote the number of *vacancies* for the type t, multiplicity m, and vertex j—that is, the number of unfilled elements $b_{jk} = \delta$ (where k is of the type t) that may be assigned the value m during generation. Then, if the inequality

$$F_{t,m,i} > F_{t,m,j} + W_{t,m,j} \qquad (1)$$

is true for any vertex type t ($1 \le t \le T$, where T is the number of vertex types) and any multiplicity m ($0 \le m \le 3$), vertex i is nonrepresentable by vertex j.

As an example, we may consider Figure 5b again: $F_{C,1,3} = 1$ (a single carbon−carbon edge); $F_{C,1,4} = 0$ and $W_{C,1,4} = 0$. Inequality 1 is fulfilled.

Another class of permutations that may not be core-invariant is encountered when the union of set 1 (the environment of vertex i) and set 2 (the environment of vertex $j = g(i)$) contains too many edges—that is, the valence of the corresponding atom is exceeded. Using the above notation and assuming that both i and j belong to the vertex type with the valence V, we may represent this case of nonrepresentability as

$$\sum_{m=1}^{3}[\mathrm{m}(\sum_{t=1}^{T} \max(F_{t,m,i}, F_{t,m,j}))] > V \qquad (2)$$

Since this criterion, in contrast to (1), is symmetrical with respect to i and j, its fulfillment means that each of the two vertices is nonrepresentable by the other one. Note, however, that (2) does not work for vertices 3 and 4 in Figure 5b, because the valence of carbon is 4, and

$$\underset{\mathrm{C-C}}{1} + \underset{\mathrm{C-N}}{1} + \underset{\mathrm{C=C}}{2} = 4$$

Thus, as soon as all vertices of core fragments are mapped on the adjacency matrices of target graphs, we may determine nonrepresentable vertices according to the above-described (and maybe also some other) criteria and store the results in the matrix $\mathbf{NR} = (NR_{ij})$, where NR_{ij} is equal to -1 if i is nonrepresentable by j and 0 in the opposite case. Now, while considering permutations from the $S(I,J)$ group during the test for strong canonicity at any stage of generation (section 2.5.1), we may automatically omit all permutations g_I or g_J for which there exists such i that $NR_{i,g(i)} = -1$. Thus, the search is reduced, and the efficiency of generation significantly increases.

The procedure of determining vertices that are nonrepresentable by each other resembles (to some extent) determination of orbits in the graph of the core fragment. Indeed, these two procedures yield the same classifications for the case in Figure 5, but this is not true in the general case. For example, consider generation for the formula C_6H_{12} with the core fragment $C^1{-}C^2{-}C^3$ (for simplicity, the vertices of this core fragment are numbered). Vertices C^1 and C^2 belong to different orbits of this fragment. However, core-invariant permutations exchanging these two vertices are possible—for example, just consider any molecular graph that contains a C−C−C−C chain formed by this fragment with an additional C atom attached to it.

2.5.2.3. Stepwise Construction of Core-Invariant Permutations. In spite of the simplifying procedures described above, exhaustive search for matrix-increasing core-invariant permutations in the $S(I,J)$ group is often very inefficient, all the more so because core-invariant permutations often form a minor part of this group. So, as was noted above, a simple search should be substituted by directed construction of the target permutation.

For example, we consider the partially filled matrix **B** in Figure 5h and try to find an admissible core-invariant permutation that would increase this matrix.

As is easily noticed, swapping of the filled lines/columns 1 and 4 produces a matrix where the very first element "outside" the fixed part is greater than the corresponding element in **B** (because $b_{g(1)g(7)} = b_{47} = 1 > b_{17} = 0$). However, this permutation is not core-invariant, because it changes some of the fixed elements: for example, $2 = b_{12} \neq b_{g(1),g(2)} = b_{42} = 0$. We try to find another permutation in which the image $g(1) = 4$ would be preserved but other lines/columns would be rearranged so that the resultant permutation would be core-invariant. In other words, the target permutation g should possess the following properties:

(1) It should preserve the matrix-increasing substitution of b_{17} by b_{47}—that is, $g(1) = 4$.

(2) It should be core-invariant—*i.e.*, it should permute other lines/columns of the matrix in such a way that the resultant matrix **B**** would coincide with **B** on their fixed elements.

(3) Elements that precede b_{17} in the numerical vector should be equal for **B**** and **B** ($b_{11} = b_{11}^{**}$, ..., $b_{16} = b_{16}^{**}$)—otherwise, **B**** will not necessarily be greater than **B**. (In our example, however, this requirement is qutomatically satisfied if condition 2 is fulfilled, because elements from b_{11} to b_{16} inclusive belong to the invariant part of the matrix.)

If we manage to find such a permutation, the **B** matrix is noncanonical; in the other case, no permutation with $g(1) = 4$ may be core-invariant and matrix-increasing, and we should proceed to consideration of other permutations.

We describe the process of constructing the desired permutation step by step. Firstly, as was noted above, swapping of 1 and 4 is not core-invariant, because $b_{12} \neq b_{g(1),g(2)} = b_{42}$. So, we find the image of vertex 2, $g(2)$, for which $b_{g(1),g(2)}$ would be equal to $b_{12} = 2$. The only solution that satisfies this requirement is $g(2) = 3$ (indeed, $b_{34} = 2$). Then, since $b_{g(2),g(3)}$ must be equal to $b_{23} = 1$ (another fixed element), we obtain $g(3) = 2$. But the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 4 & 3 & 2 & 1 & 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$

is not yet core-invariant, because $b_{16} = 1$ and $b_{g(1),g(6)} = b_{46} = 0 \neq 1$. Just as before, we find $g(5) = 6$ and $g(6) = 5$. As may be proved by direct checking, now g is indeed core-invariant, and $b_{17}^{**} = b_{47} > b_{17}$. That is, **B**** > **B**, and the initial matrix is noncanonical.

In the general case, such stepwise construction of the necessary permutation is implemented as a backtracking procedure. That is, knowing $g(1) = x_1$, ..., $g(k) = x_k$, we must find such $g(k + 1)$, ..., $g(N)$ that the resultant permutation g would change neither the fixed elements of the matrix (see condition 2 in the above example) nor any of its elements up to some element b_{CD} (condition 3).

We find the possible values for $g(k + 1)$, such that $b_{g(i),g(k+1)} = b_{i,k+1}$ for all elements $b_{i,k+1}$ preceding b_{CD} in the numerical vector and for all fixed elements $b_{i,k+1}$ with $i < k + 1$. If there is no such $x_{k+1} = g(k + 1)$ value, the desired permutation may not be constructed; in the opposite case, we select one of the values thus found and choose $x_{k+2} = g(k + 2)$ in a similar manner, then x_{k+3}, ..., x_N. If we cannot find a suitable vertex x_s for some $k + 1 < s \leq N$, we return to $s - 1$ and try to choose another vertex for x_{s-1}. If, as a result, we find suitable vertices for all s from $k + 1$ to N inclusive, it means that we have arrived at the desired core-invariant permutation.

This technique of constructing permutations is apparently rigorous—in the sense that it will produce a permutation satisfying the given conditions (core-invariance and conservation of the element values up to some element b_{CD}) if such a permutation exists. However, it does not yield *all* core-invariant permutations for the given partially filled matrix, so if the desired permutation does not exist for the given $g_I \in S(I)$, $g_J \in S(J)$, and b_{CD}, consideration of some other permutations g_I and g_J still may be necessary at the same step (see section 2.5.1).

2.5.3. Some Notes. Thus, we have designed a method for complete exclusion of noncanonical partially filled adjacency matrices during generation. Unnecessary branches of the generation tree are pruned at the earliest possible stages, and, as a result, duplicate (isomorphic) structures are completely excluded from the final set of results. The method is apparently rigorous, because it uses a straightforward approach for consideration of permutations within the $S(I,J,C)$ group. All of the simplifying procedures are implemented so that they let us "omit" only those permutations which *a priori* cannot produce an admissible core-invariant partially filled matrix greater than the current one.

The approach of SMOG explicitly depends neither on the symmetry of core fragments nor on the number and structure of their free valences, with one exception. As was noted in section 1.2 of this paper (see Figure 2), representation of core fragments by imaginary atoms during generation may entail some difficulties. However, these difficulties are avoided if two conditions are satisfied for the given core fragment: (a) the automorphism group of its set of free valences is equivalent to that of a single atom (an evident example is a fragment where only one atom has free valences), and (b) no additional copies of this fragment may be assembled from the remaining atoms and/or from other core fragments during generation. Indeed, thus we avoid both situations shown in Figure 2. Core fragments that satisfy these two conditions simultaneously are called *groups* in SMOG;[1] for them, the generate−embed approach may be used without complications (and, moreover, is often preferable). For example, we consider Table 1 and Figure 6: fragments **9** and **10** are groups for the formula $C_9H_{11}NO_2$; fragment **8** is a group when used for this formula together with the core fragment **3**. Indeed, each of the fragments **8**−**10** contains only one atom with free valences, and the target structures for the molecular formula in question may contain only one copy of fragment **9** or **10** (or only one copy of fragment **8** if another oxygen-containing core fragment **3** is present)—otherwise, the number of nitrogen (or oxygen) atoms in the resultant molecules would be exceeded in comparison with the given molecular formula. Therefore, SMOG may treat such fragments as single imaginary atoms during generation.

One more detail should be noted. Evidently, only non-overlapping substructures may be used as multiatomic construction units for generation. However, in real chemical problems we often have to consider some substructures that

Table 1. Comparison of the GENM and SMOG Generators: Data for the Molecular Formula $C_9H_{11}NO_2$

no.	core fragments (in Figure 6)	no. of structures: GENM generated	no. of structures: GENM after removal of duplicates	no. of structures: SMOG	CPU time, s: GENM 386/20 MHz	CPU time, s: SMOG Pentium/90 MHz	success/failure ratio: connectivity test	success/failure ratio: strong canonicity test
1	1, 4	2 064	1 845	1 845	7.6	10.0	13.55	4.23
2	2, 4	13 584	10 264	10 264	25.0	39.5	11.70	5.62
3	3, 7, 8	6 185	3 147	3 147	13.0	6.2	15.92	3.61
4	3, 8, 9	1 891	940	940	3.5	2.0	11.08	3.25
5	3, 8, 10	36 575	17 468	17 468	51.0	38.5	24.25	4.22
6	5, 4	220	215	215	0.5	0.3	6.84	3.81
7	6, 6, 9	8 825	4 457	4 457	15.0	23.0	2.97	4.32
8	2, 8	20 316	19 896	19 896	52.0	80.0	2.95	9.85

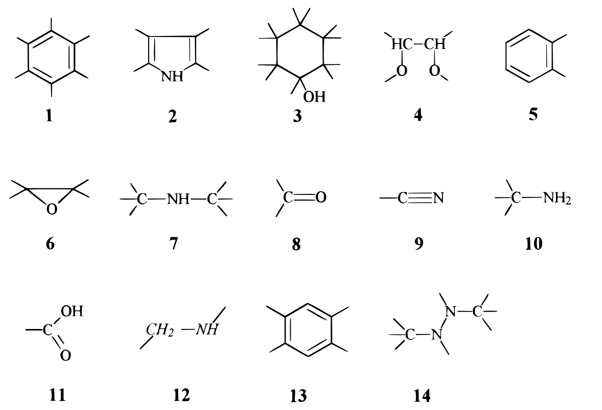


Figure 6. Core fragments used for generation of isomers by the GENM and SMOG programs. Free valences may be saturated in all possible ways, including formation of multiple bonds or (except for atoms with italicized labels) bonds with hydrogen.

are included in GOODLIST but generally may overlap in target structures:[4,7] for example, if we state that the molecules to be generated must contain the benzene ring and the pyrrole ring that may possibly overlap, then we should construct not only molecules that contain the two rings separately but also molecules that contain them in some fused form (*e.g.*, as in indole). In the case when the possibility of overlapping is explicitly specified, SMOG at first determines the parts of different GOODLIST fragments that *cannot* overlap with other GOODLIST fragments in any chemical structures (for example, the C═C and C≡C subfragments may not overlap, because the existence of their common atoms would require the presence of carbon with the valence 5 or higher). Then, the nonoverlapping parts thus determined are used as ready construction units, and other parts of the fragments from GOODLIST have to be assembled from separate atoms during generation. The presence of whole fragments in the generated or partially generated molecular graphs has to be checked by a special subgraph-searching procedure.[1,39]

3. RESULTS AND DISCUSSION

Generally speaking, demonstration of the adequacy of the above approach should have two aspects. Firstly, we should show (by several practical examples, since the theoretical proof was given above) that the program based on the proposed algorithm indeed ensures exhaustive irredundant generation. Secondly, we should compare the efficiency of the program with that of some other structure-generating software products and then make some conclusions as to the practical value of SMOG compared to other generators.

Actually, demonstrating the exhaustiveness and irredundancy is not difficult: for example, one may just consider some problems for which the number of resultant structures is already definitely known from some sources (or may be found with the use of some other graph-theoretical techniques or formulas) and compare the overall number of the generated structures with the required value. Of course, examples where other programs fail to yield the correct number of isomers would be the most convincing. However, these latter examples are very scarce, because (as was noted above) most authors of generation algorithms usually concentrate on data concerning generation from separate atoms[8,9a,14,15,26] rather than assembly with the use of arbitrary multiatomic fragments, even in cases where the possibility of such assembly is actually envisaged in the algorithm.[11,12] The exception is a paper on the aforementioned GENM program.[13c]

For the same reason, a comprehensive or at least sufficiently convincing comparison of the program efficiencies is even more problematic. Actually, some data on the performance of different isomer generators are available from literature[9,10,12−15] (judging by these results, SMOG appears to be faster than most programs used at present,[1] except for the highly efficient MOLGEN[9] and GENM[13]), but these data almost exclusively refer to generation with the use of separate atoms only. Naturally, their extrapolation to the case of assembly with the use of multiatomic fragments would be most inadequate.[40] Again, as far as we are aware, suitable data on the results and efficiency of fragment-using generation are available in only one comparatively recent publication, the one that concerns the GENM program.[13c]

Therefore, we decided to illustrate the work of our software in this paper in the following way. Firstly, we estimated the degree of irredundancy for GENM and SMOG, as well as their comparative efficiency. At the second stage of testing, we tested the irredundancy of SMOG by examples of some actual problems that had formerly been encountered in the field of computer-assisted structure elucidation.

The results listed in Table 1 (for the molecular formula $C_9H_{11}NO_2$, to which the majority of results in ref 13c refers) and Table 2 (for other molecular formulas) illustrate a comparison between SMOG and GENM (see also Figure 6). In all examples shown in these tables, the numbers of isomers generated by SMOG are smaller compared to GENM. However, after retrospective removal of duplicates[13c] from the sets of results obtained by GENM, the final numbers

Table 2. Comparison of the GENM and SMOG Generators: Data for Other Molecular Formulas

			no. of structures			success/failure ratio	
			GENM				
no.	Molecular formula	core fragments (in Figure 6)	generated	after removal of duplicates	SMOG	connectivity test	strong canonicity test
1	$C_6H_{10}O_4$	11	1 974	1 971	1 971	1.24	1.75
2	$C_6H_{10}N_2$	12	7 644	6 739	6 739	18.31	1.86
3	$C_{12}H_{20}N_2O$	13, 14	32 554	15 778	15 778	4.80	1.94

of *different* structures are invariably equal to the numbers of structures produced by SMOG. (Examples where GENM generated a completely duplicate-free set of isomers are not shown in Table 1; in all those examples, both programs produced equal numbers of isomers.) That is, apparently, SMOG indeed yields all and only nonisomorphic molecular graphs in the cases listed in the tables.[41]

Note that both problematic cases (the need to account for the symmetry of fragments, as in Figure 2a, and the possible assembly of additional copies of core fragments during generation, as in Figure 2b) are handled successfully by SMOG (*e.g.*, examples 1 and 8 in Table 1 or examples 1 and 3 in Table 2).

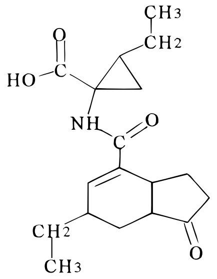
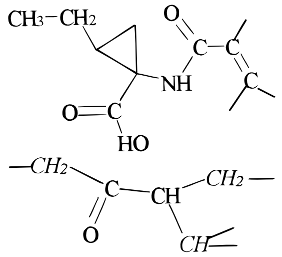
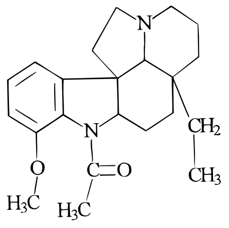
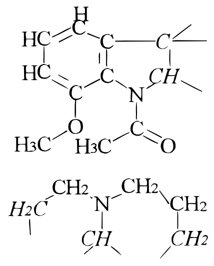
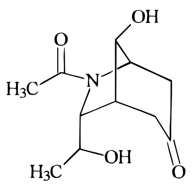
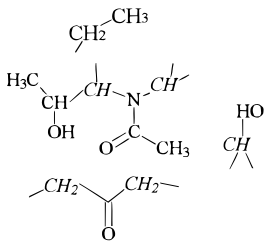
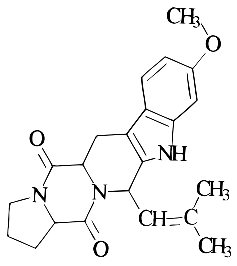
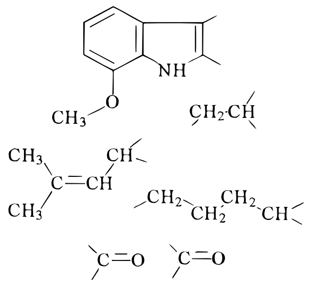
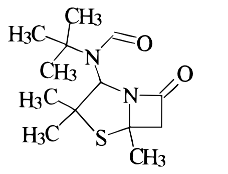
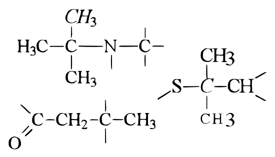
Of course, everything comes at a cost. As to exclusion of duplicates, comparison with GENM is favorable for our generator; as to efficiency, not so favorable. In our attempts to get rid of redundancy, we have to make a very rigorous canonicity test at each generation step rather than employ the efficient semiheuristic procedures developed in GENM (as was shown by sampler and profiler utilities, the test for strong canonicity in SMOG is responsible for 70−90% of the generation time). As a result, the process of isomer construction in the presence of core fragments is dramatically retarded.[42] Indeed, as is apparent from Table 1, the performance of SMOG on a Pentium/90 MHz personal computer is comparable to that of GENM on a 386/20 MHz one! However, in spite of this comparative slowness, we believe that SMOG still may sometimes be more useful for rapid computation of the actual number of the desired isomers, because the search for duplicates among numerous molecules generated by GENM may take a long additional time and must become an increasingly difficult task for large sets of results.

As an additional interesting parameter that may be used for characterization of the program performance, we used the success/failure ratio for the strong canonicity test (that is, the ratio between the numbers of positive/negative answers given by this test during generation; see Figure 4); for a more complete description, we also found the same ratio for the connectivity test, which is performed at each step before the strong canonicity test (see the last two columns in Table 1 or 2). Note that the success/failure ratio for the strong canonicity test is lower than this ratio for the connectivity test but nevertheless moderately high in most of the problems considered. If we make allowance for the fact that the number of theoretically possible noncanonical structures is usually much higher than the number of canonical ones, the success/failure ratio for the strong canonicity test may be fairly high because of two possible reasons: either the overwhelming majority of noncanonical intermediate structures are discarded because the matrix is filled according to the principle of weakly canonical filling (see section 2.5.2.1) or the test for strong canonicity indeed removes the majority of unsatisfactory structures at the earliest possible stages of generation, thus effectively reducing the number of failures in such tests at subsequent steps. In fact, both explanations seem to be valid. For example, if we remove the test for strong canonicity (preserving only the principle of weakly canonical filling), the number of adjacency matrices generated by the program increases by a factor of 2−15 (for the problems considered in Table 1) at the expense of noncanonical results. However, the fraction of positive answers in the strong canonicity test is notably higher than the inverse of this factor ($^1/_{15}$ − $^1/_2$), which means that most noncanonical structures are actually “collectively” excluded at early stages of generation.

Finally, the examples in Table 1 were used to estimate the efficiency of the three techniques listed in section 2.5.2. This estimation consisted of comparison of the generation times for the original version of SMOG and for the specially designed version where the procedures corresponding to one of these techniques (the one to be estimated) were excluded. Separation into sets of stability and the use of the weak canonicity rule (section 2.5.2.1) ensured less than a 2-fold increase in the program performance for the cases in Table 1; however, this technique must play a much more important role in cases where core fragments are small compared to the size of target molecules (because the number of core-invariant permutations is greater in such cases). Preliminary exclusion of some permutations from consideration (section 2.5.2.2) makes generation 2−10 times faster (depending on the number of atoms with different neighbors and “incompatible” environments in the core fragments). Finally, stepwise construction of core-invariant permutations (section 2.5.2.3) plays the most important role for the cases in Table 1: the generation times with and without this technique may differ more than by 2 orders of magnitude.

In addition to problems in Tables 1 and 2, we also tested SMOG by examples of some actual tasks that had earlier been encountered in the field of computer-assisted structure elucidation and handled with the use of other generators. These examples are listed in Table 3 (as in Figure 6, italicized labels at some atoms in the “core fragments” column mean that these atoms may form no additional covalent bonds with hydrogen[43]). In all cases considered, SMOG generated apparently duplicate-free sets of isomers that contained the actual structures of the target compounds among other molecular graphs. Naturally, when other experimental constraints were introduced in addition to the set of core fragments (BADLIST, the sizes of cycles, the hybridization states of atoms, *etc.*), the numbers of molecules generated by SMOG usually decreased by several orders of magnitude, but, of course, the sets of isomers remained irredundant and still contained the actual structures.[44−46]

Table 3. Examples of Generation with the Use of SMOG: Some Actual Problems Encountered in Computer-Assisted Structure Elucidation

trivial name of compd	molecular formula	refs	actual structural formula	generation by SMOG: core fragments	generation by SMOG: no. of structures
coronatine	$C_{18}H_{25}NO_4$	6c, 44			3759
aspidospermine	$C_{22}H_{30}N_2O_2$	3b			3853
actinobolamine	$C_{11}H_{17}NO_4$	45			11
	$C_{22}H_{25}N_3O_3$	9c			72678
	$C_{13}H_{22}N_2O_2S$	11, 46			3142

Many other tests (not described in this paper) also invariably showed that the above-described approach indeed ensured both exhaustiveness of generation and complete exclusion of duplicates in the presence of arbitrary core fragments.

4. CONCLUSION

As was stated above, many authors of existing isomer generators fully or partially ignored the important problem of ensuring irredundancy during generation in the presence of known multiatomic fragments of arbitrary structure. A way to solve this problem without the use of an additional retrospective check for duplicates was suggested in this paper. Considering the present-day scarcity of literature data on results of generation with the use of arbitrary fragments, we may hope that our program SMOG (after sufficient additional testing) can serve as a kind of a reference tool in this respect.

Presently, a new version of SMOG is under development. We expect it to be several times faster than the present one due to better implementation of the algorithm and the use of some heuristics as auxiliary symmetry criteria in the determination of canonicity. However, a rigorous test for strong canonicity will be preserved.

Of course, as far as generation of large molecules is concerned, SMOG has got the same inevitable limitations as all deterministic generators (the time required for solution of generation problems exponentially increases with an increasing number of atoms), but it is applicable for small- and medium-sized molecules (for example, up to 30 non-hydrogen atoms), especially if large core fragments are known in advance.

SMOG runs on IBM PC-compatible computers. It is a freeware program supplied with a user-friendly graphic interface, a detailed manual, and on-line help. SMOG is presently available from Archives of Computational Chemistry List (top page http://ccl.osc.edu/chemistry.html) or from the FTP server of the Organic Chemistry Division (Chemistry Department, Moscow State University).[47]

REFERENCES AND NOTES

(1) Molchanova, M. S.; Shcherbukhin, V. V.; Zefirov, N. S. Computer Generation of Molecular Structures by the SMOG Program. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 888−899.

(2) Lederberg, J.; Sutherland, G. L.; Buchanan, B. G.; Feigenbaum, E. A.; Robertson, A. V.; Duffield, A. M.; Djerassi, C. Applications of Artificial Intelligence for Chemical Inference. I. The Number of Possible Organic Compounds. Acyclic Structures Containing C, H, O, and N. *J. Am. Chem. Soc.* **1969**, *91*, 2973−2976.

(3) (a) Masinter, M.; Sridharan, N. S.; Lederberg, J.; Smith, D. H. Applications of Artificial Intelligence for Chemical Inference. XII. Exhaustive Generation of Cyclic and Acyclic Isomers. *J. Am. Chem. Soc.* **1974**, *96*, 7702−7714. (b) Carhart, R. E.; Smith, D. H.; Brown, H.; Djerassi, C. Applications of Artificial Intelligence for Chemical Inference. XVII. An Approach to Computer-Assisted Elucidation of Molecular Structure. *J. Am. Chem. Soc.* **1975**, *97*, 5755−5762.

(4) (a) Smith, D. H.; Gray, N. A. B.; Nourse, J. G.; Crandell, C. W. The Dendral Project: Recent Advances in Computer-Assisted Structure Elucidation. *Anal. Chim. Acta* **1981**, *133*, 471−497. (b) Carhart, R. E.; Smith, D. H.; Gray, N. A. B.; Nourse, J. G.; Djerassi, C. GENOA: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures. *J. Org. Chem.* **1981**, *46*, 1708−1718. (c) Lindley, M.; Schoolery, J.; Smith, D.; Djerassi, C. Application of the Computer Program GENOA and Two-Dimensional NMR Spectroscopy to Structure Elucidation. *Org. Magn. Reson.* **1983**, *21*, 405−411.

(5) Lindsay, R. K.; Buchanan, B. G.; Feigenbaum, E. A.; Lederberg, J. *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*; McGraw-Hill: New York, 1980.

(6) (a) Shelley, C. A.; Hays, T. R.; Roman, R. V.; Munk, M. E. An Approach to Automated Partial Structure Expansion. *Anal. Chim. Acta* **1978**, *103*, 121−132. (b) Shelley, C. A.; Munk, M. E. CASE, a Computer Model of the Structure Elucidation Process. *Anal. Chim. Acta* **1981**, *133*, 507−516. (c) Lipkus, A. H.; Munk, M. E. Automated Classification of Candidate Structures for Computer-Assisted Structure Elucidation. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 9−18.

(7) Christie, B. D.; Munk, M. E. Structure Generation by Reduction: A New Strategy for Computer-Assisted Structure Elucidation. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 87−93.

(8) Zhu, S.; Zhang, J. Exhaustive Generation of Structural Isomers for a Given Empirical Formula—A New Algorithm. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 34−38.

(9) (a) Grund, R.; Kerber, A.; Laue, R. MOLGEN, a Computer Algebra System for Construction of Molécular Graphs. *Commun. Math. Chem. (MATCH)* **1992**, *27*, 87−131 (in German). (b) Wieland, T.; Kerber, A.; Laue, R. Principles of the Generation of Constitutional and Configurational Isomers. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 413−419. (c) Benecke, C.; Grund, R.; Hohberger, R.; Kerber, A.; Laue, R.; Wieland, T. MOLGEN+, a Generator of Connectivity Isomers and Stereoisomers for Molecular Structure Elucidation. *Anal. Chim. Acta* **1995**, *314*, 141−149. (d) Benecke, C.; Grund, R.; Kerber, A.; Laue, R.; Wieland, T. Chemical Education via MOLGEN. *J. Chem. Educ.* **1995**, *72*, 403−406. (e) Wieland, T. The Use of Structure Generators in Predictive Pharmacology and Toxicology. *Arzneim.-Forsch. (Drug Res.)* **1996**, *46* (I), 223−227.

(10) (a) Funatsu, K.; Miyabayashi, N.; Sasaki, S. Further Development of Structure Generation in the Automated Structure Elucidation System CHEMICS. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 18−28. (b) Funatsu, K.; Susuta, J.; Sasaki, S. Introduction of Two-Dimensional NMR Spectral Information to an Automated Structure Elucidation System CHEMICS: Utilization of 2D-INADEQUATE Information. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 6−11. (c) Funatsu, K.; Sasaki, S. Recent Advances in the Automated Structure Elucidation System CHEMICS. Utilization of Two-Dimensional NMR Spectral Information and Development of Peripheral Functions for Examination of Candidates. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 190−205.

(11) Bauer, J.; Fontain, E.; Ugi, I. IGOR and RAIN-the First Mathematically Based Multipurpose Problem-Solving Computer Programs for Chemistry and Their Use as Generators of Constitutional Formulas. *Commun. Math. Chem. (MATCH)* **1992**, *27*, 31−47.

(12) (a) Bohanec, S.; Zupan, J. Structure Generator GEN. *Commun. Math. Chem. (MATCH)* **1992**, *27*, 49−85. (b) Bohanec, S. Structure Generation by the Combination of Structure Reduction and Structure Assembly. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 494−503.

(13) (a) Molodtsov, S. G.; Piottukh-Peletsky, V. N. Generation of All Nonisomorphic Chemical Graphs from a Given Set of Structural Fragments. In *Algorithms for the Analysis of Structural Information*; Vychislitel'nye Sistemy: Novosibirsk, Russia, 1984; Vol. 103, pp 51−58 (in Russian). (b) Molodtsov, S. G. Computer-Aided Generation of Molecular Graphs. *Commun. Math. Chem. (MATCH)* **1994**, *30*, 213−224. (c) Molodtsov, S. G. Generation of Molecular Graphs with a Given Set of Nonoverlapping Fragments. *Commun. Math. Chem. (MATCH)* **1994**, *30*, 203−212.

(14) Luinge, H. J. AEGIS, A Structure Generation Program in Prolog. *Commun. Math. Chem. (MATCH)* **1992**, *27*, 175−189.

(15) Contreras, M.; Valdivia, R.; Rozas, R. Exhaustive Generation of Organic Isomers. (a) 1. Acyclic Structures. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 223−230. (b) 2. Cyclic Structures: New Compact Molecular Code. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 481−491. (c) 3. Acyclic, Cyclic, and Mixed Compounds. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 610−616.

(16) Hu, Ch.-L.; Xu, L. Principles for Structure Generation of Organic Isomers from Molecular Formula. *Anal. Chim. Acta* **1994**, *298*, 75−85.

(17) Faulon, J.-L. On Using Graph-Equivalence Classes for Structure Elucidation of Large Molecules. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 338−348.

(18) Faulon, J.-L. Stochastic Generator of Chemical Structure. (a) 1. Application to the Structure Elucidation of Large Molecules. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1204−1218. (b) 2. Using Simulated Annealing to Search the Space of Constitutional Isomers. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 731−740.

(19) Bangov, I. P. Computer-Assisted Structure Generation from a Gross Formula. (a) 3. Alleviation of the Combinatorial Problem. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 277−289. (b) 4. Fighting against Graph-Isomorphism Disease. *Commun. Math. Chem. (MATCH)* **1992**, *27*, 3−30. (c) 7. Graph Isomorphism: A Consequence of the Vertex Equivalence. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 318−324.

(20) Abe, H.; Yamasaki, T.; Fujiwara, I.; Sasaki, S. Computer-Aided Structure Elucidation Methods. *Anal. Chim. Acta* **1981**, *133*, 499−506.

(21) Gray, N. A. B. *Computer-Assisted Structure Elucidation*; Wiley: New York, 1986.

(22) In ref 13c, the term "GOODLIST" is used in the same sense as "core fragments" in this paper. In our terminology, however, the difference between GOODLIST and the set of core fragments is rather subtle but still notable. In fact, GOODLIST is the whole set of fragments that must be present in the target molecules. At the same time, *some* fragments from GOODLIST (not necessarily all of them; see ref 1) may be used as ready construction units during generation, and only these fragments are called core fragments. Another subtle difference from the terminology of ref 13c is described in note 37.

(23) Pólya, G. Combinatorial Enumeration for Groups, Graphs, and Chemical Compounds. *Acta Sci. Math.* **1937**, 145−254.

(24) Randic, M. Recognition of Identical Graphs Representing Molecular Topology. *J. Chem. Phys.* **1974**, *60*, 3920−3928.

(25) Randic, M. On Canonical Numbering of Atoms in a Molecule and Graph Isomorphism. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 171−180.

(26) (a) Kvasnicka, V.; Pospichal, J. Canonical Indexing and Constructive Enumeration of Molecular Graphs. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 99−105. (b) Kvasnicka, V.; Pospichal, J. An Improved Version of the Constructive Enumeration of Molecular Graphs with Prescribed Sequence of Valence States. *Chemom. Intell. Lab. Syst.* **1993**, *18*, 171−181.

(27) Other fragments are used indirectly within CHEMICS and must be checked for their presence in the generated structures. Also, CHEMICS is generally known to be somewhat nonrigorous as to the exclusion of duplicates (for example, see refs 12a, 15c, *etc.*).

(28) To extend the notion of the automorphism group for a set of free valences, we represent all of the free valences by imaginary bonds that connect the corresponding atom(s) of the fragment to some dummy atoms (bonds with an unspecified terminus in Figure 2). Thus, we obtain the "expanded" graph of the fragment. The automorphism group of this expanded graph is regarded as the automorphism group of the set of free valences.

(29) We had no possibility to check if the approach of MOLGEN finally yields irredundant results for arbitrary sets of core fragments, because recent papers describing this software (refs 9b−e) do not contain the data sufficient for proving or refuting this assumption. It would be especially interesting to see how the situation associated with

construction of additional copies of fragments during generation (Figure 2b) is handled.

(30) (a) Faradjev, I. A. Constructive Enumeration of Combinatorial Objects. In *Algorithmic Investigations in Combinatorics*; Faradjev, I. A., Ed.; Nauka: Moscow, 1978; pp 3−11 (in Russian). (b) Faradjev, I. A. Generation of Nonisomorphic Graphs with a Given Degree Sequence. In *Algorithmic Investigations in Combinatorics*; Faradjev, I. A., Ed.; Nauka: Moscow, 1978; pp 11−19 (in Russian). (c) Zaichenko, V. A.; Ivanov, A. B.; Rosenfeld, M. Z.; Faradjev, I. A. Algorithm of Canonicity Verification for a Partially Filled Adjacency Matrix of a Graph. In *Algorithmic Investigations in Combinatorics*; Faradjev, I. A., Ed.; Nauka: Moscow, 1978; pp 19−25 (in Russian).

(31) A version of Faradjev's approach is also used in Kvasnicka's generator.[26b] However, we will not consider this program in the present paper, because it is not aimed at direct use of arbitrary multiatomic fragments.

(32) Harary, F. *Graph Theory*; Addison-Wesley: Reading, MA, 1969.

(33) Actually, any other univalent atom may play the same implicit role in the generation of molecular graphs as hydrogen: for example, generation is identical for the molecular formulas C_6H_6 and C_6Cl_6.[1] However, to avoid unnecessary complications, hereafter we refer to such univalent atoms simply as hydrogens.

(34) Hendrickson, J. B.; Toczko, A. G. Unique Numbering and Cataloguing of Molecular Structures. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 171−177.

(35) Structural generation is aimed at producing connected molecular graphs, which correspond to covalently bound molecules. Methods for testing whether any full complement of a partially filled matrix may correspond to a connected structure (*i.e.*, whether a partially assembled structure contains any disconnected parts without free valences) are known and simple, so we will not dwell further on this problem.

(36) Rosenfeld, M. Z. Construction and Properties of Some Classes of Strongly Regular Graphs. *Usp. Mat. Nauk* **1973**, *28*, 197−198 (in Russian).

(37) The term used in ref 13c is "pieces of stability" rather than "sets of stability". We changed this term in order to stress that these sets may be represented not only by single ranges ("pieces") but also by several disjoint ranges in our case.

(38) Another way of reducing the search in the $S(I,J,C)$ group is consideration of its orbits, according to Faradjev.[30c]

(39) Nechepurenko, M. I.; Popkov, V. K.; Mainagashev, S. M.; Kaul', S. B.; Proskuryakov, V. A.; Kokhov, V. A.; Gryzunov, A. B. *Algorithms and Programs for the Solution of Problems on Graphs and Networks*; Nauka: Novosibirsk, 1990; pp 123−157 (in Russian).

(40) Another complication is due to the fact that literature data on the efficiency of different programs are not fully suitable for comparison, because the data sets available for different generators actually represent generation times that refer to the use of different computers with varying performances and processor types. As the possible way out, the authors of the GEN generator[12b] proposed a comparison of algorithms with respect to their formal computational complexity; however, they also admitted that a lack of literature data and a strong dependence of the computational complexity on the actual problem under consideration make such a comparison impossible, at least presently.

(41) Aromatic rings, such as benzene or pyridine, are regarded in this paper as formally consisting of alternating single and double bonds (rather than "aromatic" bonds of some intermediate multiplicity), because the same approach is used by the majority of existing generators, including GENM. That is, single and double bonds in aromatic rings are viewed as topologically nonequivalent. In general, SMOG can also handle them as equivalent, according to the user's desire, as is described in ref 1; in this case, the numbers of structures in some examples of Table 1 are additionally reduced.

(42) As a result, if small (for example, two-atom) fragments are included in GOODLIST, sometimes it may be even preferable to assemble structures from separate atoms and check molecular graphs for the presence of these fragments during assembly (the straightforward approach mentioned in section 1.2!) rather than use these fragments directly as construction units. For larger fragments (≥3 atoms), however, the latter approach becomes incomparably more efficient.

(43) For such atoms, the condition $H_{g(i)} \geq H_i$ at the end of section 2.1 is transformed into $H_{g(i)} = H_i$.

(44) Ichibara, A.; Shiraishi, K.; Sato, H.; Sakamura, S.; Nishiyama, K.; Sakai, R.; Furusaki, A.; Matsumoto, T. The Structure of Coronatine. *J. Am. Chem. Soc.* **1977**, *99*, 636−637.

(45) Munk, M. E.; Sodano, C. S.; McLean, R. L.; Haskell, T. H. Actinobolin: I. Structure of Actinobolamine. *J. Am. Chem. Soc.* **1967**, *89*, 4158−4165.

(46) Ugi, I.; Wischhöfer, E. Isonitrile XI: Synthesis of a Simple Derivative of Penicillic Acid. *Chem. Ber.* **1962**, *95*, 136−140.

(47) The distribution archive of SMOG (the MS-DOS version) may be obtained via FTP (either ftp://org.chem.msu.su/pub/chemistry/software/SMOG or ftp://ccl.osc.edu/pub/chemistry/software/MS-DOS/SMOG). Using the first of these URLs, you may also obtain the "lite" version of the program, which was actually used for calculations cited in this paper. Any suggestions concerning possible improvement of the program functionality or removal of bugs will be accepted with gratitude. Please mail them to mary@lexa.ru.

CI970015P