SANDIA REPORT
SAND 2004-0960
Unlimited Release
Printed April 2004

Enumerating Molecules

Jean-Loup Faulon, Donald P. Visco, Jr., and Diana Roe

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Sandia National Laboratories

132

SAND 2004-0960
Unlimited Release
Printed April 2004

Enumerating Molecules

Jean-Loup Faulon[*]
Computational Biology Department
Sandia National Laboratories
P.O. Box 969, MS 9951
Livermore, CA

Donald P. Visco, Jr.
Department of Chemical Engineering
Tennessee Technological University
Box 5013
Cookeville, TN

Diana Roe
Biosystems Research Department
Sandia National Laboratories
P.O. Box 969, MS 9951
Livermore, CA

ABSTRACT

This report is a comprehensive review of the field of molecular enumeration from early isomer counting theories to evolutionary algorithms that design molecules in silico. The core of the review is a detail account on how molecules are counted, enumerated, and sampled. The practical applications of molecular enumeration are also reviewed for chemical information, structure elucidation, molecular design, and combinatorial library design purposes. This review is to appear as a chapter in *Reviews in Computational Chemistry* volume 21 edited by Kenny B. Lipkowitz.

ACKNOWLEDGMENT

Funding for this work was provided by the Math. Information and Computer Science program of the U.S. Department of Energy and Sandia National Laboratories under grant number DE-AC04-94AL85000.

CONTENTS

Figures

Tables

Equations

Enumerating Molecules: Why

Enumerating molecules is a mind-boggling problem that has fascinated chemists and mathematicians alike for more than a century. Taking the definition from various dictionaries, to enumerate means (1) "to name things separately, one by one", and (2) "to determine the number of, to count." Interestingly enough, both definitions have been taken when enumerating molecules. Historically, the latter definition was first used, and mathematical solutions were devised to count molecules. Some of the solutions developed were not only valuable to chemists but to mathematicians as well. Indeed, as we shall see in this chapter, while trying to solve the problem of counting the isomers of paraffin structures[1] or counting substituted aromatic compounds,[2] important concepts in graph theory and combinatorics were developed. The terms *graph* and *tree* were even coined in a chemistry context.[3]

About four decades ago, with the advance of computer science, researchers started to look at the former definition of enumeration, and devised computer codes to explicitly list molecules. Again while studying this challenging problem, important concepts in computer science were developed. Artificial intelligence textbooks[4] generally quote DENDRAL, a code to enumerate molecules, as the first expert system.

Historically, molecular enumeration has brought a fertile ground of research between chemistry, mathematics, and computer science. Still today new concepts and techniques are being developed at the interstice of these fields.[5]

Enumerating molecules is not only an interesting academic exercise but has practical applications as well. The foremost application of enumeration is structure elucidation. Ideally, the wishful bench chemist collects experimental data (NMR, MS, IR,...) for an unknown compound, the data is fed to a code, and the resulting unique structure is given back. Although such a streamlined picture is not yet fully automated, and may never be, there are commercial codes that can, for instance, list all structures matching a given molecular formula, an IR spectrum, or an NMR spectrum. Another important application is in molecular design. Here the problem is to design compounds (drugs, for example) that optimize some physical, chemical, or biological property or activity. Although not as prolific as structure elucidation, molecular design has introduced some novel stochastic solutions to molecular enumeration. Finally, with the advent of combinatorial chemistry, molecular enumeration takes a central role as it allows computational chemists to construct virtual libraries, test hypotheses, and provide guidance to design optimal combinatorial experiments.

Our primary goal in this chapter is to explain how molecules are enumerated. This is the objective of the first section. We start with the problem of counting molecules, then describe how molecules are explicitly enumerated, and finish with a review of stochastic techniques to sample molecules. Our discussion is directed toward structure elucidation and molecular design. However, these applications use nearly all aspects of counting, enumerating, and sampling. Prior to understanding how molecules can be elucidated and designed, important theoretical concepts and interesting results relevant to chemistry have to first be assimilated.

The purpose of the second section of this chapter is to review the practical applications of molecular enumeration and to give the reader interested in any of these applications pointers to relevant codes and techniques. In particular, the numbers of isomers for specific molecular series are given, popular structure elucidation codes are reviewed, computed-aided structure elucidation successes are surveyed, and the connections between structure enumeration and combinatorial library design are established. The field of molecular design using inverse quantitative structure activity relationship is also reviewed. We conclude the chapter outlining future research directions.

Before we start, we want to point out that this chapter is limited to structural (i.e., 2D) enumeration and does not cover conformational (i.e., 3D) enumeration. This latter topic has already been discussed in the book series for small and medium-sized molecules[6] and peptides.[7]

Enumerating Molecules: How

The term *enumerating* has been used in the literature for both listing molecules one by one and determining the number of molecules corresponding to a given set of constraints. In this chapter, we use the term *counting* for the latter case, and we utilize the term *enumerating* only when molecules are explicitly listed. Starting with some elementary definitions from graph theory, we then describe how molecules are counted, enumerated, and finally stochastically sampled. The counting, enumerating, and sampling subsections can be read separately. While counting is mostly solved through mathematical treatments, enumerating and sampling are essentially algorithmic problems. In each of the following subsections, theoretical results are first explained and illustrated with examples relevant to chemistry. Second, chemical applications are surveyed. To illustrate the problem being studied, a question is attached to each subsection. The answers of the questions can be found in the text.

From Graph Theory to Chemistry

We provide here elementary definitions later used to count, enumerate and sample molecules. Rather than a formal mathematical presentation, examples and illustrations are given.

A *simple graph* G is defined as an ordered pair $G = (V(G),E(G))$, where $V = V(G)$ is a nonempty set of elements called *vertices*, and $E = E(G)$ is a set of unordered pairs of distinct element of V called *edges*. In most cases of chemical interest the sets V and E are finite. An example of a simple graph is given in Figure 1.(a). Of course, there is a relationship between graphs and chemical structures. Sylvester[3] proposed the term graph in 1878 on the basis of the structural formulae of molecules. Figure 1.(a) can, for instance, be viewed as a representation of cyclohexane. But there are molecules that do not fit the simple graph picture. A *multigraph* is a graph where the edge set is not necessarily composed of distinct pair of vertices, in other words, *multiple edges* are allowed in a multigraph. A multigraph is without a loop when vertices are not allowed to be paired with themselves. Figure 1.(b) is a representation of benzene. In a simple graph or a multigraph, the *degree* of a vertex is the number of edges attached to it, and the *multiplicity* of an edge is the number of times that edge occur in the graph. In Figure 1 graph (a) contains vertices of degree 1 and 4, and all edges have multiplicity 1; in multigraph (b) the vertices have degrees 1 and 4 and the edges have multiplicities 1 and 2. The *degree sequence* of a graph or a multigraph is the sequence of numbers of vertices having a given degree starting with degree 0 and ending with the maximum degree for all vertices. Graph (a) in Figure 1 has no vertices of degree 0, 12 vertices of degree 1, no vertices of degree 2 and degree 3, and 6 vertices of degree 4, the degree sequence is (0,12,0,0,6). Graph (b) has the degree sequence (0,6,0,0,6).

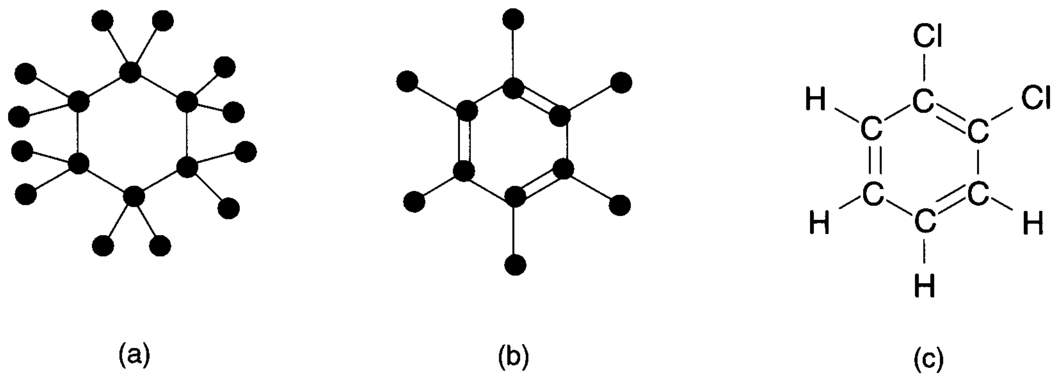


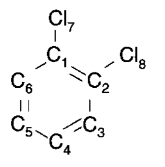
Figure 1. (a) Simple graph, (b) multigraph, and (c) molecular graph

While Figure 1.(b) could correspond uniquely to benzene, one cannot distinguish 1,2-dichlorobenzene from 1,4-dichlorobenzene using this representation. To make the distinction between the two compounds one has to attach to each vertex, a label, or color, that is unique to each element of the periodic table (for instance, the atomic symbol). Finally, in a molecular structure atoms are always connected through some bonds, in other words, a molecular structure is in one piece. A *molecular graph* is thus defined as a connected multigraph with vertices colored by the atomic symbols of the periodic table. We use the term color instead of label since, as we shall see next, labeled graphs have a specific definition in graph theory. Figure 1.(c) is the molecular graph of 1,2-dichlorobenzene. Clearly, in a molecular graph, each vertex is an atom and each edge is a bond. The terms *atom valence* replace the terms vertex degree, and *bond order* replace edge multiplicity. Note that with the exception of rare gases, a molecular graph comprises more than one atom. Because molecular graphs are connected, their valence sequences start with valence 1 and usually end with valences 4 or 5 for most organic compounds. The valence sequence of benzene is (6,0,0,6).

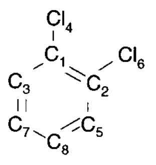
Now that we have defined molecular graphs, we need to find an appropriate representation for computer manipulation and storage. Assuming our molecular graph G has n

atoms, we first start to label each atom with numbers 1 through n. We then create a vector of n entries where each entry i, $1 \leq i \leq n$, is the symbol of atom i. We also create an nxn matrix called the *adjacency matrix*, where each entry i,j, $1 \leq i,j \leq n$ is set to the order of the bond between atom i and atom j. The maximum bond order is 3, and the order is set to 0 when the two atoms are not bonded. Examples of adjacency matrices are given in Figure 2. Note that the diagonals of the adjacency matrices are filled with 0s, as atoms are not bonded to themselves. Adjacency matrices are symmetric matrices, since when atom i is bonded to j, atom j is also bonded to i. A convenient way to store adjacency matrices into a compact code was introduced by Kudo and Sasaki.[8] This code, called the connectivity stack, is obtained by reading the upper triangle of the adjacency matrix row by row from left to right. Examples of connectivity stacks are given in Figure 2. Connectivity stacks can be compared. Let $A = a_1a_2\ldots\ a_i \ldots$ and $B = b_1b_2\ldots b_i \ldots$ be two connectivity stacks where a_i and b_i take the values 0,1, 2, or 3. We then write $A \geq B$ if there is an index i such that $a_i \geq b_i$, $a_{i-1} = b_{i-1},\ldots, a_1 = b_1$. Taking the example of Figure 2 the connectivity stack of graph G_2 is greater than the connectivity stack of graph G_1.

G_1



G_2



$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\
2 & 2 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
3 & 0 & 1 & 0 & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
4 & 0 & 0 & 2 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
5 & 0 & 0 & 0 & 1 & 0 & \mathbf{2} & \mathbf{0} & \mathbf{0} \\
6 & 1 & 0 & 0 & 0 & 2 & 0 & \mathbf{0} & \mathbf{0} \\
7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \\
8 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\qquad
\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & \mathbf{2} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
2 & 2 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\
3 & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{2} & \mathbf{0} \\
4 & 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
5 & 0 & 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{2} \\
6 & 0 & 1 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} \\
7 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & \mathbf{1} \\
8 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0
\end{array}$$

2000110 100001 20000 200 00 0 **2110000 001100 00020 0000 002 00 1**

Figure 2. Two hydrogen-suppressed molecular graphs with corresponding adjacency matrices and connectivity stacks.

In order to code a molecular graph we have to label all the atoms of our graph, and transform the graph into what is called in graph theory a *labeled graph*, i.e., a graph for which each vertex has a distinct label. This, of course, does not mean that there is a one-to-one correspondence between molecules and labeled graphs, as the two different labeled graphs shown in Figure 2 correspond to the same molecule. There are some instances however, where graphs used in chemistry can be appropriately represented by labeled graphs. For instance, in linear reaction networks and protein and gene networks all vertices have a unique label (e.g., a compound name). Another example is combinatorial libraries obtained by attaching reactants to a scaffold having no symmetry. In this case, the reactants have a unique name, and the reacting sites on the scaffold being different, can be labeled uniquely. In general, molecules should not be considered as labeled graphs, and as we shall see in this chapter, the techniques used to count, enumerate, and sample molecules are all derived from combinatorial results obtained with *unlabeled* graphs.

While molecules are not appropriately represented by labeled graphs, they are stored and manipulated (by computers) as such. We thus need to find a way to detect when two labeled representations correspond to the same molecular structures. Two labeled representations correspond to the same (unlabeled) graph if a one-to-one mapping between the two sets of labels can be found to also map the edges of the graphs. This mapping is called *isomorphism*. Formally, two labeled graphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ are isomorphic if a one to one mapping π from V_1 to V_2 can be found such that $\pi(E_1) = E_2$. Note that with molecular graphs one has to restrict the mapping π between atoms having the same atomic symbol. Using the notation $(i\ j)$ to specify that atom i from graph G_1 is mapped to atom j in graph G_2, the isomorphism (1 1)(2 2)(3 5)(4

8)(5 7)(6 3)(7 4)(8 6) maps graph G_1 to graph G_2 in Figure 2. Note that (1 2)(2 1)(3 3)(4 7)(5 8)(6 5)(7 6)(8 4) is also an isomorphism from G_1 to G_2.

Because several labelings of the same molecular graph can occur, it is important to distinguish one of them. We shall call this labeling the *canonical* one. There are several ways of obtaining a canonical labeling. The one we chose in this chapter is the one leading to the maximal connectivity stack. Taking all the possible 8! = 40320 possible labeling of the graphs in Figure 2, one can verify (with the help of a computer code) that there is no other labeling of the vertices having a connectivity stack greater than the one of graph G_2. Of course there are better ways of canonizing a connectivity stack than checking all the possible labelings. Algorithms capable of doing this can be found in the literature[9] but reviewing them is not the purpose of this chapter. The computational complexity of canonizing a general graph is unknown, i.e., no fast algorithm has yet been found. However, it has been shown that molecular graphs can theoretically be canonized efficiently.[9] Furthermore some rather fast graph canonizers such as Brendan McKay's code Nauty[10] can easily be adapted to canonize molecular graphs.

From the definition of isomorphism given earlier, two atoms x_1 of graph G_1 and x_2 of graph G_2 are isomorphic if an isomorphism can be found matching x_1 to x_2 and matching the bonds of G_1 to those of G_2. For instance in Figure 2 atom 4 in G_1 is isomorphic to atom 8 in G_2. Now, if $G_1 = G_2$, we say that the two atoms are *equivalent* and instead of isomorphism we use the term *automorphism.* Taking again the example of graph G_1 in Figure 2, and taking again the notation $(i\ j)$ to specify that atom i is mapped to atom j, the mapping (1 1)(2 2)(3 6)(4 5)(5 4)(6 3)(7 7)(8 8) of the vertices of G_1 leads to a graph identical to G_1. That mapping, also called a *permutation*, is an automorphism. The permutation notation can be simplified noting that when $(i$

j) occurs (*j i*) also occurs, and writing (*i*) instead of (*i i*). Thus, the permutation (1 1)(2 2)(3 6)(4 5)(5 4)(6 3)(7 7)(8 8) reduces to (1)(2)(36)(4 5)(7)(8).

Several isomorphisms may exist between two graphs (cf. G_1 and G_2 in Figure 2). Similarly, a given graph may have several automorphisms. The *automorphism group* of a graph is the set of all of its automorphisms. The automorphism group of the hydrogen suppressed molecular graph of benzene is given in Figure 3. This group is the dihedral D_{6h} group.

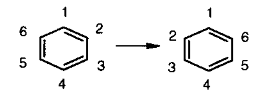
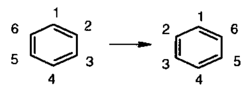
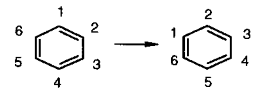
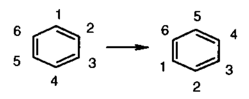
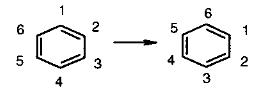
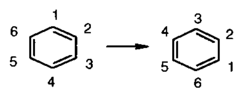
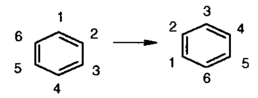
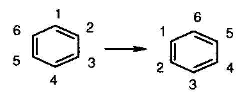
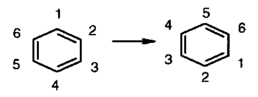
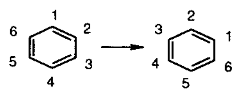
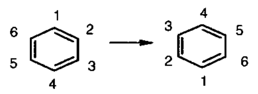
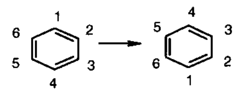
Symmetry operation		Permutation	Symmetry operation		Permutation
E		(1)(2)(3)(4)(5)(6)	$\sigma_v^{(1)}$		(1)(4)(2 6)(3 5)
C_6^-		(1 2 3 4 5 6)	$\sigma_v^{(2)}$		(3)(6)(1 5)(2 4)
C_6^+		(6 5 4 3 2 1)	$\sigma_v^{(3)}$		(2)(5)(1 3)(4 6)
C_3^-		(1 3 5)(2 4 6)	$\sigma_v^{(4)}$		(1 6)(2 5)(3 4)
C_3^+		(5 3 1)(6 4 2)	$\sigma_v^{(5)}$		(1 2)(3 6)(4 5)
C_2		(1 4)(2 5)(3 6)	$\sigma_v^{(6)}$		(1 4)(2 3)(5 6)

Figure 3. List of all (12) automorphisms for hydrogen suppressed benzene. In the permutation notation (1 2) reads "1 goes to 2 and 2 goes to 1", (1 3 5) reads "1 goes to 3, 3 goes to 5, and 5 goes to 1."

Because two atoms are equivalent if they can be mapped by an automorphism, one can partition the atoms into *equivalent classes* using the automorphism group of a graph. In graph theory, the atom equivalent classes are named the orbits of the automorphism group. Chemically, atoms that belong to the same equivalent class are symmetrical, and among other properties have the same chemical shift in NMR spectra. There are many algorithms in the chemistry literature to compute the atom equivalent classes of molecular graphs.[9]

Another term we use in this chapter is the *subgraph* of a graph. A subgraph of a graph is obtained by selecting any subset of vertices of the graph, and selecting any subset of edges of the graph that are attached to the selected vertices. In chemistry, subgraphs are molecular fragments. As depicted in Figure 4, the fragments of a molecular graph may or may not overlap.

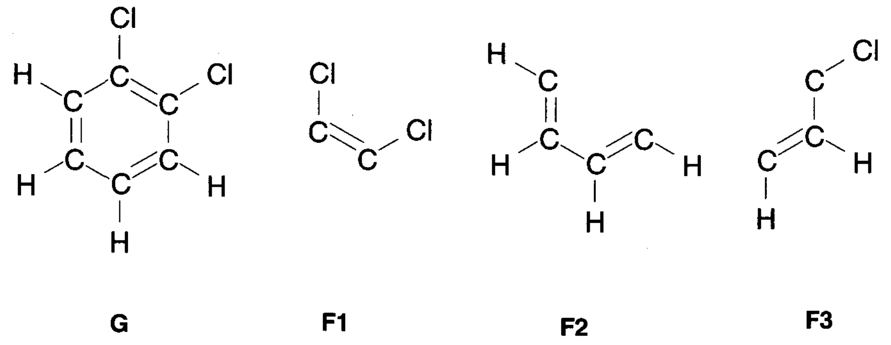


Figure 4. Fragments of a molecular graph G. F1 and F2 are non overlapping fragments, F3 overlap with both F1 and F2.

We finish this subsection with a few additional definitions. A molecular graph is a *tree* if it does not contain cycles. Multiple bonds are allowed in molecular trees, and alkanes and alkenes are examples of molecular trees. A rooted tree is a tree where one vertex (the root) is distinguished from the others. Isotopically mono-labeled alkanes are a rooted tree. Of course not all molecules are trees, but all molecules are *bonded valence* graphs. Bonded valence graphs are graphs for which the degree of any vertex is below some threshold. For most organic molecules the maximum valence of any atom is 4 (sometimes 5), and for any molecule the number of bonds attached to any atom is always limited due to the three dimensional space limitations surrounding an atom. Thus, molecular graphs are bonded valence graphs. This is an important property because bonded valence graphs can usually be treated more easily than general graphs. For instance, isomorphism can be solved efficiently for that class of graph.[11] The term *efficient* has a specific meaning here. An algorithm is said to be efficient if the time taken and the space allocated to complete the job is a polynomial of the size of the problem. With a molecular graph the size of the problem is usually the number of atoms. An example of an efficient algorithm is Kudo-Sasaki's quadratic $O(n^2)$ time and space algorithm that computes the connectivity stack from an adjacency matrix.[8] Problems that cannot be solved by a polynomial time and space algorithm are said to be *intractable*. Searching all the occurrences of a fragment (subgraph) in a molecular graph is an intractable problem.[12]

Counting Structures: How many isomers has decane?

Counting labeled and unlabeled graphs

In this subsection, we briefly summarize results relevant to labeled graphs. We then survey the work on counting series by Cayley, Pólya, Harary, and Read. Particular attention is given to Pólya's work since it has lead to many applications in chemistry.

While labeled graphs are not the appropriate objects to describe molecular graphs, we recall that combinatorial libraries, protein and gene networks, and linear reaction networks can be represented by labeled graphs. Furthermore, some of the results given below will later be used to count unlabeled graphs and molecular graphs.

We first note that there are $\binom{n}{2} = n(n-1)/2$ possible distinct edges between *n* vertices, and there are $\binom{n(n-1)/2}{k}$ ways of choosing *k* edges in a set of $n(n-1)/2$ edges. Summing for all possible *k* values gives the number of labeled graphs of *n* vertices:

$$L_n = \sum_{k=0}^{n(n-1)/2} \binom{n(n-1)/2}{k} = 2^{n(n-1)/2} \qquad [1]$$

Since the objects we are interested in this chapter are connected, let C_k be the number of connected labeled graphs of *k* vertices. There are kC_k rooted connected labeled graphs since there are *k* ways of choosing a root. The number of rooted, labeled graphs of *n* vertices in which the root is in a connected component containing *k* vertices is $kC_k\binom{n}{k}L_{n-k}$. This expression, summed from $k=1$ to *n*, is equal to nL_n, which is the number of rooted labeled graphs. Thus, $nL_n = \sum_{k=1}^{n} kC_k\binom{n}{k}L_{n-k}$, from which we derive the recursive formula for counting the number of connected labeled graphs of *n* vertices:

$$C_n = 2^{n(n-1)/2} - \frac{1}{2}\sum_{k=0}^{n-1} k\binom{n}{k}2^{(n-k)(n-k-1)/2}C_k \qquad [2]$$

Interestingly enough, investigations related to counting unlabeled graphs started with the pragmatic problem of calculating the number of paraffin structures. Cayley was the first to propose a solution,[1] and doing so he introduced the notion of trees.[13] Applications of Cayley's counting formula are tedious and prone to errors. Cayley himself made several errors in his work, some of which were later corrected by Herrmann.[14] Almost 75 years after Cayley's initial work, Henze and Blair[15] proposed a recursive formula much easier to apply than the Cayley formulation. The next significant advance came with the work of Pólya[16] and his famous theorem. Most of today's counting techniques are making use of Pólya's theory and we shall first describe the theory prior to using it to count objects relevant to chemistry.

Pólya Theory of Counting. In 1935, Pólya proposed a counting theory that is probably the most powerful counting technique available to chemists. In fact, in his original paper Pólya already illustrated his theory by counting the number of structural isomers when hydrogen atoms in benzene are successively substituted with monovalent atoms or groups.[16] The theory relies on the concept of the cycle index, *Z(A),* where *A* is a permutation group with object set $X = \{1,2,\ldots,n\}$, and Z stands for the German word *Zyklenzeiger* (meaning cycle index). Applied to a graph, *X* is the set of vertices, and *A* is the automorphism group of the graph as defined in subsection "From Graph Theory to Chemistry". Keeping in mind that the automorphism group of a graph takes into account the permutations, or symmetry operations, namely the proper rotation axes of the structure, the cycle index of a given permutation, α, is obtained by decomposing α into disjoint cycles, formally,

$$Z(\alpha; s_1, s_2, \ldots, s_n) = \prod_{k=1}^{n} s_k^{j_k(\alpha)} \qquad [3]$$

where s_k is a variable representing cycles of length *k*, and $j_k(\alpha)$ is the number of cycles s_k in α. To illustrate cycle decomposition, let $\alpha = \sigma_v^{(1)}$ be the reflection of benzene perpendicular to the axis going through atoms 1 and 4. As depicted in Figure 3, $\alpha =$ (1) (4) (2 6) (3 5), is composed of 2 cycles of length 1: (1) and (4), and 2 cycles of length 2: (2 6) and (3 5). Its cycle index is: $Z(\alpha) = s_1^2 s_2^2$. The cycle index of an automorphism group A is simply obtained by summing the cycle decompositions for all the permutations in the group and dividing by the size of the group:

$$Z(A; s_1, s_2, ..., s_n) = \frac{1}{|A|} \sum_{\alpha \in A} Z(\alpha; s_1, s_2, ..., s_n) = \frac{1}{|A|} \sum_{\alpha \in A} \prod_{k=1}^{n} s_k^{j_k(\alpha)} \quad [4]$$

From Figure 3 and Table 1 it is easy to verify that the cycle index of benzene is:

$$Z = \frac{1}{12}(s_1^6 + 4s_2^3 + 2s_3^2 + 2s_6^1 + 3s_1^2 s_2^2) \quad [5]$$

Table 1. Cycle index for benzene. Automorphism permutations are listed in Figure 3.

Symmetry operation	permutation	Cycle index
E	(1)(2)(3)(4)(5)(6)	s_1^6
C_6^-	(1 2 3 4 5 6)	s_6^1
C_6^+	(6 5 4 3 2 1)	s_6^1
C_3^-	(1 3 5)(2 4 6)	s_3^2
C_3^+	(5 3 1)(6 4 2)	s_3^2
C_2	(1 4)(2 5)(3 6)	s_2^3
$\sigma_v^{(1)}$	(1)(4)(2 6)(3 5)	$s_1^2 s_2^2$
$\sigma_v^{(2)}$	(3)(6)(1 5)(2 4)	$s_1^2 s_2^2$
$\sigma_v^{(3)}$	(2)(5)(1 3)(2 4)	$s_1^2 s_2^2$
$\sigma_v^{(4)}$	(1 6)(2 5)(3 4)	s_2^3
$\sigma_v^{(5)}$	(1 2)(3 6)(4 5)	s_2^3
$\sigma_v^{(6)}$	(1 4)(2 3)(5 6)	s_2^3

To introduce Pólya's theorem we take the example of counting the numbers of isomers obtained when substituting hydrogen atoms in benzene with chlorine atoms. Pólya's theorem applied to this problem states that the number of isomers, when k hydrogen atoms are substituted by k chlorine atoms, is the coefficient C_k of the generating function, $C(x)$, obtained by substituting in the cycle index each variable s_k by $(1+x^k)$. While the proof of this can be found in Pólya's paper,[16] or more recent sources such as the book of Harary and Palmer, [17] intuitively the substitution comes from the fact that each hydrogen atom can either be replaced, or not, by a chlorine atom. These 2 possibilities (0 or 1) are expressed by the function $x^0+x^1 = 1+x$, which Pólya calls the *figure* generating function. Now, observing that there are more ways of coloring an object with no symmetry than ways of coloring an object where all vertices are symmetrical, one realizes that the automorphism group of the studied object has to play a role in counting the number of configurations. The exact relationship between the number of configurations and the automorphism group is given by Pólya's theorem.

Theorem (Pólya). The configuration generating function, or counting series, *C(x),* is obtained by substituting the figure generating function, *c(x),* in the cycle index, by replacing every occurrence of s_k in the cycle index by $c(x^k)$. Thus:

$$C(x) = Z(A; c(x), c(x^2), c(x^3), \ldots) \qquad [6]$$

A corollary of Pólya's theorem it that the total number of configurations, *N*, obtained after coloring an object with permutation group *A* with *n* colors is obtained by replacing every occurrence of s_k in the cycle index of *A* by *n*. Formally,

$$N(A; s_1, s_2, \ldots, s_n) = \frac{1}{|A|}\sum_{\alpha\in A}\prod_{k=1}^{n} n^{j_k(\alpha)} = \frac{1}{|A|}\sum_{\alpha\in A} n^{\sum_{k=1}^{n} j_k(\alpha)} = \frac{1}{|A|}\sum_{\alpha\in A} n^{|o(\alpha)|} \qquad [7]$$

where $|o(\alpha)|$ is the number of orbits of the permutation α. For a graph, $|o(\alpha)|$, is the number of vertex equivalent classes induced by α.

As an illustration, the generating function $c(x) = 1+x$ substituted in the benzene cycle index of eq. 5 gives the counting series:

$C(x) = 1/12\ [(1+x)^6 + 4(1+x^2)^3+2(1+x^6)+2(1+x)^2(1+x^2)^2]$

$$= 1 + x + 3x^2 + 3x^3 + 3x^4 + x^5 + x^6 \qquad [8]$$

and the total number of configurations is $1/12\,[(2)^6 + 4(2)^3+2(2)+2(2)^2(2)^2] = 13$. The coefficients of eq. 8 represent the number of isomers of benzene (1), chlorobenzene (1), dichlorobenzene (3), trichlorobenzene (3), etc.., up to hexachlorobenzene (1). The various structural isomers counted in eq. 8 are listed in Figure 5.

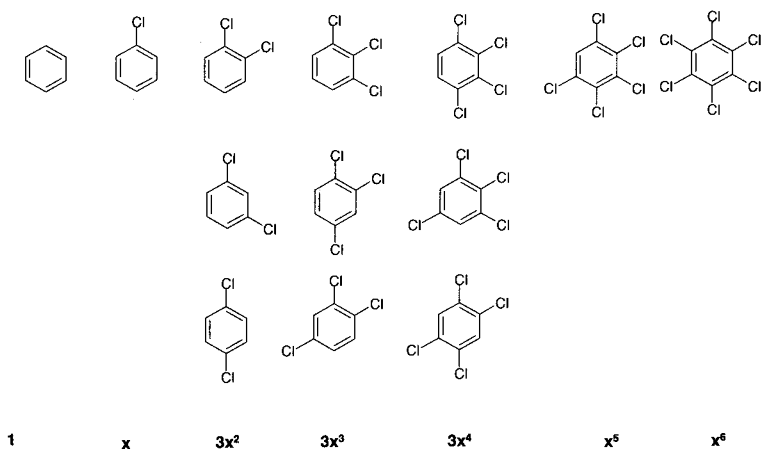


Figure 5. Count of k-chloro-benzene isomers using Pólya's theorem.

Counting molecules

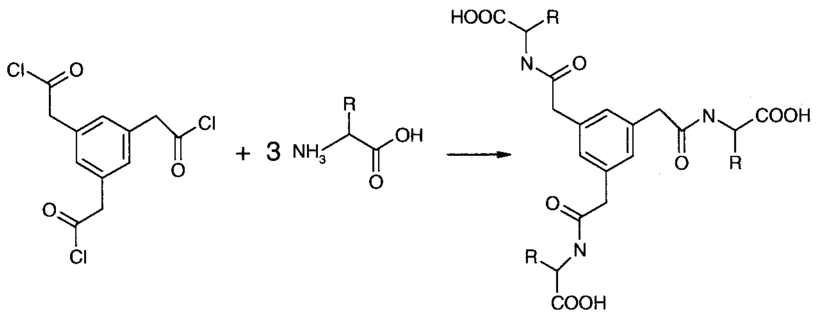
While we have already seen examples on how Pólya's theorem can be used to enumerate chemical compounds, we consider this idea in greater detail in this subsection. The most general problem of this kind is to determine the number of isomers given a molecular formula. While this problem can be solved using explicit enumeration techniques (cf. "Enumerating Structures" subsection), currently there are no counting series that provides the number of isomers of a given molecular formula. However, if we lower our expectations and confine our attention to some restricted class of compounds, a mathematical treatment of the problem then becomes possible.

The most straightforward use of Pólya's theorem is with substituted or labeled hydrocarbons. Indeed, we have already seen that the count of structural isomers obtained after substituting hydrogen atoms in benzene with chlorine atoms is derived directly by plugging the figure generating function $c(x) = 1+x$, into the cycle index of benzene. In Table 2 the same exercise is carried out for other benzenoid hydrocarbons, and the number of isomers obtained after substituting k hydrogen atoms is the x^k coefficient of the corresponding counting series. This type of calculation can also be performed to count substituted fullerenes,[18-20] polyhedral cages,[20] and substituted cycloalkanes.[21] A general approach to counting substituted isomers based on their symmetries can be found in Baraldi and Vanossi.[22]

Table 2. Cycle indices and counting series for some substituted benzenoids and hydrocarbons cages

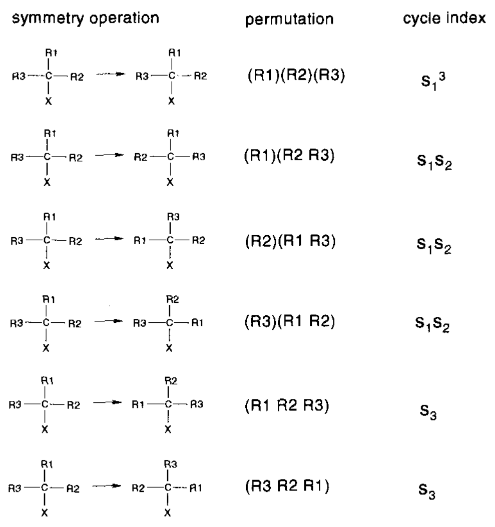
Benzenoids and Cages	Symmetry group	Cycle index	Counting series
Benzene	D_{6h}	$^1/_{12}\,(s_1^6 + 4s_2^3 + 2s_3^2 + 2s_6^1 + 3s_1^2s_2^2)$	$1 + x + 3x^2 + 3x^3 + 3x^4 + x^5 + x^6$
Naphthalene	D_{2h}	$^1/_4\,(s_1^8 + 3s_2^4)$	$1 + 2x + 10x^2 + 14x^3 + 22x^4 + 14x^5 + 10x^6 + 2x^7 + x^8$
Anthracene	D_{2h}	$^1/_{12}\,(s_1^{10} + s_1^2 s_2^4 + 2s_2^5)$	$1 + 3x + 15x^2 + 32x^3 + 60x^4 + 66x^5 + 60x^6 + 32x^7 + 15x^8 + 3x^9 + x^{10}$
Phenanthrene	C_{2v}	$^1/_2\,(s_1^{10} + s_2^5)$	$1 + 5x + 25x^2 + 60x^3 + 110x^4 + 126x^5 + 110x^6 + 60x^7 + 25x^8 + 5x^9 + x^{10}$
Tetracene	D_{2h}	$^1/_4\,(s_1^{12} + 2s_2^6)$	$1 + 3x + 21x^2 + 55x^3 + 135x^4 + 198x^5 + 236x^6 + 198x^7 + 125x^8 + 55x^9 + 21x^{10} + 3x^{11} + x^{12}$
Triphenylene	D_{3h}	$^1/_6\,(s_1^2 + 2s_2^6 + 2s_3^4)$	$1 + 2x + 14x^2 + 38x^3 + 90x^4 + 132x^5 + 166x^6 + 132x^7 + 90x^8 + 38x^9 + 14x^{10} + 2x^{11} + x^{12}$
C_{60}	I_h	$^1/_{120}\,(24s_{10}^6 + 20\,s_6^{10} + 24\,s_5^{12} + 20\,s_3^{20} + 16\,s_2^{30} + 15\,s_1^4\,s_2^{28} + s_1^{60})$	$1 + x + 23x^2 + 303x^3 + 4190x^4 + 45718x^5 + 418\,470x^6 + 3\,220\,218x^7 + 21\,330\,558\,x^8 + 123\,204\,921x^9 + 628\,330\,629x^{10} + \ldots$

Another direct application of Pólya's theorem is to compute the sizes of combinatorial libraries that can be generated from a scaffold and a set of reactants. We recall that the total number of configurations obtained after coloring an object with *k* colors is obtained by substituting *k* in the cycle index of the object. Library sizes are computed by taking the scaffold as the object and the reactant as the colors. As an example, consider the following reaction scheme:



There are three reacting sites on the benzene ring, the cycle index of benzene reduced to these three sites is $^1/_6(s_1^3+3s_1s_2+2s_3)$ (cf. Figure 3 for a list of the permutations involved). According to eq. 7, attaching *n* different R reactants to tri-acidcloride will result in a library of size $^1/_6(n^3+3n^2+2n)$. Now, $n = 2$ different reactants will give 4 compounds, and if the reactants are all the possible ($n = 20$) amino acids, the library will be composed of 1540 compounds. Scaffolds in library design often have no symmetry, and if such a scaffold is composed of *r* reacting site, its cycle index is s_1^r. The size of a library composed of *n* reactants for a scaffold with no symmetry and *r* reacting sites is n^r.

A bit more challenging is the use of Pólya's theorem to count alkyl groups. An alkyl group has the formula $-C_nH_{2n+1}$, and contains one free bond or bonding site. An alkyl group is a rooted tree because the carbon atom carrying the bonding site can be distinguished from the other. Let $A_n(x)$ be the counting series for alkyl groups having *n* atoms. The remarkable idea in counting alkyl groups using Polya's theorem is to use a figure generating function that is the counting series itself. It is thus a recursive process where the number of alkyl group of n atoms is counted from the number of alkyl groups of n-1 atoms. To apply Pólya's theorem we must first determine the group of permutations attached to atom number n. The permutations attached to any carbon atom in an alkyl chain are listed in Figure 6.



symmetry operation	permutation	cycle index
	(R1)(R2)(R3)	s_1^3
	(R1)(R2 R3)	s_1s_2
	(R2)(R1 R3)	s_1s_2
	(R3)(R1 R2)	s_1s_2
	(R1 R2 R3)	s_3
	(R3 R2 R1)	s_3

Figure 6. Permutation group and cycle index for carbon atoms in alkyl groups.

Clearly, all alkyl groups attached to a carbon atom are interchangeable. The permutation group is called the *symmetric* group S_3 with cycle index $^1/_6$ $(s_1^3 + 3s_1s_2 + 2s_3)$. Substituting $A_{n-1}(x)$ into the cycle index of S_3 gives a counting series representing the number of ways of attaching three alkyl groups to an additional atom. Multiplying the resulting series by x, that is, adding the additional atom number n, leads to the following counting series for alkyl groups. Starting with $A_0(x) = 1$:

$$A_n(x) = 1 + \frac{1}{6}x[A_{n-1}{}^3(x) + 3A_{n-1}(x)A_{n-1}(x^2) + 2A_{n-1}(x^3)] \qquad [9]$$

The 1 on the right-hand side must be added to ensure that the term A_0 corresponding to a hydrogen is properly counted. In the above expression the coefficient of $A_n(x)$ has to be computed only up to x^n. To avoid this restriction, it is customary to write eq. 9 up to n = ∞:

$$A(x) = \sum_{n=0}^{\infty} A_n x^n = 1 + \frac{1}{6} x[A^3(x) + 3A(x)A(x^2) + 2A(x^3)] \qquad [10]$$

The basic operations in the above expression are summations and products of polynomials and a scalar multiplication. For polynomials of order n, summations and scalar multiplications are performed using no more than n integer arithmetic operations, while polynomial products necessitates at most $O(n^2)$ integer operations. The total cost of computing the counting series is therefore $O(n^3)$. The first elements of the series are $A(x) = 1 + x + x^2 + 2x^3 + 4x^4 + 8x^5 + 17x^6 + 39x^7 + 89x^8 + 211x^9 + 507x^{10} + \ldots$

The number of isomers of acyclic compounds

The coefficients $A_0, A_1, \ldots, A_n$ of eq. 10 can be used to evaluate the number of isomers for several families of acyclic compounds comprising up to n carbon atoms. Computationally, all these numbers can be obtained by summations, products, and scalar multiplications of the polynomial $A(x)$. The results that follow have been derived by Read.[23]

Primary alcohols. The primary alcohols are of the form R-CH_2-OH where R is an alkyl group with n-1 carbons atoms. To maintain the correct number of carbon atoms, the counting series for primary alcohols becomes:

$$xA(x) \qquad [11]$$

Secondary alcohols. The secondary alcohols are of the form R_1-CH(R_2)-OH, where R_1 and R_2 are alkyl groups. To count these isomers we apply Pólya's theorem with the figure counting series $A(x)$-1 because R_1 and R_2 are not hydrogen atoms. The permutation group is the symmetric group S_2 because R_1 and R_2 are interchangeable, and the counting series for secondary alcohols is:

$$xZ(S_2; A(x)-1) = 1/2x[A^2(x)-2A(x)+A(x^2)] \qquad [12]$$

Tertiary alcohols. The formula for a tertiary alcohol is OH-C(R_1)(R_2)(R_3). The counting series is obtained using the same arguments as for secondary alcohols but with the permutation group S_3 instead of S_2:

$$xZ(S_3; A(x)-1) = 1/6x[A^3(x)-3A^2(x)+3A(x)A(x^2)-3A(x^2)+2A(x^3)] \qquad [13]$$

Aldehydes and ketones. These compounds have the form R_1-C=O-R_2, where R_1 and R_2 are alkyl groups and, possibly, hydrogen atoms. Since hydrogen atoms are included, the counting series is:

$$xZ(S_2; A(x)) = 1/2x[A^2(x)+A(x^2)] \qquad [14]$$

Alkynes. The formula for acetylene compounds takes the form R_1-C≡C-R_2 and because there are two additional carbon atoms when no terminal hydrogens exists, the counting series is:

$$xZ(S_2; A(x)) = 1/2x^2[A^2(x)+A(x^2)] \qquad [15]$$

Esters. The general ester formula is R_1-C=O-OR_2, with R_1 and R_2 being alkyl groups. R_1 can be a hydrogen atom but not R_2 otherwise the compound would be an acid. Consequently, the counting series are $A(x)$ for R_1 and $A(x)$-1 for R_2. One more carbon must be added in the ester formula. The final counting series becomes:

$$xA(x)[A(x)-1] \qquad [16]$$

Isotopically labeled alkanes. This is the class of alkanes where one carbon atom has been labeled with, for instance, a C-13 isotope. The general formula for these compounds is C(R_1)(R_2)(R_3)(R_4), where C is the labeled carbon atom and R_i, i=1,…,4 are alkyl groups whose counting series is $A(x)$. Since all alkyl groups can be exchanged with one another around the labeled carbon atom, the permutation group is the symmetric group S_4 with cycle index $1/24(s_1^4+6s_1^2s_2+3s_2^2+8s_1s_3+6s_4)$ and the counting series for labeled alkanes is:

$$\begin{aligned} P(x) &= \sum_{n=1}^{\infty} P_n x^n = xZ(S_4; A(x)) \\ &= \frac{1}{24}x[A^4(x)+6A^2(x)A(x^2)+3A^2(x^2)+8A(x)A(x^3)+6A(x^4)] \end{aligned} \qquad [17]$$

We now turn our attention to a class of compounds that is not of primary importance in chemistry but the results are used later to derive the counting series for alkanes. These structures are of the type R_1-R_2 where R_1 and R_2 are non hydrogen alkyl groups with counting series $A(x)$-1. The permutation group is S_2 and the counting series is:

$$Q(x) = \sum_{n=1}^{\infty} Q_n x^n = Z(S_2; A(x)-1) = \frac{1}{2}[(A(x)-1)^2 + A(x^2)-1] \qquad [18]$$

Alkanes. One may think that counting alkanes would be less difficult than counting alcohols, ketones, esters, and other substituted or labeled structures especially since these compounds are all derived from alkanes, but this is not the case. Actually, Cayley, Henze and Blair, and even Pólya had a great deal of difficulty finding the alkane counting series. Their solutions are rather complex involving tree centers and bicenters. For instance, in the case of Cayley the solution for alkanes was developed in 1875,[1] 18 years after finding the counting series for rooted trees.[13] It was only in 1948 that a simple formula for unlabeled trees was found by Otter.[24] The solution we review next was first given by Read.[23] and is an application of Otter's formula to alkanes.

Let us first consider an arbitrary unlabeled alkane. We want to find p^*, the number of different atom-labeled alkanes obtained after labeling all the carbon atoms one after another. Two carbon atoms once labeled will produced the same labeled structure if they are symmetrical. Thus, p^* is the number of equivalent classes among atoms, formally the number of orbits in the automorphism group as defined in the subsection "From Graph Theory to Chemistry". Using the same arguments, we find that q^*, the number of bond-labeled alkanes, is the number of bond's equivalent classes. Otter[24] and also Harary and Norman[17] have shown that for any unlabeled tree, $p^* - q^* + s = 1$, where $s = 1$ if the tree has a symmetric bond (e.g., a bond between two identical subtrees) and $s = 0$ otherwise. Now, if we sum the previous equation over all alkanes having n carbon atoms, we obtain $P_n - Q_n + \Sigma\, s = a_n$, where $P_n = \Sigma\, p^*$ and $Q_n = \Sigma\, q^*$ and a_n is the number of alkanes having n carbon atoms. Clearly P_n is the number of atom-labeled alkanes having n carbon atoms and is thus the n^{th} coefficient of the counting series in eq. 17. Similarly, Q_n is the n^{th} coefficient in eq. 18. In order to compute a_n we have to evaluate s. As already mentioned, s =1 for those alkanes having a bond splitting the structure into two identical alkyl groups. These alkanes must therefore have an even number n of carbon atoms and their count is simply equal to the number $A_{n/2}$ of alkyl groups having $n/2$ carbon atoms. The number of alkanes having n carbon atoms is thus $a_n = P_n - Q_n + A_{n/2}$, with $A_{n/2} = 0$ when n is odd. The corresponding counting series is obtained by multiplying a_n by x^n and summing starting with $n = 1$:

$$a(x) = \sum_{n=1}^{\infty} a_n x^n = P(x) - Q(x) + A(x^2) - 1$$

$$a(x) = \frac{1}{24} x[A^4(x) + 6A^2(x)A(x^2) + 3A^2(x^2) + 8A(x)A(x^3) + 6A(x^4)] \qquad [19]$$

$$-\frac{1}{2}[(A(x)-1)^2 - A(x^2) + 1]$$

The first elements of the series are $A(x) = 1 + x + x^2 + x^3 + 2x^4 + 3x^5 + 5x^6 + 9x^7 + 18x^8 + 35x^9 + 75x^{10} + \ldots$, and to answer the subsection's question, there are 75 structural isomers for decane. Using eq. 19, the number of alkane isomers up to 25 carbon atoms are given in Table 4 in the "Chemical Information" subsection appearing later in the chapter. Note that *a(x)* can be evaluated computationally, using the product, sum, and scalar multiplication operator on the polynomial $A(x)$ representing the alkyl group counting series. Considering the computational cost to evaluate $A(x)$, alkanes up to n carbon atoms can be counted using no more than n^3 elementary arithmetic operations.

Hydroxyl ethers. In a recent development, Wang, Li and Wang[25] proposed a counting series for compounds of the form $C_iH_{2i+2}O_j$. This is a first step toward a general counting series for molecular formulae. Their technique uses Pólya's cycle index and two generating functions for alkyl groups R(I) where the root is a carbon atom, and alkoxyl groups R(II) are rooted on an oxygen atom.

The number of stereoisomers of acyclic compounds

Stereoisomers of acyclic compounds are derived the same way as structural isomers, but the permutation group used in Pólya's cycle index is no longer the symmetric group S_3. While with structural isomers the three alkyl groups attached to any carbon atom are interchangeable, with stereoisomers, the alkyl groups can be arranged in two distinct enantiomeric forms, rectus (R) and sinister (S). Consequently, in the permutation group attached to carbon atoms, all permutations mapping an R form onto an S form must be discarded. The remaining permutations are listed in Figure 7, and the permutation group is the cyclic group C_3 with cycle index $Z(C_3) = 1/3\ [s_1^3+2s_3]$.

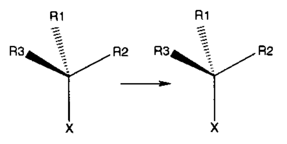
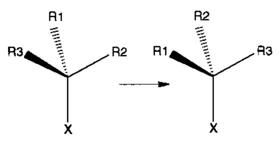
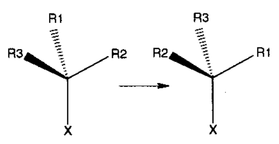
symmetry operation	permutation	cycle index
	(R1)(R2)(R3)	s_1^3
	(R1 R2 R3)	s_3
	(R3 R2 R1)	s_3

Figure 7. Permutation group and cycle index for stereo carbon atoms in alkyl groups. All permutations maintain the R or S stereocenter.

Now that we have determined the permutation group we can count stereoisomers using the formulae obtained with structural isomers but by replacing the group S_3 by C_3. For instance, from eq. 10, the counting series for alkyl groups becomes:

$$A'(x) = 1 + x\,Z(C_3;A'(x)) = 1 + 1/3x[A'^3(x)+2A'(x^3)] \qquad [20]$$

The counting series for functionalized stereoalkanes are summarized in the following table. All the results have been derived by Read.[23] The first elements of the counting series for stereoalkanes are $a'(x) = 1 + x + x^2 + x^3 + 2x^4 + 3x^5 + 5x^6 + 11x^7 + 24x^8 + 55x^9 + 136x^{10} + \ldots$, and decane thus has 136 stereoisomers.

Table 3. Counting series for the stereoisomers of functionalized alkanes

compound	formula	Counting series
alkyl groups	-R	$A'(x) = 1 + {}^1/_3x[A'^3(x)+2A'(x^3)]$
Primary alcohols	$R-CH_2-OH$	$xA'(x)$
Secondary alcohols	$R_1-CH(R_2)-OH$	$x[A'(x)-1]^2$
Tertiary alcohols	$HO-C(R_1)(R_2)(R_3)$	${}^1/_3x\{ [A'(x)-1]^3+2[A'(x^3)-1] \}$
Aldehydes, ketones	$R_1-C=O-R_2$	${}^1/_2\, x[A'^2(x)+A'(x^2)]$
Alkynes	$R_1-C\equiv C-R_2$	${}^1/_2\, x^2[A'^2(x)+A'(x^2)]$
Esters	$R_1-C=O-OR_2$	$A'(x)[A'(x)-1]$
Stereoalkanes	C_nH_{2n+2}	${}^1/_{12}x[A'^4(x)+3A'^2(x^2)+8A'(x)A'(x^3)]$ $-{}^1/_2[(A'(x)-1)^2-A'(x^2)+1]$

All the isomer counts we have given so far are derived from Pólya's theorem and the alkyl group counting series. Our intention was to illustrate the power of Pólya's counting theory and also to make things easier to follow since all formulae are derived using the same technique. The reader interested in further details on the applications of Pólya's theory to chiral and achiral compounds and to reaction processes is referred to the book of Fujita.[26] It is also worth noticing that Pólya's theory has also been applied to count staggered conformers of alkanes and monocyclic cycloalkanes.[27] Staggered conformers of alkanes are represented by systems which can be embedded in the diamond lattice. Beyond Pólya, few other methods have been proposed in the literature to count acyclic hydrocarbons. In particular, in a series of papers, Yeh gives counting series for alkanes,[28] polyenoids,[29] alkenes[30] and structures excluding steric strain[31, 32] based on Cayley's counting series. Bytautas and Klein[33] have more recently derived a new alkane counting series using graph's diameter instead of Otter's formula.

The number of benzenoids and polyhex hydrocarbons

This particular class of hydrocarbons has lead to numerous investigations and probably deserves an entire chapter to be properly reviewed. Here we summarize only the major findings related to counting. The reader further interested by polyhexes and benzenoids can consult the books of Gutman and Cyvin [34-37] as well as the books of Dias.[38, 39] These books, as well as that by Trinajsticc,[40] provide valuable information regarding the counting and enumeration of Kekulé structures and the conjugated-circuit model, neither of which is reviewed here due to space limitations.

As illustrated in Figure 8, a *polyhex* is a connected system of congruent regular hexagons such that two hexagons either share exactly one edge or are disjoint. Among polyhex hydrocarbons are *helicenes* such as heptahelicene, which are non planar, and *coronoids*, such as cyclodecakisbenzene, which are systems with holes. The most heavily studied class of polyhexes has been, by far, *benzenoid* hydrocarbons, which are planar and simply connected. In other words, benzenoid hydrocarbons are condensed polycyclic unsaturated fully conjugated hydrocarbons composed of six-membered rings. The class of benzenoid hydrocarbon is further divided into two subsets: *catacondensed* and *pericondensed.* Catacondensed benzenoids, such as phenanthrene, are systems where all carbon atoms are lying on the perimeter of the structure.

Pericondensed benzenoid are structure having $n_i \neq 0$ internal atoms, e.g. atoms that do not belong to the perimeter. Phenalene ($n_i = 1$) and pyrene ($n_i = 2$) are examples of pericondensed benzenoids. Finally, all polyhexes are either *Kekuléan* (cf. pyrene) or *non-Kekuléan* (cf. phenalene) depending on whether or not they possess Kekulé structures. While we are discussing nomenclature it is worth outlining the distinctions between benzenoid hydrocarbons and polycyclic aromatic hydrocarbons (PAHs). PAHs possess features that are not shared with benzenoid hydrocarbons; they may contain rings with sizes different from six, they may also comprise sp3 carbons atoms, and side groups.

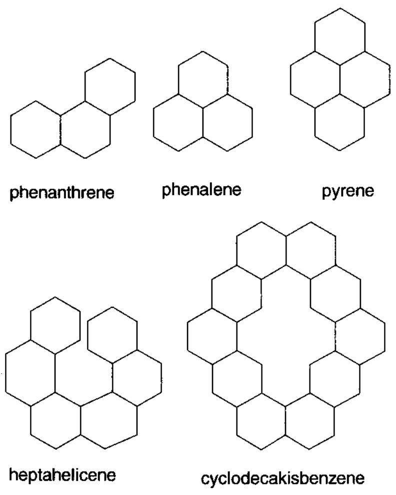


Figure 8. Some polyhex hydrocarbons.

There are essentially two types of approaches to count polyhexes. One is to make use of a counting series and Pólya's theorem while the other is an algorithmic approach based on explicit enumeration. The algorithmic approach is reviewed in the "Enumerating Structures" subsection as counting is performed through enumeration and each solution is actually generated. It is nonetheless worth mentioning that the algorithmic approach can be used to count planar benzenoid systems while the former approach cannot, as helicenes are included in the counting series. Additionally, there are further limitations with counting series. Polyhex hydrocarbons that cannot be represented by tree-like structures, such as, for instance pericondensed benzenoids with many internal atoms cannot be counted.

The first serious attempts to count polyhexes are due to Balaban and Harary[41] and Harary and Read.[42] While Balaban and Harary proposed a nomenclature and simple counting formulae for some benzenoid systems, Harary and Read derived the first counting series for catacondensed polyhexes. The catacondensed systems counted by Harary and Read include helicenes. These are also named catafusenes and, strictly speaking, are not benzenoids since they can be non-planar.

To count catafusenes like with alkyl groups and alkanes, we first derive a counting formula for bond-rooted catafusenes. A bond-rooted catafusene is a catafusene where one periferal bond (the root) has been labeled. We can distinguish two kinds of bond-rooted catafusenes according to whether one or two hexagons are attached to the hexagon containing the root bond (cf. Figure 9). Note that these are the only possibilities if perifusenes are to be avoided. We call them *S*-catafusenes and *D*-catafusenes, respectively.

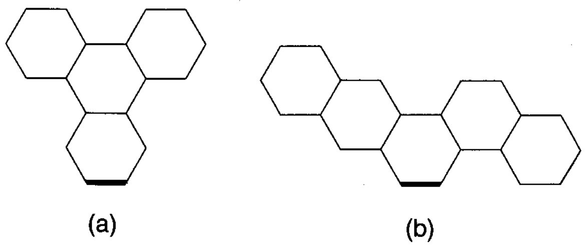


Figure 9. Bond-rooted catafusenes. (a) *S*-catafusene. Only one hexagon adjoins the hexagon with the root bond (thick line). (b) *D*-catafusene. Two hexagons adjoin the hexagon with the root bond.

Let S_n and D_n denote the numbers of S-catafusenes and D-catafusenes having n hexagons, and let $U_n = S_n + D_n$ be the total of bond-rooted catafusenes with n hexagons. From Figure 9 it is easy to be convinced that:

$$S_n = 3U_n \qquad D_{n+1} = \sum_{k=1}^{n-1} U_k U_{n-k} \qquad [21]$$

We now define the three generating functions $S(x) = \sum_{i=1}^{\infty} S_i x^i$, $D(x) = \sum_{i=1}^{\infty} D_i x^i$ and

$U(x) = \sum_{i=1}^{\infty} U_i x^i$. Since $U_n = S_n + D_n$ we have:

$$U(x) = S(x)+D(x)+x \qquad [22]$$

The x on the right hand side come from the fact that $U_1 = 1$ while $S_1 = D_1 = 0$. Substituting eq. 21 into eq. 22, we derive the counting series for bond-rooted catafusenes hydrocarbons:

$$U(x) = 3xU(x)+xU^2(x)+x \qquad [23]$$

We now wish to count catafusenes in which one hexagon (the root) has been distinguished from the other. Such a rooted catafusene is obtained by taking the root hexagon and attaching one, two, or three of its bonds to a bond-rooted catafusene. As depicted in Figure 10, there are four ways this can be done.

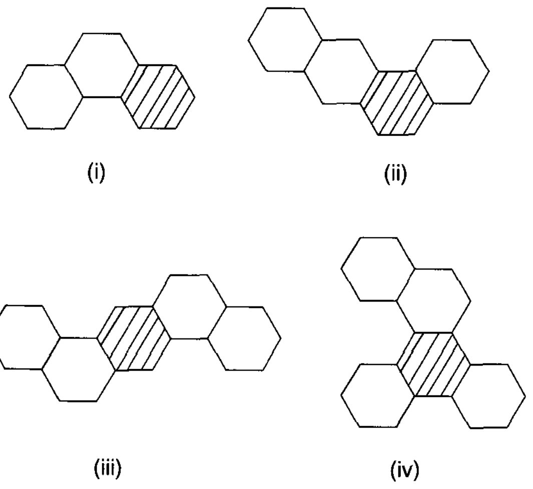


Figure 10. The four types of rooted catafusenes. The root hexagon is the shaded one. (i) only one bond-rooted catafusene is attached, (ii) two bond-rooted catafusenes are attached in "meta" position, (iii) two bond-rooted catafusenes are attached in "para" position, (iv) three bond-rooted catafusenes are attached.

Using the four cases depicted in Figure 10, the number of rooted catafusene of type (i) having n+1 hexagons is the number U_n of bond-rooted catafusenes, and the counting series for type (i) rooted catafusenes is $xU(x)$. Note that in order to count the root hexagon in the counting series one has to multiply $U(x)$ by x. To count rooted catafusenes of type (ii) comprising n+1 hexagons, we have to choose two bond-rooted catafusenes having, respectively, k and n-k hexagons. This procedure is similar to the calculation of D_{n+1} in eq. 21. Thus, the counting series for rooted catafusenes of type (ii) is: $xU^2(x)$. With the rooted catafusenes of type (iii) we have the possibility of building catafusenes, which are invariant under a rotation of 180°, such as in Figure 10 (iii). The permutation group attached to the root hexagon in case (iii) is the symmetric group S_2, and applying Pólya's theorem one finds the counting series for type (iii) rooted catafusenes to be: $xZ(S_2,U(x)) = x/2[U^2(x)+U(x^2)]$. Finally, to count rooted catafusenes of type (iv) one first observes that this time we have a possibility of symmetry under rotations of 120°. The permutation group is therefore the cyclic group C_3 (already encountered in when counting stereoisomers). Using Pólya's theorem, the counting series for type (iv) rooted catafusenes is:

$xZ(C_3,U(x)) = x/3[U^3(x)+2U(x^3)]$. Summing all the terms corresponding to cases (i) through (iv), the counting series for rooted catafusenes becomes:

$$F(x) = x + xU(x) + \frac{3}{2}xU^2(x) + \frac{1}{2}xU(x^2) + \frac{1}{3}xU^3(x) + \frac{2}{3}xU(x^3) \qquad [24]$$

The derivation of the counting series for unlabeled catafusenes can be found in Harary and Read.[42] Their solution makes use of Otter's formula[24] the same way the counting series for alkanes was derived using counting series for labeled alkanes and alkyl group. The counting series for unlabeled catafusenes is:

$$\begin{aligned} H(x) &= F(x) - \frac{1}{2}[U^2(x) - U(x^2)] \\ H(x) &= x + xU(x) + \frac{1}{2}(3x-1)U^2(x) + \frac{1}{2}(1+x)U(x^2) + \frac{1}{3}xU^3(x) + \frac{2}{3}xU(x^3) \end{aligned} \qquad [25]$$

So far we have regarded a catafusene and its mirror image as distinct, provided that the catafusene has no symmetry that would allow it to be rotated into its mirror image. The counting series in eq. 25 was corrected by Harary and Read[42] to count only once catafusenes and their mirror images. The series is:

$$h(x) = \frac{1}{12}(1+9x) - \frac{1}{12}(1-x)(1-5x)U(x) + \frac{1}{4}(3+5x)U(x^2) + \frac{1}{3}xU(x^3) \qquad [26]$$

The first terms of this counting series are: $h(x) = x + x^2 + 2x^3 + 5x^4 + 12x^5 + 37x^6 + 123x^7 + 446x^8 + 1689x^9 + 6693x^{10} + \ldots$

Other counting series have been developed, expanding on the initial work of Harary, Balaban, and Read. Harary-Read numbers have been classified and deconvoluted according to symmetries.[43, 44] Counting series have been developed for Fluorantenoids and Fluorenoids,[45] annelated catafusenes,[45] catacondensed monohiptafusenes,[45] and catacondensed octagonal systems.[45] Cyvin *et al.* have developed a combinatorial summation method that does not invoke counting series and explicit reference to Pólya's theorem. The method has been used to count perifusenes with one[46] and two internal vertices.[47]

Finally, we should mention the work on conjugated polyene hydrocarbons, which are not polyhexes, but have been counted[45] using a treatment similar to the one we just described for catafusene. The counting series for polyene hydrocarbons is:

$$p(x) = \frac{1}{12}[4U(x^3) + (6 + \frac{9}{x})U(x^2) + \frac{4}{x}U(x) - \frac{1}{x^2}U(x)] \qquad [27]$$

where $U(x)$ is the number of bond-rooted polyenes with a counting series similar to eq. 23:

$$U(x) = 2xU(x)+xU^2(x)+x \qquad [28]$$

The number of molecular cages (fullerenes and nanotubes)

To the best of our knowledge, isomers for fullerenes, nanotubes, spheroalkanes, and other molecular cages have so far been counted only through explicit enumeration (cf. "Enumerating Structures" subsection). In other words, we are not aware of any formula, counting series, or applications of Pólya's theorem from which one could compute the number of isomers for these compounds. In fact, molecular cages present a challenge for Pólya's theory of counting. Looking back, all compounds we have treated so far are either acyclic or have acyclic representations (cf. Balaban and Haray's paper[41] to see how catafusenes can be represented by trees). While a solution to enumerate general graphs, including cyclic graphs, using Pólya's theorem appeared in 1955,[17] difficulties arise with the class locally restricted graphs.[23] A locally restricted graph is a graph where the degrees of its vertices are predefined. Molecular cages are regular graphs where all atoms have the same degree (for instance three for fullerenes), they thus belong the class of locally restricted graphs.

In conclusion, we have seen how Pólya's theory of counting is a powerful and efficient tool to count chemical objects. All the counting series derived in this review can be computed using no more than $O(n^3)$ elementary arithmetic operations for compounds comprising up to n carbon atoms or n hexagons. Yet, there are difficulties deriving counting series for locally restricted graphs, especially if these graphs cannot be represented by trees. A substantial number of chemical compounds unfortunately belong to that difficult class of graphs. Each atom in a molecular graph has a specific degree given by the valence of the atom. Thus, molecules are *always* locally restricted graphs, and unless they have acyclic representations, molecules cannot easily be dealt using counting series. To overcome these difficulties an alternative is to use the explicit enumerations. We review this next.

Enumerating Structures: Are there any isomers of decane having seven methyl groups?

Enumerating labeled and unlabeled graphs

We begin with the enumeration of labeled graphs because, as with counting, they are easier to deal with. The algorithm we outline next for enumerating labeled graphs will later be used and modified to enumerate unlabeled graphs.

Our goal here is to enumerate all possible graphs that can be constructed with a set of vertices labeled 1 though n. The algorithm given in Scheme I is recursive. At each step of the recursion we augment the graph by one edge. We start with a graph containing no edges; this is our first labeled graph. Next, we add one edge between any pair of vertices $[i,j]$, $1 \leq i \leq n, j > i$. Clearly there are $n(n\text{-}1)/2$ of such edges. Each of $n(n\text{-}1)/2$ possibilities is a different labeled graph containing one edge. For each of these graphs a second edge is then added in all possible ways. To avoid generating the same labeled graph, the second edge $[k,l]$ must be lexicographically greater than the first, i.e., $[k,l] > [i,j]$ ($k > i$ or $k = i$ and $l > j$). To be convinced the requirement is necessary, consider the graphs U_1 and V_1 having, respectively, [1,2] and [3,4] as the first edge. Without lexicographic ordering one can add edge [3,4] to U_1 and edge [1,2] to V_1. The two resulting graphs are identical, both being composed of edges [1,2] and [3,4]. Now, the lexicographic requirement is sufficient since the edges of any labeled graph can be sorted lexicographically. The process of adding edges is repeated until no more can be added, i.e., edge $[n\text{-}1,n]$ already belongs to the graph. Running the algorithm given in Scheme I without constraints, we generate $m = n(n-1)/2$ labeled $(n,1)$-graphs having n vertices and one edge, and $\binom{m}{2}$ $(n,2)$-graphs having two edges, which is the number of ways of selecting two edges in a set of m edges. In general, the algorithm produces $\binom{m}{q}$ (n,q)-graphs with q edges. Summing all the contributions the total number of labeled graphs is 2^m in agreement with eq. 1.

```
Scheme I: Label-Enumeration(G)
1.   IF graph G is completed
2.      PRINT G
3.   ELSE
4.      FOR all edge e lexicographically greater than the edges of G DO
5.          IF constraints are not violated for the graph G U e
6.             Label-Enumeration(G U e)
7.          FI
8.      DONE
9.   FI
```

The algorithm given in Scheme I can also be run using constraints (cf. step 5) such as degree sequence, specific ranges for the number of edges, number of connected components, and cycle sizes. Additionally, some edges between specific labels may be forbidden, and the presence or absence of specific subgraphs may also be imposed. The above algorithm has actually been used to count and enumerate gene regulatory networks matching gene expression profiles (i.e., mRNA concentrations).[48] The algorithm was run with two constraints: a list of forbidden edges compiled from the expression profiles, and a maximum degree (2 and 3). Scheme I can also be used to generate combinatorial libraries when the scaffold has no symmetry. In such a case, the number of edges is at most the number of reacting sites on the scaffold and the only edges authorized are between scaffold and reactants.

In order to use Scheme I to enumerate unlabeled graphs one needs to remove duplicates, i.e., isomorphic graphs. Of course, this can be done after generating all labeled graphs with n vertices, but this becomes quite lengthy (i.e., $2^{n(n-1)/2}$) even for modest n. A better strategy is to build unlabeled (n,q)-graphs from unlabeled $(n,q\text{-}1)$-graphs. This can be carried out by augmenting all unlabeled $(n,q\text{-}1)$-graphs by one edge. But again, one has to remove duplicates. Observing that $n(n\text{-}1)/2\text{-}(q\text{-}1)$ edges can augment any unlabeled $(n,q\text{-}1)$-graph, and letting $N_{n,q\text{-}1}$ be the number of $(n,q\text{-}1)$-graphs, one has to test isomorphism between $[n(n\text{-}1)/2\text{-}(q\text{-}1)\ ]^2 N_{n,q\text{-}1}{}^2$ pairs of graphs. The problem is that $N_{n,q}$ scales exponentially with n and q.[49] The ideal solution would be to augment each unlabeled $(n,q\text{-}1)$-graph by one edge without having to be concerned with isomorphism. Fortunately this is possible as Read[50] has shown that the canonical representation of any (n,q)-graph is an augmentation of the canonical representation of exactly one $(n,q\text{-}1)$ graph. Recall from the subsection "From Graph Theory to Chemistry" that the canonical representation of a graph is a unique ordering of its vertices, such as the one for instance that maximizes its connectivity stack. Using Read's results, Scheme I can easily be modified to produced unlabeled graphs. The modified algorithm in given in Scheme II and is named *orderly generation.*

```
Scheme II: Orderly-Generation-Read-Faradzev(G)
1.   IF graph G is completed
2.      PRINT G
3.   ELSE
4.       FOR all edge e lexicographically greater than the edges of G DO
5.          IF constraints are not violated for the graph G U e
6.          AND CANON(G U e) = G U e
7.              Orderly-Generation-Read-Faradzev(G U e)
8.          FI
9.      DONE
10.  FI
```

The orderly algorithm is to enumeration what Pólya's theorem is to counting. Orderly generation is generally attributed to Read,[50] although Faradzev[51] independently published an orderly technique. Both Read and Faradzev use the fact that a graph is legitimate if it is identical to its canonical representation (cf. step 6, CANON(G U e) = G U e). To this end, an artificial ordering must be imposed on the set of graphs that are generated such that a canonical representative always contains a subgraph that is also canonical. A more general orderly algorithm proposed by McKay[52] does not require artificial ordering of graphs and is thus independent of the way the canonical code is constructed. The only requirement is that the canonization procedure induces an ordering of the edges of the graph being canonized. An example of McKay algorithm is given in Scheme III. This algorithm produces all canonical edge augmentations of a given graph G having q-1 edges (steps 4-9), resulting in a set S of labeled graphs G' with q edges. Identical graphs are removed from the set S (step 10). Then, in steps 11-16, for every (n,q)-graph G' in S, the algorithm explicitly searches the (n,q-1)-graph it came from. In other words the algorithm searches the parent of every child produced. The parent is obtained removing the last edge e' in CANON(G') (step 12). If the parent (e.g., graph G'-e') is the one that was just augmented (i.e, graph G) then the child is legitimate (step 13), and the algorithm in recursively run with G' (step 14), otherwise, graph G' is ignored.

```
Scheme III: Orderly-Generation-McKay(G)
1.    IF graph G is completed
2.       PRINT G
3.    ELSE
4.        S = ø
5.        FOR all edges e not already in G DO
6.            IF constraints are not violated for the graph G' = G U e
7.               S = S U G'
8.            FI
9.        DONE
10.       Remove duplicates from the set S
11.       FOR all graph G' of S DO
```

```
12.             let e' be the last edge of CANON(G')
13.             IF CANON(G'-e') = CANON(G)
14.                Orderly-Generation-McKay(G')
15.             FI
16.        DONE
17.    FI
```

One issue we have not yet addressed with orderly generation is computational complexity. While orderly generation is certainly faster than labeled enumeration followed by a removal of the duplicated structures, is it the optimum solution? First we have to ask what optimum means when dealing with enumeration. We certainly cannot hope for a polynomial time algorithm since the number of solutions may be exponentially large, and it already takes an exponential time just to write the solutions. The best we can hope for is an algorithm that runs in polynomial time per output. Such an algorithm indeed exists at least theoretically, as was shown by Goldberg.[53] Precisely Goldberg proved that an orderly algorithm can be designed to generate all graphs of n vertices adding one vertex at a time (not an edge) such that the time delay between two outputs is polynomial. In the proof, Goldberg uses the fact that there are always more graphs of n vertices than n-1 vertices, and that canonization can be performed in polynomial time for more than half of the graphs of n vertices. This implies that the enumeration tree always grows, that is, to every n-1 vertex graph corresponds at least one n vertex graph. Unfortunately, that proof cannot be used directly when growing graphs by adding edges, because the number of (n,q)-graphs is not necessarily greater than the number of $(n,q\text{-}1)$-graphs. For example, there is only one $(n,n(n\text{-}1)/2)$-graph, which is the complete graph (each vertex is connected to all others). There is also one $(n,n(n\text{-}1)/2\text{-}1)$-graph, a complete graph without one edge. However, there are several ways of removing a second edge and, thus, there is more than one $(n,n(n\text{-}1)/2\text{-}2)$-graph. Goldberg's result is thus not directly applicable to Schemes II or III. More generally, there is no guarantee that locally restricted graphs, such as molecular graphs restricted by valence sequences, can be constructed in an iterative process such that the number of graphs at a given iteration is always greater than the number of graphs of the previous iteration. While the theoretical complexity of enumerating molecular graphs is still an open problem, in practice as we shall see next, there exist fast algorithms to enumerate molecules.

As far as general graphs are concerned, some codes are available for their enumeration. In particular, two codes to enumerate small graphs and bipartite graphs can be downloaded along with Nauty, a graph canonizer we mentioned earlier.[10]

Enumerating Molecules

Enumerating molecules is not only the main subject of this chapter but it has also been a prolific field of research for decades. Rather than reviewing every single approach that has so far been taken, we have chosen to present examples of orderly generation. Our reasons are many. First, as discussed earlier, orderly generation is the most elegant technique to enumerate graphs. Second, no other technique has had as many applications in chemistry than orderly generation. Finally, focusing on one technique will help the reader understand how molecules are enumerated. As we shall see in all the subsections that follow, the main problem in applying orderly generation to a specific class of molecules is to find the appropriate canonical code. That is, a code that uniquely represents the class of molecules one wants to enumerate, and a code that is easily computable, ideally, in polynomial time.

Acyclic molecular graph enumeration

As with counting, it is simpler to enumerate acyclic structures than cyclic ones. For this reason the field of molecular structure enumeration started with acyclic hydrocarbons with an algorithm published by Nobel Laureate J. Lederberg.[54] The algorithm was later integrated into a code named DENDRAL and was used to enumerate the isomers for a variety of acyclic compounds containing C, H, O, and N atoms.[55] Much could be said about the DENDRAL project which is described in many computer science textbooks as the first expert system. The reader further interested by DENDRAL is referred to the books by Lindsay *et al.* and Gray,[56, 57] where the history of the project is reviewed. A decade after the initial DENDRAL effort, a powerful approach appeared based on the *n*-tuple code developed by Knop *et al.*[58] We present this technique in the context of an orderly algorithm.

The *n*-tuple code is a set of non-negative integers smaller than *n*, the number of atoms of an acyclic molecular structure. Each number in the *n*-tuple represents the degree of an atom in the structure or in one of its substructures. To compute the *n*-tuple of a structure one first chooses a starting atom (a root) as illustrated in Figure 11.(a). For the purpose of this example any atom will do, but as we shall see later the root atom must be the atom with the highest degree if one is to construct a canonical represent of the *n*-tuple. The first element of the tuple is *k*, the degree of the root. Next, the root and the all bonds attached to it are removed from the structure, thus creating *k* disconnected substructures. The process is repeated for each of the *k* substructures where the new roots are the atoms that were bonded to the initial root. The process stops when all atoms have been removed.

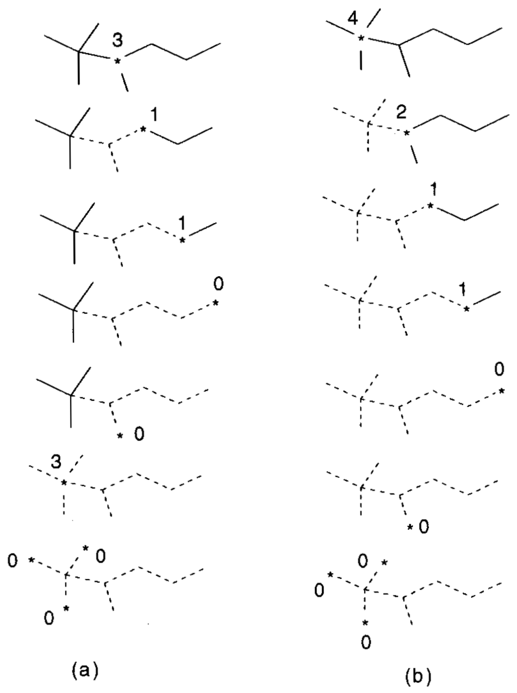


Figure 11. Some n-tuple codes for 2,2,3-trimethylhexane. Successive roots are indicated with a '*' symbol. (a) The code is 311003000. (b) The code is 421100000, this code is canonical.

Looking at Figure 11(a) it is obvious the n-tuple code is nothing else but a list of atom degrees obtained by reading the structure in a depth-first order. All degrees are reduced by one except for the initial root. Now, for any given rooted structure, a canonical n-tuple (cf. Figure 11(b)) is computed using the above procedure, but at each step the tuples associated with the substructures are sorted and read in decreasing lexicographic order. Finally, to compute a canonical n-tuple for an unrooted structure, one computes the canonical n-tuples for all the structures rooted at atoms with the highest degree while keeping the lexicographically maximal tuple as the canonical represent for the structure. Note that there is no need to compute n-tuples rooted on atoms with degrees smaller than the maximum one, as these rooted structures produce lexicographically smaller n-tuples. The code corresponding to Figure 11(b) is the canonical n-tuple of 2,2,3,trimethylhexane since there is only one quaternary carbon is the structure. As shown by Hopcroft and Tarjan[59] the above canonization procedure can be implemented with an O(n) time complexity. Finally it is worth mentioning that modifications of the n-tuple code have been proposed to take into account atom and bond types.[60] Instead of just writing the degree of the atoms in the n-tuple, one also includes atom types and bond orders.

Now that we have a code to canonize acyclic structures, an orderly algorithm can be used. Next, we illustrate the use of the n-tuple code to enumerate alkanes up to n carbon atoms using a McKay type orderly generation (Scheme III). For simplicity all hydrogen atoms are ignored, and carbon atoms may thus have a number of bonds ranging between 1 and 4. As depicted in Figure 12, the initial graph contains one atom and no bond, so its canonical n-tuple is (0).

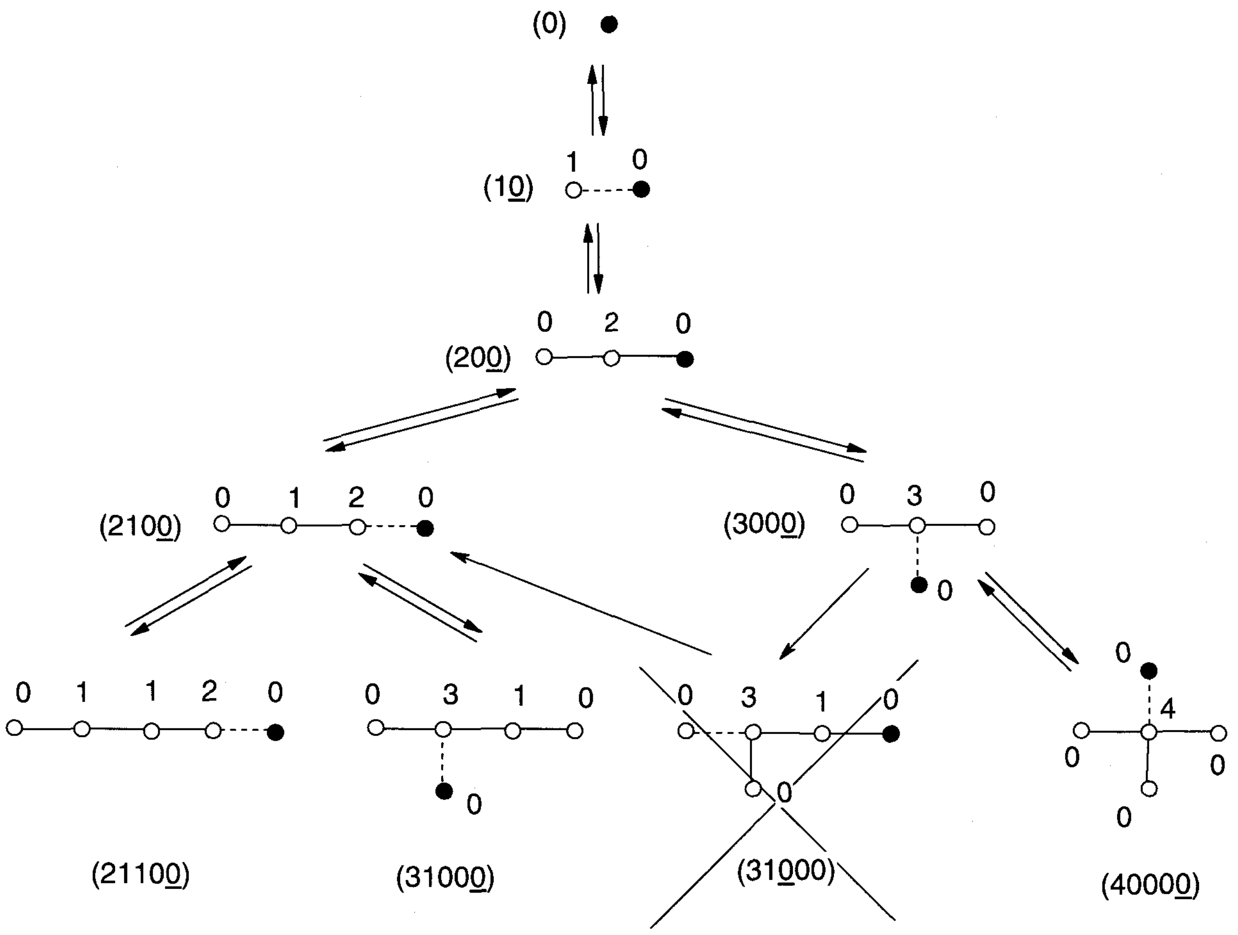


Figure 12. The three pentane isomers obtained with McKay's orderly algorithm and the *n*-tuple code. Hydrogen atoms are not represented. All atoms are carbons and can have up to four bonds. Parent-child and child-parent relationships are indicated with arrows. Canonical *n*-tuples are written in parentheses. At each layer, a bond and a new atom are added. The added atom is represented by a solid node. The last bond/atom in the canonical *n*-tuple is represented by a dashed line, and is underlined in the canonical n-tuple. A graph is rejected when its legitimate parent is not the graph it came from. This case arises when the added bond/atom is not the last digit of the canonical *n*-tuple (the dashed line is not linked to the solid node).

Following Scheme III, we first verify that the construction process is completed for structure *G* (step 1). In the present case, one must check that the required number of atoms *n* is met and that the number of bonds attached to every atom ranges between 1 and 4. If structure *G* passes the completion test it is printed (step 2), otherwise one augments *G* in all possible ways by adding a bond *e* and a new atom (step 5). Augmentations violating the maximum valence requirement are rejected (step 6). For all other *G* U *e* structures, a canonical *n*-tuple *G'* is constructed, and *G'* is added to the set of *n*-tuples *S* (step 7). Duplicated *n*-tuples are removed (step 10). For each resulting *n*-tuple *G'* in *S*, McKay's algorithm removes the last edge of *G'* (step 12), which in the present case is the last digit of the *n*-tuple. If the resulting *n*-tuple equals the *n*-tuple of the initial graph *G* (step 13) then *G'* is a legitimate child of *G*, and the process repeats itself with *G'* (step 14), otherwise *G'* is an illegitimate child and is ignored.

The application of Scheme III to generate alkane structures up to pentane is illustrated in Figure 12, where examples of legitimate and illegitimate parent-child relationships are depicted. Of course Figure 12 could be expanded up to decane, and one could then answer the subsection question "Are there any isomers of decane having seven methyl groups?". As we shall see later, there are more efficient ways to enumerate all decane isomers having seven methyl groups.

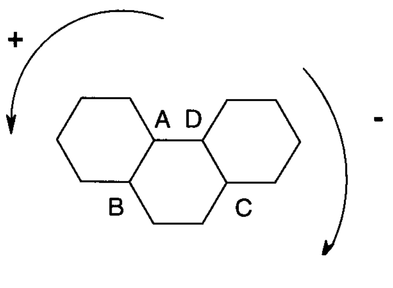
The *n*-tuple technique has lead to numerous implementations and extensions. In particular Contras *et al.* extended the *n*-tuple enumeration algorithm in a series of papers to acyclic compounds with heteroelements and multiple bonds,[60] cyclic structures,[61] mixed compounds,[62] acyclic stereoisomers,[63] and unsaturated stereoisomers.[64, 65] One should also mention the tree enumeration technique proposed by Lukovits.[66] Instead of an *n*-tuple Lukovits uses a compressed adjacency matrix (CAM). The CAM is a vector where each element e_i represents a column *i* of the adjacency matrix (a_{ij}). The value of element e_i is the row number $j < i$ for which a bond appears, i.e., $a_{ij} > 1$. Lukovits proposes a set of rules to generate all trees having a maximal CAM.[67] The technique may not be as efficient as the *n*-tuple code as during the construction process many structures do not meet the rules and are thus rejected.

Benzenoids and polyhex hydrocarbons enumeration

The reader is referred to the "Number of benzenoids and polyhex hydrocarbons" subsection for the definition and classification of benzenoids and polyhex hydrocarbons, as well as for additional references for this class of compounds, which is only partially reviewed due to space limitations. Let us recall that the direct counting approach has difficulties with molecules that cannot be represented by tree-like structures, such as pericondensed polyhexes. Furthermore, the counting approach is unable to separate non-planar polyhexes (helicenes) from planar benzenoids. Consequently, for benzenoids and polyhexes, enumeration is not only a valuable tool that provides a concise description of the structures being enumerated, but enumeration is also used to compute isomer numbers that cannot be derived otherwise.

The first algorithm to enumerate polyhexes was proposed by Balasubramanian *et al.*[68] The enumeration of planar simply connected polyhexes to $h = 10$ hexagons,[69] $h = 11$,[70] and $h = 12$[71] used this algorithm. The next advance in polyhex enumeration came from a code based on the dual graph associated with every polyhex.[72] This code allowed enumeration of all polyhexes for $h = 13$,[73] $h = 14$,[74] $h = 15$,[72] and $h = 16$.[75] The next progress was made by Tosic *et al.*[76] who proposed a lattice based approach using a *"cage"* within which the polyhexes are placed. This method led to enumeration of all polyhexes with $h = 17$.[76] Three years later Caporossi and Hansen[77] developed a McKay type orderly algorithm and enumerated polyhexes up to $h = 21$ and $h = 24$.[78] Finally, in 2002 another lattice based method was proposed and polyhexes were enumerated up to $h = 35$.[79] Next, we briefly describe the orderly generation and the lattice enumeration approaches.

Orderly generation of polyhexes. As usual with orderly generation algorithms, polyhexes comprising h hexagons are constructed from polyhexes having h-1 hexagons. To avoid repetitions, each polyhex with h hexagons is generated from one and only one parent, i.e., a polyhex with h-1 hexagons. As we have already seen with alkanes in Figure 12, once a structure is generated from a potential parent, its canonical code must be scanned to verify if the parent is legitimate. In order to apply Scheme III to polyhexes we only have to find the appropriate canonical code. One possible code used for this purpose is the Boundary Edges Code (BEC).[77] This code is outlined next and illustrated in Figure 13.



	+	-
A	$\underline{5351}$	1535
B	3515	5153
C	5153	3515
D	1535	$\underline{5351}$

Figure 13. BEC code. Canonical codes are underlined. Starting at vertex A and turning clockwise, one first encounters 1 edge from the center face, then, one finds 5 edges belonging to the right face, next are the 3 edges from the center face, and finally 5 edges belonging to the left face. Turning clockwise, the BEC code starting at A is 1535.

Beginning at any external vertex of degree three, which thus belongs to only two hexagons, follow the boundary of the polyhex noting by a digit the number of edges on the boundary for each successive hexagon. The procedure is repeated clockwise and counterclockwise, the canonical code is the lexicographically maximum code. In Figure 13, one observes that the code is unique but may be obtained in several ways in case of symmetry of the polyhex. The high efficiency of the BEC code is due to an alternative way to check whether a polyhex must be considered or not as being legitimate. To this end, Caporossi and Hansen[77] established the following rule: a polyhex is legitimate if and only if the first digit of its BEC code corresponds to the last added hexagon. This simple rule induces the enumeration tree illustrated in Figure 14 up to $h = 4$. Note that the cost of determining whether or not a polyhex is legitimate equals the cost of computing the BEC code, $O(h^2)$. Caporossi and Hansen[77] assessed the computational time per output of their algorithm and it appears to increase quadratically with the system size.

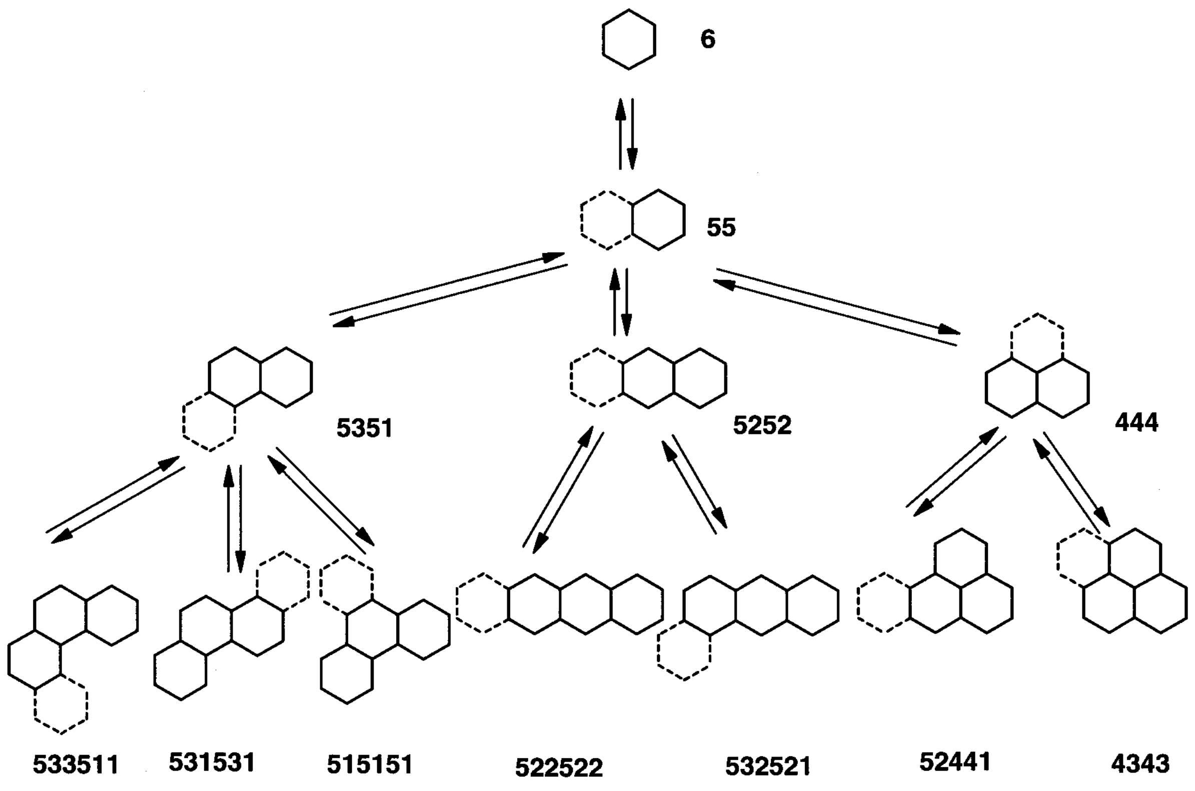


Figure 14. The 7 polyhexes with 4 hexagons obtained with orderly generation and BEC code. At each layer, the last added hexagon (dashed lines) corresponds to the first digit in the BEC code.

Lattice enumeration of benzenoids. Lattice enumeration techniques make use of the fact that there are only eight symmetry groups associated with benzenoids.[35] These are (1) C_s for benzenoids of h hexagons with no rotational or reflection symmetry, (2) C_{2v} for those with one axis of reflection symmetry, (3) C_{2h} for those invariant with respect to rotations through π, (4) D_{2h} for those with two axes of reflection symmetry and invariant with respect to rotations through π, (5) C_{3h} for those invariant with respect to rotations through $2\pi/3$, (6) D_{3h} for those with three axes of reflection symmetry and invariant with respect to rotations through $2\pi/3$, (7) C_{6h} for those invariant with respect to rotations through $\pi/3$, and finally, (8) D_{6h} for those with six axes of reflection symmetry and invariant with respect to rotations through $\pi/3$. In terms of these, the number of benzenoids b_h comprising h hexagons may be written as:

$$b_h = C_s^{(h)} + C_{2v}^{(h)} + C_{2h}^{(h)} + D_{2h}^{(h)} + C_{3h}^{(h)} + D_{3h}^{(h)} + C_{6h}^{(h)} + D_{6h}^{(h)} \qquad [29]$$

where, for instance, $C_s^{(h)}$ is the number of benzenoids of h hexagons with symmetry C_s. Now, let B_h be the number of fixed hexagonal systems. Fixed hexagonal systems are simply all the possible benzenoids one can construct on a hexagonal lattice disregarding rotational and reflection symmetries. From the above definitions of symmetry groups it is easy to verify that:

$$B_h = 12C_s^{(h)} + 6C_{2v}^{(h)} + 6C_{2h}^{(h)} + 3D_{2h}^{(h)} + 4C_{3h}^{(h)} + 2D_{3h}^{(h)} + 2C_{6h}^{(h)} + D_{6h}^{(h)} \qquad [30]$$

Eliminating $C_s^{(h)}$ we arrive at:

$$b_h = \frac{1}{12}(B_h + 6C_{2v}^{(h)} + 6C_{2h}^{(h)} + 9D_{2h}^{(h)} + 8C_{3h}^{(h)} + 10D_{3h}^{(h)} + 10C_{6h}^{(h)} + 11D_{6h}^{(h)}) \qquad [31]$$

The lattice enumeration technique consists of generating and counting all the hexagonal systems that appear on the right-hand side of eq. 31 to evaluate b_h. Let us start with B_h, the number of fixed hexagonal systems of size of h. Generating fixed polygonal systems on lattices can be solved by enumerating self avoiding polygons on lattices. This problem has been studied in the physics literature and will not be reviewed here. The reader interested by this particular problem is referred to the work of Enting and Guttmann.[80] To enumerate benzenoids, Vöge *et al.*[79] use the Enting and Guttmann technique, while Tosic *et al.*[76] use an original algorithm based on a brute force approach enumerating all fixed hexagonal systems on a lattice.

Once B_h has been computed, the other terms of eq. 31 are derived as follows. We first consider the elements of $C_{2v}{}^{(h)}$. Each element of $C_{2v}{}^{(h)}$ can be decomposed into two identical $h/2$ hexagonal systems, joined together at the symmetry axis. Thus, the elements of $C_{2v}{}^{(h)}$ can be generated from the elements of $B_{h/2}$. Similar arguments apply to the elements of $C_{2h}{}^{(h)}$. From the definitions of the symmetry groups given previously, it is easy to verify that the elements of $\mathrm{D}_{2h}{}^{(h)}$ can be generated from the fixed hexagonal systems of $B_{h/4}$, the elements of $C_{3h}{}^{(h)}$ from $B_{h/3}$, the elements of $D_{3h}{}^{(h)}$ and $C_{6h}{}^{(h)}$ from $B_{h/6}$, and the elements of $D_{6h}{}^{(h)}$ from $B_{h/12}$. Thus, all elements in eq. 31 can be computed from B_h. In other words, benzenoids can be counted and enumerated from the enumeration of fixed hexagonal systems.

Results obtained using this approach as well as the orderly generation technique have been compiled in Table 7 in the "Chemical Information" subsection appearing later in the chapter.

Molecular cages enumeration (fullerenes and nanotubes)

Fullerenes, nanotubes, spheroalkanes, and other molecular cages belong to the class of regular graphs. A regular graph is a graph where all the vertices have the same degree. Among the class of regular graphs of interest in chemistry are (k,g)-cages where all the atoms have the same valence k and all rings are at least of size g. We first review the literature for regular graphs and cages, and then describe algorithms specifically designed for fullerenes.

Regular graphs and cages. Enumerating regular graphs is one of the oldest problem in combinatorics. In the 19th century Jan de Vries[81] enumerated all the 3-regular graphs, also named cubic graphs, up to 10 vertices. The first computational approach is due to Balaban,[82] who in 1966 enumerated all cubic regular graphs up to 10, and later 12 vertices.[82] In 1976, Bussemaker *et al.*[83] computed all cubic graphs up to 14 vertices. About the same period Faradzev[51] worked out the case for 18 vertices when he suggested the general orderly algorithm presented in Scheme II. In 1986, McKay and Royle settled the case for 20 vertices[84] while in 1996 Brinkmann[85] enumerated all 24 vertices cubic graphs, and (3,8) cages up to 40 vertices. Finally in 1999, based of the Brinkmann technique, Meringer enumerated all k-regular graphs up to $k = 6$ and a number of vertices ranging between 15 and 24. [86] Meringer's orderly algorithm is an integral part of the latest version of the MOLGEN isomer generator.[87] Next, we describe this algorithm, which a classical example of the Read-Faradzev orderly generation.

Meringer's algorithm generates all k-regular graphs of n vertices. The process starts with an initial graph, G, composed of n vertices labeled 1 through n and no edges. Meringer's algorithm is recursive, thus, following scheme II, in steps (1) and (2) the graph is printed if it is fully constructed. That is, if all the n vertices have k neighbors. When the graph is not fully constructed, in step (4) all edges, e, are enumerated only when they are lexicographically greater than the edges built so far. In steps (5) and (6) the algorithm checks if the graph G U e obtained for each enumerated edge, e, is identical to its canonical representation, i.e., G U e = CAN(G u *e*). When the graph is canonical and the additional constraints of step (5) are verified the same process is repeated; the algorithm backtracks otherwise. The main constraint in step (5) is regularity. All vertices must have at most *k* neighbors and supplementary constraints such as connectivity and minimum cycle size (girth) may also be added. According to its author, the most time consuming part of the algorithm is the canonization step. To reduce the number of times graphs are canonized, not all possible edges are enumerated in step (4), but only the edges attached to the lexicographically smallest vertex having less than *k* neighbors.

Fullerenes and nanotubes. A fullerene is a spherically shaped carbon molecule composed exclusively of five and six membered rings. In the language of graph theory, a fullerene is a 3-regular spherical map having pentagonal and hexagonal faces only. Furthermore, by definition any fullerene C_n, $n \geq 20$, has exactly twelve pentagons and $n/2$-10 hexagons. Because of these restrictions, the polyhexes, benzenoids, and regular cages generators presented earlier cannot directly be used here. For instance, the BEC canonical code cannot be applied because fullerenes do not have edges on their boundaries. The early algorithms that enumerate fullerenes[88-90] do not make use of orderly generation. Yet, there are no reasons why orderly generation could not potentially be applied, provided that a canonical code exists to uniquely identify fullerenes. Next we describe the spiral canonical code for fullerenes,[91-93] we then propose a sketch of a Read-Faradzev orderly generation taken from the algorithms of Fowler and Manolopoulos,[93] and Brinkmann.[85]

The spiral canonical code for a C_{24} fullerene is illustrated in Figure 15. Starting at one face, chose a first neighboring face and an orientation (clockwise or counterclockwise). Visit all faces of the fullerene by recursively choosing a new face as the next one to be visited. The next face must not have already been visited, and must be adjacent to the last face visited.

Additionally, the next face is the first one encountered running around the last face in a clockwise (counterclockwise) direction from the intersection with the next to last face. The code is simply the sequence of face sizes in the order they are visited. The process is repeated choosing all faces one after another as the starting one, choosing all possible first neighbors, and choosing the two possible directions. The lexicographically minimum code is the canonical one. The major pitfall of the spiral code is that not all fullerenes admit ring spirals,[91] however, this problem can be overcome by identifying the edges adjacent to consecutive faces and adding these identifiers to the spiral code.[92]

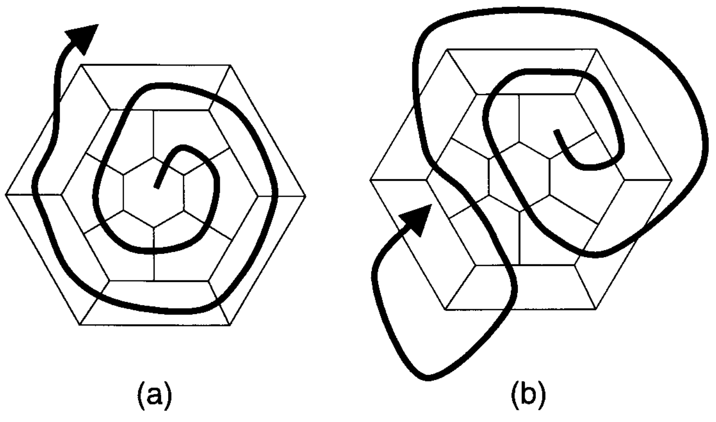


Figure 15. Spiral codes for C_{24}. (a) Starting at a hexagonal face the code is 65555555555556 = $65^{12}6$. (b) Starting at a pentagonal face the code is 55555655655555 = $5^5 65^2 65^5$. Code (b) is canonical.

Now that we have a way to canonize fullerenes, we construct fullerenes adding pentagonal or hexagonal faces one at a time starting with a pentagonal face, otherwise the final spiral codes would not be canonical. In other words, *n* digits spirals (i.e., *n* faces fullerenes) are constructed from *n*-1 digits spirals (i.e., *n*-1 faces fullerenes) by appending to the code either a

'5' or a '6'. Let s_{n-1} be a n-1 digits spiral code, the child $s_n = s_{n-1}5$ ($s_{n-1}6$) is legitimate if the canonical spiral code of that child is indeed $s_{n-1}5$ ($s_{n-1}6$), if not the child is rejected. It is important to realize some spiral codes do not lead to final fullerenes at all. For instance, starting with eleven 5's in the code, i.e., eleven pentagonal faces, we can see that this code cannot lead to a fullerene unless we have no hexagon and the structure to be constructed is C_{20}. Consequently, the orderly generation applied to fullerene creates unproductive branches in the enumeration tree.

Faster than the algorithm described above is the technique proposed by Brinkmann, Dress *et al.*[94-96] Instead of building fullerenes from the ground up, this algorithm generates structures by gluing together "benzenoid" patches composed of five and six membered rings. This approach was taken because fullerenes can be decomposed into either two or three patches following a Petrie path. Petrie paths are constructed as follows: start at any edge e_1 in the fullerene and with a scissor cut that edge. Next cut edge e_2 on the right side of e_1, cut edge e_3 on the left side of the e_2, and repeat the process turning alternatively right and left until you reach an edge e_k that has already been cut. If $e_k = e_1$, you have separated the fullerene in two patches. Now, if $e_k \neq e_1$, the fullerene is also separated in two parts, but the job is not completed because one part is partially cut, i.e., the part containing edges $e_1, e_2, \ldots, e_{k-1}$. Take that part and start again at e_1 but now cut in the opposite direction; you will eventually split the part into two patches, and create a total of three patches. Because any fullerenes can be decomposed into at most three patches, from a given number h of hexagons, all fullerenes can be constructed by attaching in all possible ways a catalogue of all patches composed of at most h hexagons and twelve pentagons. Results obtained using this algorithm can be found in Table 8 in the "Chemical Information" subsection appearing later in the chapter.

Prior to closing this subsection we should also mention a simple algorithm that enumerates the isomers of a toroidal polyhex.[97] Toroidal polyhexes are fullerenes embedded on the surface of a torus. The word fullerene is not quite appropriate here since the authors enumerate only structures having six membered rings (not five). This limitation greatly reduces the number of solutions. The number of isomers is found to increase at only a modest rate that does not exceed 30% of the number of atoms.

General structural isomer enumeration

By general structural isomer enumeration we mean the enumeration of all the molecular graphs corresponding to a molecular formula. We do not include here solutions that construct molecular structures from additional constraints, such as the presence or the absence of substructural fragments. Enumeration with constraints is reviewed in the next subsection.

Techniques to enumerate molecules (including cyclic ones) from a molecular formula appeared in the 1970s. The first algorithm to do so, CONGEN,[98] was a product of the DENDRAL project. The solution consisted of decomposing the molecular formula into cyclic substructures, which were combined by bridges to get molecules. The cyclic substructures were built from a database of 3,000 elementary cycles. A second approach, simpler in principle, has been the technique chosen by the researchers involved in the CHEMICS project.[99] In this approach only canonical structures are generated. However, orderly generation was not applied in the earlier version of CHEMICS. Instead, all labeled structures were generated and non-canonical ones were rejected. A similar approach was also taken by the authors who developed the ASSEMBLE generator,[100] although this code was designed to combine fragments. Since the above initial developments, CONGEN, CHEMICS, and ASSEMBLE have lead to numerous improvements, most of which involve enumeration with constraints.

Another development to enumerate isomers has been a method based on an atom's equivalent classes. In this method pioneered by Bangov,[101] and generalized by Faulon,[102] the atoms corresponding to the molecular formula are partitioned into equivalent classes. Next, a class of atom is selected and all the atoms of the class are saturated; that is, bonds are added until each selected atom has a number of bond equals to its valence. Atom saturation is performed in all possible ways and to avoid generating isomorphic structures, non-canonical graphs are rejected. For each resulting graph, equivalent classes are computed again, a new unsaturated class is chosen, and the process is repeated until all atoms are saturated. It is worth noting that with the equivalent-classes technique, one can chose the atoms to be saturated. Thus, one can drive the process to first build tree-like structures, choosing classes of atoms that do not create cycles when being saturated, and then create cycles adding bonds to the unsaturated atoms of the trees. The advantage of building tree-like structures first is that one can canonize them efficiently using, for instance, the *n*-tuple code mentioned earlier. For acyclic isomers the equivalent-classes algorithm is efficient since canonization can be performed in linear time. However, for all other compounds, the cost of canonization has to be factored in.

The next approach to enumerate isomer is orderly generation. One of the first algorithms is due to Kvasnicka and Pospichal.[103] Their orderly technique is based on Faradzev's algorithm. The proposed solution constructs all molecular graphs of maximum valence matching given numbers of atoms and bonds. The technique was soon modified to enumerate all molecular graphs matching a prescribed valence sequence.[104] Faradzev's orderly generation was also used in developing the SMOG program that enumerates compounds from molecular formulae using fragments.[105, 106] The isomer generator MOLGEN[87, 107, 108] is also based on orderly generation.

The latest development with isomer enumeration is the method of homomorphisms proposed by Grüner *et al.*[87] Interestingly, the homomorphism method is a systematization of the early solution developed within the DENDRAL project. The homomorphism method has been implemented in the latest version of MOLGEN.[98] The enumeration relies on a strategy of determining how all molecular graphs with a given valence sequence, can be built up recursively from regular graphs. Grüner *et al.* observe that any molecular graph *G* can be decomposed into two subgraphs: *T,* a subgraph comprising all atoms of a fixed valence, for instance the largest valence, and *H,* a subgraph composed of the remaining atoms. Attached to the two subgraphs an incidence structure, *I,* is constructed such that each column corresponds to an atom *t* of *T*, each row to an atom *h* of *H* and noting a bond connecting two atoms *t* and *h* by the entry 1 in the corresponding place of *I*. The authors then prove that all possible valence sequences for *T* and *H* and all possible numbers of entries 1 in each row and each column of *I* can be determined directly from the valence sequence of *G*. The above decomposition of the valence sequence is repeated recursively until all resulting valence sequences correspond to regular graphs. The strategy obviously reduces the construction problem of molecular graphs with prescribe valence sequences to that of regular graphs and the problem of pasting the subgraphs *T* and *H* together. Regular graphs are constructed using Meringer's algorithm[86] presented earlier, and all possible ways of pasting *T* and *H* are enumerated using an orderly algorithm. According to the authors the resulting algorithm is very fast as it has been able to determine up to 10^{30} molecular graphs (without actually constructing them) corresponding to valence sequences up to 50 atoms.

Molecular graph enumeration with constraints

Molecular structure enumeration subjected to constraints has practical application in structure elucidation and molecular design. Many codes have been developed to address these two applications, most of them can be found in the section entitled "Enumerating Molecules: What are the uses". For structure elucidation, the constraints are generally composed of fragments that must be present and/or absent in the final solutions. With molecular design, the goal is to generate all the structures matching a specified property or activity. This problem, also named inverse imaging, is generally solved in a two steps procedure. First, from the target property or activity a molecular descriptor is computed. This is usually done thought a quantitative-structure activity relationship where the molecular descriptors are fragments, or topological indices. In a second step, all structures matching the descriptor value are enumerated. We next present the methods that have been developed for structure elucidation and molecular design purposes.

Enumerating structures using molecular fragments. We first consider the simple case where the molecular fragments do not overlap. Each fragment must be unsaturated and, thus, contain some free bonds or bonding sites. Then, the problem consists of connecting the bonding sites together in all possible ways. This process can be solved by generating all possible labeled graphs where the vertices are bonding sites. Duplicates can be eliminated in a post-process,[109] or non-canonical graphs can be rejected as they are generated such as in ASSEMBLE[110] and CHEMICS.[111] Another solution is to use the equivalent-classes algorithm, where the equivalent classes are computed only for the unsaturated atoms and of course only these atoms are saturated.[102] Orderly generation can and has been used to enumerate structures from fragments.[105, 112] During the orderly process the search for canonical structures is performed without permuting the elements of the adjacency matrices that corresponds to the fragments. Any of the aforementioned algorithms can be used to answer the subsection question, “are there any isomers of decane having exactly seven methyl groups?” All solutions (if any) must contain seven methyl groups. Since the final structure has the molecular formula $C_{10}H_{22}$, the additional fragments are 3 carbon atoms and one hydrogen atom. The above 11 fragments were given as input to the equivalent-classes algorithm the code returned two solutions: (2,2,3,4,4)-pentamethyl-pentane and (2,2,3,3,4)-pentamethyl-pentane.

In most structure elucidation instances fragments unfortunately do overlap. For instance, consider the fragments provided by ^{13}C NMR spectra. To each ^{13}C NMR peak there is a corresponding fragment (the environment of a ^{13}C carbon atom) and two neighboring atoms in the probed structure have corresponding overlapping fragments. The problem of overlapping fragments can be addressed with manual intervention as in GENOA[113] another product of the DENDRAL project. At first the code’s user selects one fragment as a core. The user then chooses a second fragment and the code generates all possible ways of breaking those two fragments into non-overlapping, ever smaller fragments. The process is repeated until all fragments have been decomposed into non-overlapping ones. Final structures are then generated assembling the non-overlapping fragments using a technique similar to those we just presented.

More systematic is the approach taken with the EPIOS code.[114, 115] A large database of assigned ^{13}C NMR spectra is the source of a library of carbon-centered fragments to which are assigned chemical shifts and signal multiplicities. Using the experimental spectrum, fragments are extracted from the database and the construction proceeds by attaching carbon atoms only if their fragments overlap. Partially assembled structures with chemical shift deviations that exceed a preset threshold are discarded. Once the structures are fully assembled, a spectrum prediction code is run and the predicted spectrum is checked against the experimental one. Structure assembly using overlapping information is also the method implemented in the SpecSolv system.[116]

Another method dealing with overlapping fragments was devised using the so called *signature equation*.[9] The signature of an atom is a fragment comprising all atoms and bonds that are at a specified distance *h* from the probed atom. The fragment is written as a tree with a height equal to the specified distance, the tree is canonized, and the signature is written reading the tree in a depth first order. Examples of signatures of various heights are given in Figure 16.

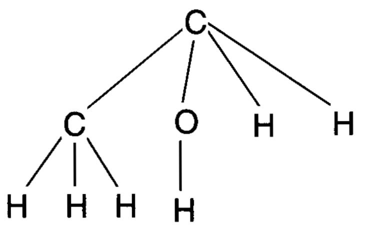


Figure 16. The figure depicts the fragment centered on the carbon atom attached to the alcohol group in ethanol. The height-0 signature of this carbon atom is ${}^{0}\sigma(C) = C$, the height 1 signature is ${}^{1}\sigma(C) = C(COHH)$, and the height 2 is ${}^{2}\sigma(C) = C(C(HHH)O(H)HH)$. The height 1 signature of ethanol is obtained summing the height 1 signatures for all atoms, ${}^{1}\sigma(\text{ethanol}) = C(COHH) + C(CHHH) + O(CH) + 5H(C) + H(O)$. The height 1 signature of the bond C–O is the difference between the signature of ethanol and the signature of the structure where the bond has been removed, ${}^{1}\sigma(C–O) = C(COHH) + O(CH) - C(CHH) - O(H)$.

The signature of a molecule or a molecular fragment is simply the sum of all its atomic signatures. The signature of a bond is the difference between the signature of the structure containing the bond and the signature of the structure where the bond has been removed. Now, assuming we know the signature up to a certain height of a yet unresolved compound and assuming we also know that the compound contains a number of fragments that may or not overlap, the purpose of the signature equation is to compute lists of non overlapping fragments matching the signature of the unresolved compound. Simply stated, fragments and signatures are related by the expression: signature of the fragments + signature of the interfragment bonds = signature of the unknown compound. Formally, lists of non-overlapping fragments are computed solving the equation with unknowns x_i and y_j:

$$\sum_i x_i {}^h\sigma(\text{fragment } i) + \sum_j y_j {}^h\sigma(\text{bond } j) = {}^h\sigma(\text{unknown compound}) \qquad [32]$$

The variables x_i and y_j are, respectively, the number of fragments i present in the final structure, and number of interfragment bonds j. The signature equation (eq. 32) is an integer equation and can be solved using integer linear programming (ILP) tools.[117] Note however, that in general ILP problems are intractable.[12] Each solution of eq. 32 is a list of non-overlapping fragments and interfragment bonds. To enumerate the final structures each list of fragments and interfragment bonds is fed to an isomer generator working with non-overlapping fragments. In the structure elucidation instances where the signature equation was used[118] elemental analysis, NMR and functional group analysis provided the height 0, 1 and 2 signatures of the unknown compounds, and fragments were derived from chemical degradation and pyrolysis.

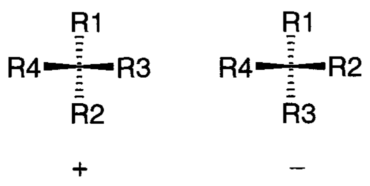
An elegant approach dealing with overlapping fragments is the structure reduction method proposed by Christie and Munk.[119] In contrast with all enumeration algorithms we have presented so far, this method begins with a hyperstructure containing of all possible bonds between unsaturated atoms. The algorithm removes inconsistent bonds until valences of atoms are respected. This results in a more efficient way to deal with overlapping fragments since all the fragments are contained (i.e., are subgraphs) in the hyperstructures, and as bond deletion occurs, the resulting graphs are kept if they still contain the fragments and are rejected otherwise. While it is not clear from reading the original paper on structure reduction how duplicated structures are removed, orderly generation can certainly be used to avoid the production of duplicates. Checking that fragments occur in a given structure requires running a subgraph isomorphism routine. As already stated, general subgraph isomorphism is an intractable problem.[12] In a recent development the structure reduction method was coupled with a convergent structure generation technique.[120, 121] In this technique instead of having a list overlapping fragments, a network of substructures is first constructed. Substructures are linked in this network when they overlap, and alternative neighborhoods are indicated when overlapping is ambiguous. The initial structure is a hyperstructure composed of all possible bonds between atoms. The reduction method is used to determine all possible ways in which the substructures of the network can be mapped to the actual atoms of the structure being constructed.

Enumerating structures using molecular descriptors. Enumerating molecules matching molecular descriptors or topological descriptors is a long-standing problem. Surprisingly, there are not many reports in the literature providing answers to the question. Most of the proposed techniques are stochastic in nature and are reviewed in the "Sampling structures" subsection. In a series of five papers Kier, Hall, and co-workers[122-126] reconstruct molecular structures from the count of paths up to length $l = 3$. Their technique essentially computes all the possible valence sequences matching the count of paths up to length $l = 2$. Then, for each valence sequence, all the molecular structures are generated using a classical isomer generator (cf. General structural isomer enumeration subsection), and the graphs that do not match the path length $l = 3$ count are rejected. Skvortsova *et al.*[127] use a similar technique but from the count of paths they derive a bond sequence in addition to the valence sequence. A bond sequence counts the number of bonds between each distinct pair of atom valences. The two sequences are then fed to an isomer generator that produces all the structures matching the sequences. Regrettably, the authors do not provide details on how the isomer generator deals with the bond sequence. Another approach to enumerate molecular graphs matching a given signature has appeared recently.[128] As defined earlier the signature is the collection of all atoms environments in a molecule (cf. Figure 16). Like other fragmental molecular descriptors, it has been shown that signature works well in quantitative-structure activity relationships.[129] The input information to the algorithm is a signature. To each atomic signature one associates an atom in the initial graph. At first, the graph is composed of isolated atoms without any bond. The construction proceeds by adding bonds one at a time using the equivalent-classes technique (cf. General structural isomer enumeration subsection). Orderly generation can also be used to enumerate structure matching signatures. During the generation process, bonds are created only if the signatures of the bonded atoms are compatible, and the resulting graph is canonical. This algorithm is capable of enumerating molecular structures up to 50 non-hydrogen atoms on a time scale of few CPU seconds.[128]

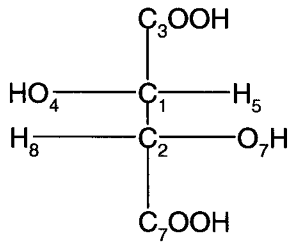
Stereoisomer enumeration

Few approaches have been reported to enumerate stereoisomers.[63, 64, 130-134] We describe here the technique proposed by Nourse.[130] This method has been developed within the CONGEN[131] structure generator, but is also the method used by MOLGEN.[108] Nourse's technique computes all stereoisomers of a given structural isomer. Thus, to enumerate the stereoisomers of a given molecular formula one first generates all structural isomers using the techniques presented earlier. Then, for each structure, one applies Nourse's algorithm. There are essentially three steps in this algorithm. (1) All potential stereocenters are determined for the given structural isomer. (2) A permutation group called the configuration group is constructed from the automorphism group of the structure (cf. definition in "From Graph Theory to Chemistry" subsection). (3) The permutations of the configuration group are applied to all possible orientations of the stereocenters, and orientations found identical under the permutations are removed. The number of stereoisomers is the number of remaining orientations.

A stereocenter is defined to be any trivalent or tetravalent atom with at most one hydrogen which is not part of an aromatic system or cumulenes with H_2-ends, and not triple bonded. A stereocenter has two possible orientations induced by the labels of the neighboring atoms. These labels are simply the atom numbers defined by the generator that was used to produce the structural isomer and these numbers remain unchanged during stereoisomer enumeration. Because the orientation is defined by an arbitrary labeling, the notation +,- is used instead of the *R*,*S* nomenclature. However *R*,*S* notations can be restored in a post process.[131] Let R1 < R2 < R3 < R4 be four atom labels attached a given stereocenter, the two possible orientations are:



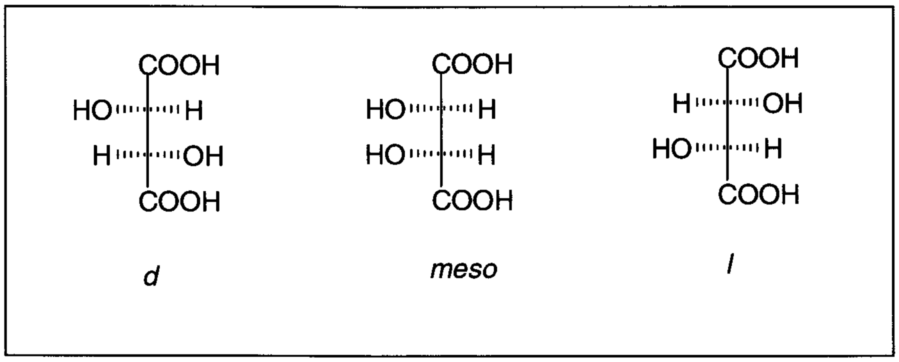
For a structure comprising n stereocenters, each having two possible orientations, there are 2^n potential stereoisomers. Taking the example of tartaric acid of Figure 17.(a), this structure has two stereocenters (C_1 and C_2). The potential stereoisomers are [++], [+-], [-+], and [--].Some of these stereoisomers are identical due to the symmetry of the structure. Using the labels of Figure 17.(a), there are only two permutations in the automorphism group preserving the structure of tartaric acid: (1)(2)(3)(4)(5)(6)(7)(8) and (12)(36)(47)(58). In this case the configuration group is simply the set of permutations of the automorphism group restricted to the stereocenters: (1)(2) and (12). To compute the exact number of stereoisomers one applies the configuration group to all potential stereoisomers, and removes all equivalent orientations. The application of the configuration group on the four possible stereosiomers of tartaric acid is given in Figure 17(b). The three resulting stereoisomers are depicted in Figure 17(c).



(a)

	(1)(2)	(12)	stereoisomer
[++]	[++]	[++]	*d*
[+ -]	[+ -]	[- +]	*meso*
[- +]	[- +]	[+ -]	*meso*
[- -]	[- -]	[- -]	*l*

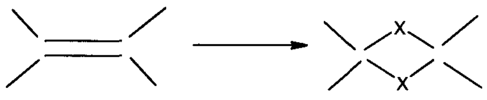
(b)



(c)

Figure 17. The stereoisomers of tartaric acid. (a). Tartaric acid structural isomer with atom labels 1 through 8 (only atoms attached to stereocenters are labeled). (b) Application of the configuration group {(1)(2), (12)} on the four possible stereoisomers. The second and third stereoisomers are identical. (c) The three resulting stereoisomers, a *meso* form and a *dl* pair.

From the tartaric acid example it may seem that the configuration group is no different than the automorphism group restricted to the stereocenters. However, there are more complicated cases where permutations can change the orientations of stereocenters even when the stereocenters are not permutated. As an example consider the permutation (1)(24)(3) acting on the labels of 1,2,3,4-tetrachlorocyclobutane. Stereocenter C_1 is attached to C_2, C_4, a chlorine atom, Cl, and a hydrogen atom, H. The permutation (1)(24)(3) change this order to C_4,C_2, Cl, and H, consequently, the orientation of C_1 is reversed by (1)(24)(3). The same observation can be made for C_3. To indicate that the orientations of C_1 and C_3 are reversed by the permutation (1)(24)(3), Nourse uses the notation (1')(24)(3'). Application of (1')(24)(3') on the stereoisomer [++++] gives the correct configuration [-++-], which differs from [++++], the configuration given by (1)(24)(3). Finally, a stereoisomer induced by double bonds can also be enumerated using Nourse's technique. When double bonds are involved, a special configuration group is computed. This group is the product of the atom automorphism groups and bond automorphism groups. A simpler solution was latter suggested by Wieland *et al.*[135]and consists of converting double bonds into single bonds with fictitious bivalent nodes:



Expanding on Nourse's technique, Wieland[133] proposed an enumeration algorithm of stereoisomers where the valence of the stereocenters can be larger than four.

To conclude this subsection on enumeration, it seems that enumerating structural isomers is no longer a technical challenge. The reader not convinced of this can access the web page of the journal MATCH,[136] enter any molecular formula, and visualize the list of corresponding isomers. The algorithm used to produce this list is MOLGEN. While not every compound family can be counted, as far as isomer enumeration is concerned up to 50 non hydrogen atoms, all molecular graphs can be enumerated according the authors of MOLGEN. Unfortunately, structural elucidation and molecular design problems do not fit this optimistic picture. The pitfall of isomer enumeration is the number of solutions produced. Of course, the number of solutions can be reduced by adding constraints, but, the problem becomes computationally harder and most likely intractable, especially when dealing with overlapping fragments. The usual way to deal with intractable problems in computer science is to use stochastic techniques where solutions are only guaranteed up to some probability. The purpose of the next section is to review the stochastic techniques used to sample molecular structures for the purpose of structure elucidation and molecular design.

Sampling Structures: What is the decane isomer with the highest boiling point?

The premise of the sampling approach is the following question: Is it necessary to generate all of the molecular graphs corresponding to a set of constraints in order to design compounds having specified activities or properties? As far as structure elucidation is concerned, the question is whether or not the concept of a unique chemical graph has a physical or chemical significance for complex natural compounds such as *lignin*, *coal, kerogen,* or *humic substances*. As far as sampling is concerned, there is no method of choice like there was for counting molecules (Polya's theory) and enumerating structures (orderly generation). The reason perhaps is that the field is relatively new. Both in graph theory and computational chemistry, the techniques to sample graphs and chemical graphs appeared mostly in the last decade. In the subsections that follow we first summarize what can be learned from graph theory about sampling graphs and then we review their applications in chemistry.

Sampling labeled and unlabeled graphs

Randomly sampling labeled graphs of n vertices and q edges can easily be done selecting at random q pairs of vertices in the set of $n(n-1)/2$ possible pairs. Such a random selection can be done with or without replacement depending on whether or not one wishes to create multiple edges.

As we have already seen with counting and enumeration, unlabeled graphs are harder to deal with than labeled ones. Nijenhuis and Wilf[137] have shown how to sample unlabeled rooted trees. The approach was extended by Wilf[138] who gave an algorithm to sample unlabeled unrooted trees. The algorithm is based on a counting series for trees. More complicated is the case of cyclic graphs. Dixon and Wilf[139] were the first to give an algorithm for sampling unlabeled graphs with a specified number n of vertices. First, a permutation, π, of n vertices is chosen in the set of all possible permutations, that is, in the symmetric group S_n. As an example, assume the selected permutation is $\pi = (135)(246)$ (cf. C_3^- in Figure 3). Next, a graph is constructed at random from those graphs that are fixed by π, i.e., graphs like benzene that remain unchanged under the action of π. To construct this graph, the permutation π^* acting on the edges is computed from π, where for any edge $[i,j]$, $\pi^*([i,j])=[\pi(i),\pi(j)]$. Using our benzene example we have $\pi^* = (12\ 34\ 56)(13\ 35\ 15)(14\ 36\ 25)(16\ 23\ 45)(24\ 46\ 26)$. Then, for each cycle of π^* independently, one chooses with probability ½ whether *all* or *none* of the edges of the cycle will appear in the graph. Taking our benzene example one may chose edges in cycles (12 34 56) and (16 23 45) to be turned on as in Figure 18(a) or edges in cycles (13 35 15)(14 36 25)(24 46 26) as in Figure 18(b). Both of resulting graphs are drawn at random from the set of all possible unlabeled graphs of six vertices.

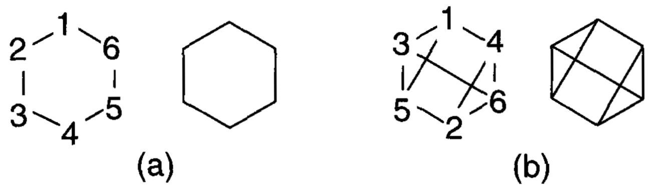


Figure 18. Two unlabeled graphs drawn at random and unchanged under the permutation $\pi = (135)(246)$.

The Dixon and Wilf technique was later expanded by Wormald[140] to sample regular graphs with degrees equal or greater than 3, and by Goldberg and Jerrum[127] to graphs of prescribed degree sequences. The case of degree sequences is of particular interest to chemistry and, in fact, in the paper published by Goldberg and Jerrum an extension to sample molecules is given. Their algorithm is a two-step procedure. First, a core structure that does not contain vertices of degree one or two, is sampled using a Dixon-Wilf-Wormald's type algorithm. Then, the core is extended adding trees and chains of trees (vertices of degree one or two). Interestingly, a parallel can be drawn between Goldberg and Jerrum's core structures and the cyclic substructures of CONGEN,[98] or the regular subgraphs of MOLGEN[87] (cf. General structural isomer enumeration subsection). In all of these approaches, structures are enumerated or sampled by first constructing cyclic subgraphs and then either connecting these subgraphs together or adding vertices and edges that do not create additional cycles. The main result of Goldberg and Jerrum's paper is that molecules can be sampled in polynomial time. This is quite an interesting result considering that the computational complexity of counting and enumerating molecules are still open questions.

Sampling molecules

As with enumeration, sampling chemical structures is used in structure elucidation and molecular design applications. With both applications in mind, three different techniques have been developed: random sampling, Monte-Carlo sampling, and genetic algorithms.

Sampling molecules at random

The first published sampling technique is a generator that constructs linear polymers at random.[141] The random construction is repeated until a polymer is found matching a given set of physical properties. Note that the method is time consuming since the solutions are not refined as the sampling progresses. In the context of drug design, a random sampling technique was proposed[142] to generate random structures by combining fragments. Specifically, fragments are chosen from a database of known drugs with a probability proportional to some statistical weight. Bonding sites are picked randomly for the chosen fragments and for the molecule built so far, and the two are joined together. Fragments are added in such a manner until the total molecular weight exceeds some predefined threshold. The random selection of bonding sites for fusion often produces structures that are chemically unstable or unusual. These structures are eliminated during a selection process based on topological indices and quantitative structure activity relationships. The structures that survive selections are archived in a database of compounds to be considered for synthesis. As in the polymer case, this latter approach appears to be time consuming for molecular (drug) design purposes, since the solutions are not improved as the algorithm progresses. Additionally, the above techniques may generate duplicated structures since they essentially sample labeled graphs. In the context of structure elucidation, a random sampling of non-identical molecular graphs was proposed in 1994.[118] The method is a randomized version of a deterministic structure generation algorithm. Underling all algorithms enumerating molecules is a construction tree (cf. examples in Figures 12 and 14) and that method selects branches at random instead of exploring all of them. Structures produced by the random selection are different if the branches of the construction tree lead to non-identical structures. Such is the case with the orderly algorithm or the equivalent-classes algorithm the sampling technique was based on. Running the algorithm it was observed that large samples of non-identical structures could be generated quite efficiently. Aside from generating non-identical structures at random the above sampling technique also provides an estimate of the number of solutions. This number is then used to carry out statistical analysis, for instance mean values and standard deviations of some properties calculated using molecular simulations on the sample can be extrapolated to the entire population of potential structures.

Monte Carlo sampling of molecules

Random sampling techniques are appropriate to calculate average properties of compounds matching specific constraints, but are rather time consuming when used to search for the best compounds matching target properties or experimental data. In such an instance, optimization methods such as Monte-Carlo or Genetic Algorithms are best suited. Monte-Carlo (MC) and Simulated Annealing (SA) are simple algorithms that were initially designed to provide efficient simulations of collections of particles in condensed matter physics.[143] In each step of these algorithms, a particle is given a small random displacement, and the resulting change, ΔE, in the energy of the system is computed. If $\Delta E \leq 0$, the displacement is accepted, and the new configuration is used as the starting point of the next step. The case $\Delta E \geq 0$ is treated probabilistically: the probability that the configuration is accepted is $\exp(-\Delta E/\mathrm{k}T)$, where k is the Boltzmann constant and T the temperature. With MC the simulations are carried out at equilibrium at a constant temperature T, while with SA the temperature is decreased according to a predefined cooling program (annealing schedule). Using a cost function in place of the energy and defining configurations by a set of parameters, it is straightforward with the above procedure to generate a population of configurations for a given optimization problem. For instance, SA techniques have been used to search for the global minimum of energy in conformational space.[144] For conformational isomers the random displacement of the MC/SA algorithm consists of slightly modifying the conformation by either moving atoms or rotating bonds.

In the structural space, any MC/SA random displacement must consist of changing the connectivity between the atoms. A solution to this problem, proposed by Kvasnicka and Pospichal,[104] and illustrated in Figure 19, is to introduce perturbations in bonding patterns starting at a randomly chosen atom. Specifically, assuming an initial structure is constructed, a linear code is computed for this structure, and atoms are ordered according to the code. Examples of suitable codes are the n-tuple code for acyclic compounds, and the connectivity stack. Next, an atom is chosen at random and the code is randomly modified starting at the chosen atom. Not every perturbation is a valid one, for instance one needs to check that after a perturbation the valences of the atoms and the total number of bonds are maintained.

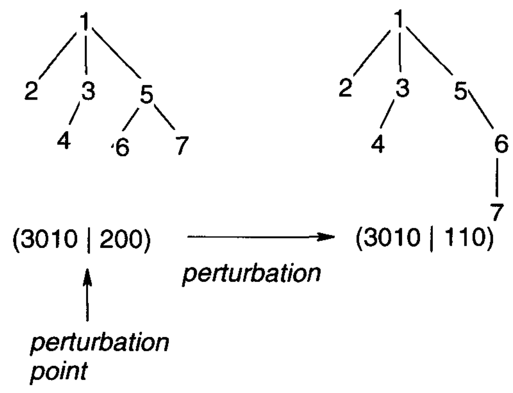


Figure 19. Perturbation of the n-tuple code of a hydrogen suppressed C_7H_{16} isomer. Stating from a randomly selected point the code is randomly modified and 2-methyl-hexane is obtained by bond perturbation of 1,2-dimethyl-pentane.

A disadvantage of the above perturbation technique is that it is difficult to control to what extent the structure is changed since bonding pattern changes start at a randomly chosen atom. Ideally, in the spirit of the MC algorithm, one would like to keep the random displacement as small as possible. To this end, another solution to the random displacement came about observing that connectivity between atoms can be changed by deleting bonds, creating bonds, or modifying bond order.[9] With the convention that a bond is deleted when its order is set to zero and a bond is created when its order is switched from zero to a positive value, all changes of connectivity can be performed by modifying the bond order. Because all structures must have the same total number of bonds, when a bond order is increased, another bond order must be decreased. Hence, changing the connectivity implies the selection of at least two bonds, or four atoms, x_1,y_1,x_2, and y_2. Let a_{11}, a_{12}, a_{21}, and a_{22} be the order of the bonds $[x_1,y_1]$, $[x_1,y_2]$, $[x_2,y_1]$ and $[x_2,y_2]$ in the initial structure and let b_{11}, b_{12}, b_{21}, and b_{22} be the order of the same bonds after the random displacement occurs. The random displacement is performed by a bond order switch. Precisely, a value $b_{11} \neq a_{11}$ is chosen at random verifying:

$$b_{11} \geq \mathrm{MAX}(0, a_{11}-a_{22}, a_{11}+a_{12}-3, a_{11}+a_{21}-3) \qquad [33]$$

$b_{11} \leq \mathrm{MIN}(3, a_{11}+a_{12}, a_{11}+a_{21}, a_{11}-a_{22}+3)$ [34]

The above equations are derived using the fact than bond orders range between 0 and 3. The orders for all other bonds are computed maintaining the valences of the atoms:

$b_{12} = a_{11}+a_{12}-b_{11}$ [35]

$b_{21} = a_{11}+a_{21}-b_{11}$ [36]

$b_{22} = a_{22}-a_{11}+b_{11}$ [37]

It has been shown that all possible structural isomers of a given molecular formula can be reached using the above bond order switch.[145] It is also worth noticing that every structure produced by the bond order switch is a valid one. Thus, contrary to the bond perturbation technique of Figure 19, there is no need to check for structure consistency. The bond order switch was used with a SA algorithm to search compounds having the maximum and minimum Wiener indices. The correct solutions were found up to 84 carbon atoms. There exist many quantitative structure-activity/property relationships between the Wiener index and the boiling point of organic compounds.[40] These relationships may not be linear but, as a general rule, the larger the Wiener number is, the higher the boiling point. Thus, searching for compounds having the highest boiling point can be achieved by finding molecular graphs having the maximum Wiener index. For dodecane, the maximum Wiener index was found by the above algorithm to be W = 286, and the corresponding structure is the linear dodecane isomer. The bond order switch was also integrated in the SENECA software and used to elucidate structures as large as triterpenes matching experimental 1D and 2D NMR spectra.[146]

Genetic algorithms to sample molecules

A genetic algorithm (GA) is a method of producing new individual examples from combinations of previous individuals, or, parents. The algorithm has the same logical structure as inheritance in biological systems. The probability that an individual will be produced and participate as a parent in a succeeding generation must be defined by some standard. For optimization purposes, the suitability of an offspring is usually assessed using a "fitness" function. This is a direct analogy to Darwin's evolutionary rules of natural selection and survival of the fittest.

The applications of GAs in chemistry have already been reviewed in this series of books.[144] We focus here on the use of genetic algorithms to sample and search molecular graphs. The implementation of a GA usually invokes three data processing steps on the genetic code: mutation, crossover (recombination), and selection. The genetic codes suitable for chemical graphs are the ones we have already seen with the MC/SA algorithms, the n-tuple code, and the connectivity stack. Mutations of the genetic code can be performed the same way random displacements are carried out in MC/SA, that is, bond perturbation or bond order switch. Several steps are required to crossover genetic codes. First two parents are selected. Next, a crossover point is chosen at random, the two codes are spliced into two segments, and the corresponding fragments are recombined taking a segment from each code. The crossover operation is illustrated in the Figure 20 with the n-tuple code. As with bond perturbations in MC/SA, there is no guarantee that a crossover operation will maintain the valences of the atoms. Thus, all structures created by crossover must be checked for consistency. This is a disadvantage that GA has versus MC/SA when bond order switching is used. An interesting solution to avoid consistency check during crossover operation appeared in 1999.[147] In this solution a bond is chosen randomly in each parent. The bond is deleted and if the parent is not spliced into two pieces, a second bond is then removed in the shortest path linking the two atoms attached to the deleted bond. The process of bond deletion is repeated until the parent is cut into two disconnected parts. The four resulting pieces (two per parent) are then recombined by saturating the atoms where bonds have been deleted.

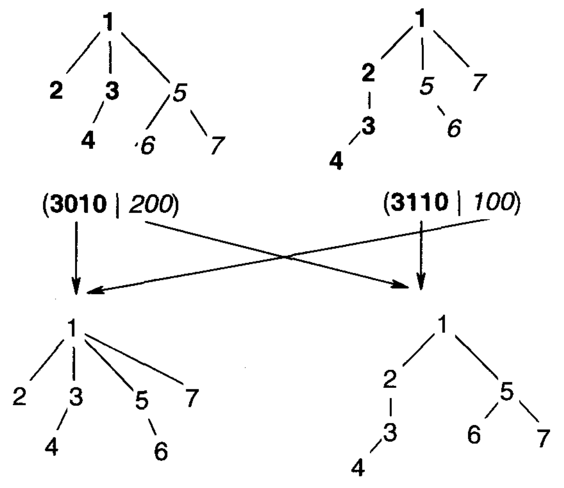


Figure 20. Crossover operation with the n-tuple code.

The last genetic operation is selection. Elements of the population are selected to form the next generation using a problem specific fitness function. Taking the simple example of searching for the structure having the highest boiling, the fitness function can be for instance the Wiener index of the structure.

The first use of a genetic algorithm to sample molecular structures was in the context of the design of polymers with desired properties.[148] Later, a paper appeared to construct combinatorial libraries[149] and targeted libraries[150] for drug design purposes. It is worth mentioning that these applications are limited to linear genetic codes and are thus unable to create individuals by recombining parents in a cyclic manner. A general GA algorithm that includes cyclic recombination was implemented for the purpose of structure elucidation of organic compounds from ^{13}C NMR spectra.[151] In this GA algorithm mutations are performed using bond perturbations as in Figure 19, and crossovers are carried out as in Figure 20. The selection operator is a root-mean-square deviation between the experimental chemical shifts and the predicted chemical shifts obtained with neural network technology. Structures up to 20 heavy atoms have been elucidated using this algorithm.

Enumerating Molecules: What are the uses?

Chemical Information

The combination of counting series and enumerating algorithms described previously in this chapter has allowed researchers to generate isomer lists for not only popular compounds, but for several specific compound classes as well. In this next section of the chapter, we provide a brief review of isomer lists available in the literature as well as tabulate some important and popular lists to provide the reader a quick resource for this information.

Alkanes and alkane-like substances have captured the interest of researchers in isomer enumeration for a long time owing to their commercial importance. For example, Henze and Blair published the first isomer enumeration of alkanes in 1931.[15] Here we provide, for reference, tables that list the number of isomers of alkanes, alkenes, alkynes,[152] and stereoalkanes

(Table 4), ketones and esters (Table 5), and primary, secondary and tertiary alcohols (Table 6) up to 25 carbon atoms.

Table 4: Isomers of Alkanes, Alkenes, Alkynes, and Stereoalkanes

Carbon Atoms	Alkanes	Alkenes	Alkynes	Stereoalkanes
1	1			1
2	1	1	1	1
3	1	1	1	1
4	2	3	2	2
5	3	5	3	3
6	5	13	7	5
7	9	27	14	11
8	18	66	32	24
9	35	153	72	55
10	75	377	171	136
11	159	914	405	345
12	355	2281	989	900
13	802	5690	2426	2412
14	1858	14397	6045	6553
15	4347	36564	15167	18127
16	10359	93650	38422	50699
17	24894	240916	97925	143255
18	60523	623338	251275	408429
19	148284	1619346	648061	1173770
20	366319	4224993	1679869	3396844
21	910726	11062046	4372872	9892302
22	2278658	29062341	11428365	28972080
23	5731580	76581151	29972078	85289390
24	14490245	202365823	78859809	252260276
25	36797588	536113477	208094977	749329719

Table 5: Isomers of Ketones and Esters

Carbon Atoms	Ketone Isomers	Ester Isomers
1	1	0
2	1	1
3	2	2
4	3	4
5	7	9
6	14	20
7	32	45
8	72	105
9	171	249
10	405	599
11	989	1463

12	2426	3614
13	6045	9016
14	15167	22695
15	38422	57564
16	97925	146985
17	251275	377555
18	648061	974924
19	1679869	2529308
20	4372872	6589734
21	11428365	17234114
22	29972078	45228343
23	78859809	119069228
24	208094977	314368027
25	550603722	832193902

Table 6: Isomers of Alcohols Series

Carbon Atoms	Primary Alcohols	Secondary Alcohols	Tertiary Alcohols
1	1	0	0
2	1	0	0
3	1	1	0
4	2	1	1
5	4	3	1
6	8	6	3
7	17	15	7
8	39	33	17
9	89	82	40
10	211	194	102
11	507	482	249
12	1238	1188	631
13	3057	2988	1594
14	7639	7528	4074
15	19241	19181	10443
16	48865	49060	26981
17	124906	126369	69923
18	321198	326863	182158
19	830219	849650	476141
20	2156010	2216862	1249237
21	5622109	5806256	3287448
22	14715813	15256265	8677074
23	38649152	40210657	22962118
24	101821927	106273050	60915508
25	269010485	281593237	161962845

While the alkane and alkane-like substances are the most important, no series of compounds has received as much interest in generating isomer series as has the polyhexes, with much being done on various classes of benzenoids.[78, 153-155] For example, isomer series are available for benzenoids of a variety of classes, including peri-condensed,[156, 157] cata-condensed,[35, 158], essentially disconnected,[159] helicenes,[158, 160] resonant sextets,[161] quinones,[162] coronenes,[163, 164] and pyrenes.[165] Recently, with the aid of a new lattice enumeration algorithm the number of benzenoid hydrocarbons was calculated up to 35 hexagons.[79] We provide this information in Table 7.

Table 7: Benzenoids isomers as a function of the number of hexagons

Hexagons	Benzenoids	Hexagons	Benzenoids
1	1	18	8553649747
2	1	19	41892642772
3	3	20	205714411986
4	7	21	1012565172403
5	22	22	4994807695197
6	81	23	24687124900540
7	331	24	122238208783203
8	1435	25	606269126076178
9	6505	26	3011552839015720
10	30086	27	14980723113884739
11	141229	28	74618806326026588
12	669584	29	372132473810066270
13	3198256	30	1857997219686165624
14	15367577	31	9286641168851598974
15	74207910	32	46463218416521777176
16	359863778	33	232686119925419595108
17	1751594643	34	1166321030843201656301
		35	5851000265625801806530

Another class of polyhexes, which are fullerenes, has also been the subject of much interest in the field of isomer generation owing to the impact that this class of compounds has made in many areas of science and technology. While tables exist for the general and isolated-pentagon rule fullerenes,[166, 167] we provide in Table 8 an isomer list for these classes up to large number of vertices, courtesy of the Fullgen code.[94, 168]

Table 8: Fullerene isomers as a function of the number of Carbons

Number of atoms	Numbers of Fullerenes	Number of atoms	Numbers of Fullerenes
20	1	100	285914
22	0	102	341658
24	1	104	419013
26	1	106	497529
28	2	108	604217
30	3	110	713319
32	6	112	860161
34	6	114	1008444
36	15	116	1207119
38	17	118	1408553
40	40	120	1674171
42	45	122	1942929
44	89	124	2295721
46	116	126	2650866
48	199	128	3114236
50	271	130	3580637
52	437	132	4182071
54	580	134	4787715
56	924	136	5566948
58	1205	138	6344698
60	1812	140	7341204
62	2385	142	8339033
64	3465	144	9604410
66	4478	146	10867629
68	6332	148	12469092
70	8149	150	14059173
72	11190	152	16066024
74	14246	154	18060973
76	19151	156	20558765
78	24109	158	23037593
80	31924	160	26142839
82	39718	162	29202540
84	51592	164	33022572
86	63761	166	36798430
88	81738	168	41478338
90	99918	170	46088148
92	126409	172	51809018
94	153493	174	55817091
96	191839	176	64353257
98	231017		

While we have provided popular and useful tables in this section for a variety of substances, many other isomer tables exist in the literature. Several of these have been generated in an attempt to verify or compare codes that generate isomers. To best aid the reader, references to these listings are provided. Novak gives a small list of some halogen derivatives of a few molecules and ions and their chirality.[169] In a series of several papers, Contreras and co-workers have isomer tables on dozens of organic compounds, both cyclic and acyclic. Wieland et. al. have generated configurational and constitutional isomers for a variety of hydrocarbons up to 10 carbon atoms.[135] Dias provides a constant isomer series for fluorenoid and fluoranthenoid hydrocarbons.[170] Luinge generated dioxane isomers as well as isomer lists for a variety of CHO and CHN compounds.[171] CHO, CHON and CON isomer lists were generated by Molodtsov in 1994. [172] A subclass of indacenoids, namely di-5-catafusenes, were studied by Cyvin *et al.* and isomer lists generated.[173] These same authors later provided isomer lists for systems containing pentagons and heptagons, with both one pentagon (azulenoids)[174] and multiple pentagons.[175] Dolhaine and co workers have provided some tables and formulae for the number of isomers of a variety of substituted molecules including benzene, anthracene, and fullerenes.[176] Dolhaine and Honig have also published a large list of inositol oligomers up to the tetramer with estimates of larger isomers for larger oligomers provided.[177] Finally, Davidson provides alkyl frequency distributions in alkane isomers up to 21 carbon atoms.[178]

Before we leave this subsection, it is useful to note that a few studies comparing various isomer generation techniques have been published in an attempt to provide both a validation of an algorithm against a test case and comparison of a variety of methods in terms of consistency and execution time. Such studies may also contain isomer generation lists of the type mentioned in the previous paragraphs. To the reader interested in these tests, we provide references.[60, 171, 179-181]

Structure Elucidation

A very clear and important application of enumerating molecules falls within a larger framework of structure elucidation. In brief, the ultimate goal of structure elucidation is to take input information and identify *the* compound that is consistent with that information. A more pragmatic goal of this endeavor is to generate *all* candidate molecules consistent with the input information. While information before or after the candidate generation is used to focus the solution space to a single solution, such an ideal result is not often met and, thus, lists of solutions (perhaps ranked) are produced. The reason for this has to do with many factors including the combinatorial explosion of isomers, the quality of the input information, and the efficiencies of the algorithms that use this information. A much more detailed assessment of this situation is provided elsewhere and the reader is referred to those works.[110, 120, 182]

Our goal in this section is to *highlight* some of the popular codes that can be used to perform structural elucidation. Where applicable, information required on the types of input is provided. We must note that not all structural elucidation codes are equal. Some are considered expert systems containing large databases of initial (stored) information and use complex algorithms that attempt to reach the ultimate goal. Others are more modest isomer generation codes with some pre- or post-processing to include experimental information to assist with the arrival of a solution (or solution set) for a particular problem. Accordingly, we will describe some isomer generation codes first and finish with the expert systems.

A program to enumerate all possible saturated hydrocarbons was introduced in 1991 by Hendrickson and Parks.[183] This code, called SKEL_GEN, was tested for structures containing up to 11 carbon atoms but limited work was extended to larger ringed structures.

In 1992 Contreras introduced CAMGEC (**C**omputer-**A**ssisted **M**olecular **GE**neration and **C**ounting),[60] which is an exhaustive, selective and non-redundant structural generation C code under a Unix platform. The program requires just a molecular formula as input. No means to input other information or for post-processing exists. Recent improvements to CAMGEC[61-65] have been presented that improve efficiency and allow for stereoisomer generation.

To aid in the interpretation of infrared spectra, Luinge developed the structure generator AEGIS (**A**lgorithm for the **E**xhaustive **G**eneration of **I**rredundant **S**tructures).[171] While it is reportedly simple to use and requires only the molecular formula as input, it is written in the PROLOG language that is computationally expensive.

Though not designed to compete with other isomer generation codes, Barone and co-workers designed an exhaustive method to generate organic isomers from base 2 and base 4 numbers called GI (**G**eneration of **I**somers).[180] They have used GI in an attempt to check the consistency between other, much faster, isomer generation codes and have found some discrepancies.

A large, yet exhaustive isomer generation package called ISOGEN[184] produces an irredundant list of structure isomers consistent with a given empirical formula. Revisions to this code with the same name uses modified algorithms that include evolutionary approaches.[185]

Le Bret put forth a novel approach to the structural elucidation problem by using a genetic algorithm to exhaustively generate isomers. The program, called GalvaStructures,[181] uses the molecular formula as input and can take various spectral information to aid in the efficiency of solution. Though stochastic, the program seems to be consistent with other generators, yet it is much slower for large problems.

The "grandfather" of knowledge-based structural elucidation codes is the DENDRAL project at Stanford University.[56, 57] DENDRAL (**DENDR**itic **AL**gorithm) provided a recipe (plan, generation, test) to exhaustively enumerate all isomers given an input set of atoms and spectral information. The generation of structures was performed with CONGEN (**CON**strained **GEN**erator) and, ultimately, with a more-advanced structure generator called GENOA (**GEN**eration with **O**verlapping **A**toms).[113] The latter code added some automated features as well as a different way to handle overlapping substructural units.

ASSEMBLE 2.0[186] is a structure generator taking molecular formula and fragments as input. On output, candidates can be ranked based on fragment spectra given on input.

CHEMICS[111] is an automated structure elucidation system for organic compounds that uses 630 fragments in developing structures. Spectroscopic data in the form of IR, ^{1}H-NMR and ^{13}C-NMR as well as bond correlations are used to limit the candidate structures output.

EPIOS (**E**lucidation by **P**rogressive **I**ntersection of **O**rdered **S**ubstructures)[187] is a code that uses a database of ^{13}C NMR spectra and generates candidate structures through overlapping fragments.

The structure generator GEN uses up to 30 fragments (obtained from, say, spectral information) as input and can be given various types of constraints during generation such as molecular formula, molecular weight, and structural considerations. The code itself is used in two systems, GENSTR and GENMAS.[179] The GENSTR system is used when specific fragments can be selected and additional information introduced into the generation process. GENMAS is used when only molecular formula is to be input.

GENM a program written in both C and Fortran that generates of all nonisomorphic molecular graphs given a set of labeled vertices with a specific valence.[172] While the code lacks post-processing, forbidden and required fragments can be input to the program.

MOLGEN, developed from MOLGRAPH,[107] is a structure elucidation code that has made its way into the commercial market and is, perhaps, the most widely-known program of its type.[108, 188] Upon input of a molecular formula, MOLGEN produces a complete set of redundancy-free isomers. MOLGEN can be used online and provides the user with the number and structure of isomers corresponding to a given molecular formula. An online version can be found in the MATCH journal webpage.[136]

When StrucEluc was first introduced, it used only 1D-NMR data with other information for structural elucidation of molecules containing fewer than 25 atoms. A recent enhancement of StrucEluc now uses a variety of 2D NMR data as well as data from IR and mass spectra.[189] Such improvements have allowed this code to elucidate product molecules containing more than 60 atoms. This program now forms a suite of programs offered under the name ACD/StructureElucidator.

COCON[190] and a web-based version (WebCocon 4J) searches for compounds of known molecular formula compatible with 2D NMR data as input. This code also is able to interpret heteronuclear multi-bond correlations as 2, 3 and 4 bond connectivities.

SpecSolv[116] is a structure elucidation system based on ^{13}C chemical shifts with additional options to use NMR information of most any kind to aid in candidate refinement. Note that SpecSolv does not require an initial molecular formula as input.[191]

The ESESOC (**E**xpert **S**ystem for the **E**lucidation of the **S**tructures of **O**rganic **C**ompounds)[192, 193] system is used for structural elucidation and presents candidate structures consistent with a molecular formula and spectroscopic data. In addition to being a structure generator, the system can extract various information including IR, NMR and COSY as constraints.

Faulon developed a stochastic structure generator, named SIGNATURE,[118, 145] and used it for a variety of natural compounds structure elucidation problems. The SENECA structural elucidation system[146] later incorporated algorithms developed by Faulon with the goal of finding the constitution of a molecule given spectroscopic data, most notably NMR.

Cocoa[119] (**CO**nstrained **CO**mbination of **A**tom-centered fragments) is a structural elucidation method, which rather than combining fragments (based on input information) to make structures, uses the information to remove fragments. Such an approach makes the best use of all input information. Cocoa was later incorporated into a larger software, SESAMI, that includes a spectrum interpreter.[194]

The HOUDINI program, part of the SESAMI system, contains elements of both structure assembly (ASSSEMBLE) and structure reduction (Cocoa).[120]

The CISOC-SES system (**C**omputerized **I**nformation **S**ystem of **O**rganic **C**hemistry-**S**tructure **E**lucidation **S**ubsystem) is another structural elucidation program to generate candidate structures given NMR information.[195] The algorithms used in this system emphasize the use of long-range distance constraints.

Elyashberg and co-workers developed a structure elucidation system called X-PERT that uses molar mass, IR and NMR spectra in combination with a gradual growth of constraints in reaching candidate solutions.[196]

Some structure elucidation success stories.

All of the codes and systems listed above have, at some point, been tested to evaluate effectiveness. Once these approaches are reported in the literature and/or presented at a conference, the next step involves adoption of the system for a particular need. However, those who adopt these systems are normally scientists from companies whose work is proprietary. Hence, most of the "real world" successes of structure elucidation are not disseminated. However, there are cases where difficult structural elucidation problems have been solved and published and we provide a few interesting examples here.

As reported by Munk,[110] the earliest example of a real-world structure elucidation problem using computational techniques was performed in 1967 to determine the structure of antinobolin. Degradation reactions of antinobolin were performed leading to substructures from which an early version of ASSEMBLE generated six viable candidates. Subsequent spectroscopic studies were performed on these six and the correct structure was isolated. This procedure, in total, reportedly took several man-years to complete. By way of comparison, this problem was revisited recently using the SESAMI system with 1D and 2D NMR data derived from actinobolin acetate. SESAMI, in conjunction with some simple experiments and other data, resolved the correct structure with the entire process numbering in days as opposed to years.

Lignin is an important polymer found in the cell wall of plants and plays a key role in a variety of industries including pulp and paper as well as fuel and wood science. While many studies had been performed on lignin, a clear and compelling picture of the structure of this polymer corroborating existing experimental information had yet to be determined. Using the SIGNATURE program in conjunction with molecular simulation as well as NMR data and known fragments for lignin monomers, Faulon and Hatcher[197] concluded that a structure with a helical template for lignin was preferred over random structures. Such a conclusion was consistent with Raman spectroscopic information. More recent uses of SIGNATURE include the design of sample structures for humic substances[198] and asphalthenes.[199]

In another, recent example, the ACD/Structure Elucidator was used to resolve 2D NMR data on a C31 alkaloid that had several ambiguities in connectivity associated with spectral overlap. Twelve candidates were revealed of which eleven were ultimately ruled out for violating a variety of constraints. The new compound, named quindolinocrypto-tackieine, was thus solved.[200]

Combinatorial Library Design

Methods for synthesizing large combinatorial libraries of organic compounds emerged in the mid- 1990s[201, 202, 203] This revolutionized drug discovery, as millions of candidate compounds could then be synthesized in parallel and evaluated using high throughput screening techniques.[204, 205] However, given that the number of compounds that could be synthesized exceeds 10^{12} for even a simple combinatorial library scheme based only on commercially available reagents,[206] and estimated to be over 10^{30} in the whole accessible chemical space,[207] the effectiveness of these early "brute force" experimental approaches were necessarily quite limited. Therefore, along with the development of combinatorial chemistry came a growth in virtual chemistry and software tools to sift '*in silico*' through large numbers of potential compounds in combinatorial libraries to select the most promising subset for synthesis and experimental testing. The ability to enumerate molecules is a crucial step in many virtual chemistry algorithms for designing combinatorial libraries, which in turn are used to discover lead pharmaceutical compounds.

Many different approaches have been taken to designing these libraries. Diversity approaches[208, 209] are used to design general exploratory libraries that maximize chemical diversity for initial drug discovery. Biased approaches are used to design focused libraries when there is *a priori* knowledge of either the structure of the target (structure-based approaches[210-215]) or a small lead compound (similarity approaches[216]). Informative design,[217] a relatively new approach, designs a library that will provide the maximum amount of information from each experimental cycle of synthesis and testing. In addition to examining the potential drug-binding properties of the library, most library design efforts try to simultaneously maximize the ADME (adsorption, distribution, metabolism and toxicity) properties of the library members, using heuristics of "drug-likeness"[218-221] as well as other functions such as cost. The cost functions may be implemented as simple post-processing filters or as objective functions to maximize. Although library design methods can deal in the chemical space of the reactants, product-based design has been shown to be superior, albeit more expensive computationally.[222, 223] For a complete review of computational techniques applied to combinatorial libraries, see Lewis *et* *al.*[209] Most of the product-based approaches involve either full or partial enumeration of the products of the combinatorial library.

The basic chemistry for combinatorial synthesis usually involves a core group or scaffold, to which a set of reagents or R groups is systematically reacted at each substitution site (see Figure 21 for some typical scaffolds). As all combinations of reagents are synthesized, the total size of the combinatorial library can be estimated using Pólya's theorem as given by eq. 7 in the "Counting structures" subsection. For the majority of combinatorial libraries, the scaffold is asymmetric and the size of the library is simply the product of the number of possible reagents at each substitution site. For example, if a scaffold has three variable positions R1,R2, and R3, each with 1000 possible reagents that can react at that site, there are 1000^3 or 10^9 possible compounds that could be synthesized. Because there are often even more than 1000 reagents commercially available per reactant site, the numbers are often even greater.

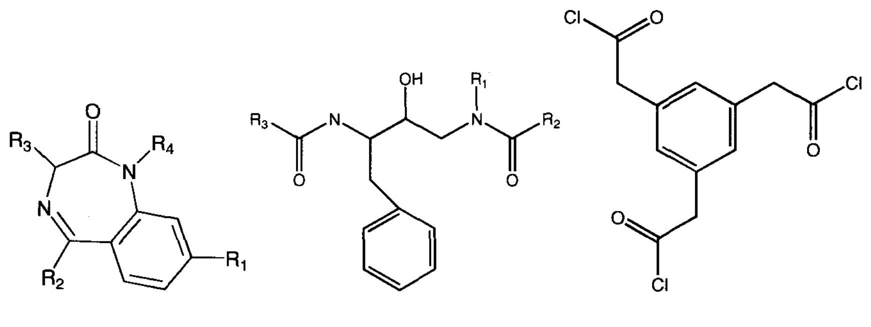


Figure 21. Benzodiazepine scaffold (left) and a statine-base peptomimetic (middle) are typical asymmetric scaffolds. Benzene triacylchloride scaffold (right) is a typical symmetric scaffold.

The first step in constructing a virtual combinatorial library for a given scaffold is to identify the pool of reagents available for each substitution site on the scaffold. This usually involves searching substructures in a database of commercially available and in-house compounds for those containing the appropriate reactivity for the synthesis protocol at each substitution site. Filters are used to eliminate reagents with inappropriate chemistries such as functional groups predicted to cause side reactions, or those that may interfere with or cause false positive tests in the biological assays, or those with insufficient 'drug-like' properties.

Once the reagents are selected the next step is to enumerate product compounds of the library. Most enumeration programs take into account only the simple case of an asymmetric scaffold, however MOLGEN-COMB[224] is specifically designed to handle symmetry. For the asymmetric scaffold case, there exist two basic approaches. The first is referred to as "fragment marking". Here, the reactant pools are treated with a pre-processing step where they are "marked" by removing the reacting functional group in each reagent and replacing it with a free valance. Enumeration consists of systematically placing all the clipped reagents onto the scaffold"[225, 226] using an algorithm similar to Scheme I given in the "Enumerating structures" subsection. A simple version of this approach has been used in the structure-based programs CombiDOCK[212] and CombiBUILD.[211] One problem with this approach is that it cannot handle all synthetic reaction types, such as the Diels-Alder reaction, or systems with no clear core scaffold such as oligomeric libraries of variable length.[227] The second approach is to use a 'reaction transform' to perform the same chemical transformations *in silico* that are being performed chemically. Advantages of this approach are that it can be used on all chemistries, does not involve any pre-processing of the reagents, and the transforms can be reused. It has the disadvantage however of being computationally more demanding and thus slower to perform. Many programs use the SMARTS molecular query notation from the Daylight toolkit[228] to design reaction transformation tools,[229] an approach that has been incorporated into the ADEPT program.[227] Commercially available programs that perform computational enumeration include

CombiLibMaker in Sybyl[122], Analog Builder in Cerius^2,[123] PRO_SELECT,[215] and the QuaSAR-CombiGen module in MOE.[230]

As full enumeration of very large combinatorial libraries is impractical, several methods have been developed to avoid explicit enumeration of all library members. Many diversity- and similarity-based design strategies use sampling approaches such as genetic algorithms,[149, 231, 232] simulated annealing,[233, 234] and stochastic sampling[235] to optimize libraries (see Gillet *et al.* [236] for a full review). Some of these approaches use descriptors that can estimate the properties of library members without explicit enumeration of the full library, either by using descriptors that can be calculated roughly from the sum of the reactants,[206, 225, 226] or by using a neural net to estimate properties from a small sampling of enumerated products.[229] Combining multiple approaches can also reduce the problem to a computationally tractable number of possible solutions. For example, a diversity search can be performed to select a smaller library that can then be explicitly enumerated as a starting point for a structure-based library design of a focused library.[237] Enumeration can also be reduced in structure-based programs, which start with the three-dimensional structure of the target, by taking a 'divide-and-conquer' strategy.[211, 212, 215] In a divide-and –conquere scheme the scaffold is first docked to the binding site. The reagents are then evaluated *individually* at each substitution site for predicted binding, thus turning the problem from n^r enumerations to r x n, where n is number of reagents and r is the number of substitution sites. Only top-scoring reagents are saved for full compound enumeration and evaluation. Virtual libraries designed in this manner have rapidly led to potent lead compounds. [210, 238]

Molecular Design with Inverse-QSAR

The forward quantitative structure activity relationship (QSAR) procedure defines an equation or a set of equations that relates a variable of interest (dependent variable) in terms of independent variables. The dependent variable is normally an activity/property of interest (binding affinity, normal boiling point, IC_{50}, etc.) while the independent variables are related to the structure of the substance. Developping a QSAR for a particular activity/property involves training the parameters of the model against a well-defined set of data (training set), with a small portion of the data held back for validation of the model (test set). Once the QSAR is effectively trained and validated, one can use this model to predict the activity/property value of a given compound by determining the values of its independent variables in a straight-forward manner. On the other hand, rather than determining an activity/property value for a particular compound from the QSAR, what if one wants to determine a compound from the QSAR given a *particular activity/property value*? This question is known as the inverse-QSAR problem (I-QSAR) and is the subject of this section.

Anyone reading this chapter has, undoubtedly, solved an inverse-type problem in one form or another. The key to efficient solution lies in the restriction of the solution space. If constraints are composed such that the solution space is limited, a brute-force technique (try all candidates) can guarantee a solution. In the field of molecular design, however, the solution space comes from all compounds that can be reasonably made from the various atoms in the Periodic Table. Hence, one needs a way to limit this solution space to arrive at candidate solutions efficiently. We describe some of these techniques below.

As mentioned earlier, Kier and Hall published a series of papers in the early 1990's that described the inverse QSAR methodology using chi indices. The QSARs they developed had a maximum of four descriptors. Example applications included the inverse design of alkanes from molar volume,[122] and the identification of isonarcotic agents.[239]

Simultaneous with the work from Kier and Hall, Zefirov and co-workers,[127] developed a similar technique using the count of paths. The QSARs they used were given in terms of three Kappa-shape descriptors and they considered three functional groups, namely alkanes, alcohols and small oxygen-containing compounds.

In 2001 Bruggemann et. al.[240] demonstrated the use of Hasse diagrams combined with a similarity measure in the generation of solutions to the inverse problem involving toxicity of algae. Their method is based in partial ordered sets and does not assume a particular model for the QSAR.

Garg and Achenie also demonstrated a reasonable approach to the solution of the I-QSAR problem in 2001.[241] Taking a target scaffold of an antifolate molecule for dihydrofolate reductase inhibition, these authors generated a QSAR for both activity and selectivity. They solved the I-QSAR problem to maximize selectivity through changing substitutents on the scaffold, subject to a constraint of a threshold activity. Finally, a work by Skvortsova, *et al.*[242] from 2003 demonstrated that the I-QSAR problem could be solved for the Hosoya index plus constraints on the number of carbon atoms for a system of 78 hydrocarbons.

All of the previous methods have limitations. As has been demonstrated above, one can limit the problem size by working with a QSAR derived with only a few descriptors. Hence, many solutions can be found associated with the given problem. Additionally, one can limit the solution space to contain, say, only hydrocarbons or alcohols. A third issue on the approaches described above concerns the degeneracy of the solution themselves. It is not uncommon for a particular value of a topological index to correspond to a large number of possible compounds.

A novel inverse-QSAR methodology has been developed recently that addresses these issues and will be described briefly next.

The Signature molecular descriptor, previously mentioned in this chapter for structural elucidation, has found utility in the solution of the inverse-QSAR problem. The reason for this is that Signature can produce meaningful QSARs[129, 243] and is the least degenerate of dozens of topological indices tested.[129] Additionally, Signature lends itself to the inversion process.[128] An algorithm that will enumerate and sample chemical structures corresponding to the numerical solutions from the I-QSAR problem has already been developed and tested for a variety of compounds including alkanes, fullerenes, and HIV-1 protease inhibitors.[128]

The inverse-QSAR problem using Signature has also been applied to a small set of LFA-1/ICAM-1 peptide inhibitors to assist in the search and design of more-potent inhibitory compounds. After developing a QSAR, the inverse-QSAR technique with Signature generated many novel inhibitors. Two of the more potent inhibitors were synthesized and tested in-vivo, confirming them to be the strongest inhibiting peptides to date.[244]

Conclusion and future directions

We have seen in this chapter that counting, enumerating, and sampling of molecular graphs from a molecular formula are not the technical challenges they once were. Counting formulae exist for a large variety of chemical compounds, isomer generators can enumerate without construction, or count, up to 10^{30} molecular graphs,[87] and sampling molecules can theoretically be performed efficiently.[245] Nonetheless, the computational complexity of counting and enumerating molecular graphs remains an unsolved problem. It is thus expected that research collaboration between mathematics, computer science, and computational chemistry will continue to devise better techniques to count and enumerate molecules.

As far as structure elucidation and molecular design are concerned, enumerating molecules from a molecular formula is only part of the problem. Indeed, as we have argued in this chapter, enumerating structures with constraints, such as including the presence or absence of overlapping fragments, is most probably an intractable problem. Alternative stochastic sampling approaches have been devised recently to overcome the difficulties of enumerating molecules with constraints. Only a few stochastic techniques have so far been published and it is likely that the sampling approach will continue to be developed and used in the near future for practical purposes such as elucidation from NMR spectra.

Even if we knew how to efficiently enumerate or sample molecular graphs under constraints, our job would not be completed. Molecules are 3D objects and, ultimately, structure generators should produced 3D structures. Enumerating stereoisomers alone is not sufficient as we also need to generate the structural conformations corresponding to the problem constraints. The natural solution that comes to mind is to first enumerate all molecular graphs matching the constraints and then to explore the conformational space of each graph. While codes exist for constructing 3D representations of molecular graphs,[246-249] exploring the conformational space of each molecular graph is a cumbersome task that must be added to the already costly endeavor of structure enumeration. Such a strategy does not appear to be computationally feasible. One alternative to avoid that computational bottleneck may be to use the geometrical enumeration we have seen for benzenoids.[76, 79] Recall that this approach consists of enumerating self-avoiding polygons on lattices. The enumeration is performed directly on a 2D lattice space (benzenoids are planar) ignoring the underlying molecular graphs. The advantage of the geometrical approach is that energetically unfavorable structures are never constructed. Such is obviously not the case when enumerating dimensionless molecular graphs. Considering that geometrical enumeration is currently the most powerful technique to enumerate benzenoids, such a promising approach may be further explored, perhaps, for structure elucidation and molecular design purposes.

References

1. A. Cayley, *Rep. Brit. Ass. Adv. Sci.,* **14**, 257 (1875). On the Analytical Forms Called Trees with Applications to the Theory of Chemical Compounds.
2. G. Polya, *C. R. Acad. Sci. Paris,* **201**, 1167 (1935). Un Probleme Combinatoire General Sur Les Groupes Des Permutations Et Le Calcul Du Nombre Des Composes Organiques.
3. D. H. Rouvray, *J. Mol. Struct. (THEOCHEM),* **54**, 1 (1989). The Pioneering Contributions of Cayley and Sylvester to the Mathematical Description of Chemical Structure.
4. S. J. Russell, and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, NJ, 2002.
5. P. W. Fowler, and P. Hansen, in *DIMAC Workshop Report*, Rutger University Press, 2001. The Working Group on Computer-Generated Conjectures from Graph Theoretic and Chemical Databases I.

6. A. R. Leach, in *Reviews in Computational Chemistry*, K. B. Lipkowitz and D. B. Boyd, Eds., VCH Publishers, New York, 1991, Vol. 2, pp. 1-55. A Survey of Methods for Searching the Conformational Space of Small and Medium-Sized Molecules.

7. H. A. Sheraga, in *Reviews in Computational Chemistry*, K. B. Lipkowitz and D. B. Boyd, Eds., VCH Publishers, New York, 1992, Vol. 3, pp. 73-142. Predicting Three-Dimensional Structures of Oligopeptides.

8. Y. Kudo, and S.-I. Sasaki, *J. Chem. Doc.,* **14**, 200 (1974). The Connectivity Stack, a New Format for Representation of Organic Chemical Structures.

9. J.-L. Faulon, *J. Chem. Inf. Comput. Sci.,* **38**, 432 (1998). Isomorphism, Automorphism Partitioning, and Canonical Labeling Can Be Solved in Polynomial-Time for Molecular Graphs.

10. B. D. Mckay, Nauty User's Guide, Version 2.2., http://cs.anu.edu.au/people/bdm/nauty/

11. E. M. Luck, *J. Comput. Sys. Sci.,* **25**, 42 (1982). Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time.

12. M. R. Garey, and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of Np-Completeness*, W. H. Freeman & Company, New York, NY, 1979.

13. A. Cayley, *Philos. Mag.,* **13**, 172 (1857). On the Analytical Forms Called Trees.

14. F. Herman, *Ber. Dtsch. Chem. Ges.,* **13**, 792 (1880). On the Problem of Evaluating the Number of Isomeric Paraffins of the Formula C_nH_{2n+2}.

15. H. R. Henze, and C. Blair, *J. Am. Chem. Soc.,* **53**, 3077 (1931). The Number of Isomeric Hydrocarbons of the Methane Series.

16. G. Polya, *Acta Math.,* **68**, 145 (1937). Kombinatorische Anzahlbestimmungen Fur Gruppen, Graphen Und Chemische Verbindungen.

17. F. Harary, *Trans. Amer. Math. Soc.*, **78**, 445 (1955). The Number of Linear, Directed, Rooted, and Connected Graphs.

18. F. Zhang, R. Li, and G. Lin, *J. Mol. Struct. (THEOCHEM)*, **453**, 1 (1998). The Enumeration of Heterofullerenes.

19. H. Fripertinger, *MATCH*, **33**, 121 (1996). The Cycle Index of the Symmetry Group of the Fullerene C60.

20. P. W. Fowler, D. B. Redmond, and J. P. B. Sandall, *J. Chem. Soc. Faraday Trans.*, **19**, 2883 (1998). Enumeration of Fullerene Derivatives C70xm of Given Symmetries.

21. R. M. Nembra, and A. T. Balaban, *J. Chem. Inf. Comput. Sci.*, **38**, 1145 (1998). Algorithm for the Direct Enumeration of Chiral and Achiral Skeleton of a Homosubstituted Derivative of a Monocyclic Cycloalkane with a Large and Factorizable Ring Size N.

22. I. Baraldi, and D. Vanossi, *J. Chem. Inf. Comput. Sci.*, **40**, 386 (2000). Regarding Enumeration of Molecular Isomers.

23. R. C. Read, *J. London Math. Soc.*, **35**, 344 (1960). The Enumeration of Locally Restricted Graphs II.

24. R. Otter, *Annals Math.*, **49**, 583 (1948). The Number of Trees.

25. J. Wang, R. Li, and S. Wang, *J. Math. Chem.*, **33**, 171 (2003). Enumeration of Isomers of Acyclic Saturated Hydroxyl Ethers.

26. S. Fujita, *Symmetry and Combinatorial Enumeration in Chemistry*, Springer-Verlag, Berlin, 1992.

27. S. J. Cyvin, B. N. Cyvin, J. Brunvoll, and J. Wang, *J. Mol. Struct. (THEOCHEM),* **445**, 127 (1998). Enumeration of Staggered Conformers of Alkanes and Monocyclic Cycloalkanes.

28. C. Y. Yeh, *J. Chem. Inf. Comput. Sci.,* **35**, 912 (1995). Isomer Enumeration of Alkanes, Labeled Alkanes, and Monosubstituted Alkanes.

29. C. Y. Yeh, *J. Phys. Chem.,* **100**, 15800 (1996). Theory of Acyclic Chemical Networks and Enumeration of Polyenoids Via Two-Dimensional Chirality.

30. C. Y. Yeh, *J. Chem. Inf. Comput. Sci.,* **36**, 854 (1996). Isomer Enumeration of Alkenes, and Aliphatic Cyclopropoane Derivatices.

31. C. Y. Yeh, *J. Chem. Phys.,* **105**, 9706 (1996). Counting Linear Polyenes by Excluding Structures with Steric Strain.

32. C. Y. Yeh, *J. Mol. Struct. (THEOCHEM),* **432**, 153 (1996). Isomerism of Asymmetric Dendrimers and Stereoisomerism of Alkanes.

33. L. Bytautas, and D. J. Klein, *J. Chem. Inf. Comput. Sci.,* **38**, 1063 (1998). Chemical Combinatorics for Alkane-Isomer Enumeration and More.

34. S. J. Cyvin, and I. Gutman, *Kekule Structures in Benzenoid Hydrocarbons*, Springer-Verlag, Berlin, 1988.

35. I. Gutman, and S. J. Cyvin, *Introduction to the Theory of Benzenoid Hydrocarbons*, Springer-Verlag, Berlin, 1989.

36. I. Gutman, and S. J. Cyvin, *Advances in the Theory of Benzenoid Hydrocarbons*, Springer-Verlag, Berlin, 1990.

37. I. Gutman, S. J. Cyvin, and J. Brunvoll, *Advances in the Theory of Benzenoid Hydrocarbons II,* Springer-Verlag, Berlin, 1992.

38. J. R. Dias, *Handbook of Polycyclic Hydrocarbons: Part A, Benzenoid Hydrocarbons*, Elsevier, Amsterdam, 1987.

39. J. R. Dias, *Handbook of Polycyclic Hydrocarbons: Part B: Polycyclic Isomers and Heteroatom Analogs of Benzenoid Hydrocarbons*, Elsevier, Amsterdam, 1988.

40. N. Trinajstic, *Chemical Graph Theory*, CRC Press, Boca Raton, 1992.

41. A. T. Balaban, and F. Harary, *Tetrahedron,* **24**, 2505 (1968). Enumeration and Proposed Nomenclature of Benzenoid Cata-Condensed Polycyclic Aromatic Hydrocarbons.

42. F. Harary, and R. C. Read, *Proc. Edinburgh Math. Soc., Ser. II,* **17**, 1 (1970). Enumeration of Tree-Like Polyhexes.

43. S. J. Cyvin, and J. Brunvoll, *J. Math. Chem.,* **9**, 33 (1992). Generating Functions for the Haray-Read Numbers Classified According to Symmetry.

44. S. J. Cyvin, J. Brunvoll, and B. N. Cyvin, *J. Math. Chem.,* **9**, 19 (1992). Harary-Read Numbers for Catafusenes: Complete Classification According to Symmetry.

45. J. Brunvoll, S. J. Cyvin, and B. N. Cyvin, *J. Math. Chem.,* **21**, 193 (1997). Enumeration of Tree-Like Octagonal Systems.

46. S. J. Cyvin, F. Zhang, and J. Brunvoll, *J. Math. Chem.,* **3**, 103 (1992). Eumeration of Perifusenes with One Internal Vertex - a Complete Mathamatical Solution.

47. S. J. Cyvin, F. Zhang, B. N. Cyvin, G. Xiaofeng, and J. Brunvoll, *J. Chem. Inf. Comput. Sci.,* **32**, 532 (1992). Eumeration and Classification of Benzenoid Systems. 32. Normal Perifusenes with Two Internal Vertices.

48. T. Akutsu, S. Miyano, and S. Kuhara, *Bioinformatics,* **16**, 727 (2000). Inferring Qualitative Relations in Genetic Networks and Metabolic Pathways.

49. F. Harary, and E. M. Palmer, *Graphical Enumeration*, Academic Press, New York, 1973.

50. R. C. Read, *Annals of Discrete Math.,* **2**, 107 (1978). Every One a Winer, or How to Avoid Isomorphism Search When Cataloguing Combinatorial Configurations.

51. I. A. Faradzev, in *Problemes Combinatoires et Theorie des Graphes*, University of Paris, Orsay, 1978, pp.131-135. Constructive Enumeration of Combinatorial Objects.

52. B. D. McKay, *J. of Algorithms,* **26**, 306 (1998). Isomorph-Free Exhaustive Generation.

53. L. A. Goldberg, *J. of Algorithms,* **13**, 128 (1992). Efficient Algorithms for Listing Unlabeled Graphs.

54. J. Lederberg, in *The Mathematical Science*, R. Cosrims, Ed., MIT Press, Cambridge,, 1969, pp. 37-51. Topology of Molecules.

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

56. R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, *Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project*, McGraw-Hill, New York, 1980.

57. N. A. B. Gray, *Computer-Assisted Structure Elucidation*, John Wiley & Sons, New York, 1986.

58. J. V. Knop, W. R. Muller, Z. Jericevi, and N. Trinajsticc, *J. Chem. Inf. Comput. Sci.,* **21**, 91 (1981). Computer Enumeration and Generation of Trees and Rooted Trees.

59. J. E. Hopcroft, and R. E. Tarjan, in *Complexity of Computer Computation*, R. Miller and E. Thatcher, Eds., Plenum, New York, 1972, pp. 131-152. Isomorphism of Planar Graphs.

60. M. L. Contreras, R. Valdivia, and R. Rozas, *J. Chem. Inf. Comput. Sci.,* **32**, 323 (1992). Exhaustive Generation of Organic Isomers. 1. Acyclic Structures.

61. M. L. Contreras, R. Valdivia, and R. Rozas, *J. Chem. Inf. Comput. Sci.,* **32**, 483 (1992). Exhaustive Generation of Organic Isomers. 2. Cyclic Structures.

62. M. L. Contreras, R. Rozas, and R. Valdivia, *J. Chem. Inf. Comput. Sci.,* **34**, 610 (1994). Exhaustive Generation of Organic Isomers. 3. Acyclic, Cyclic and Mixed Compounds.

63. M. L. Contreras, R. Rozas, R. Valdivia, and R. Aguero, *J. Chem. Inf. Comput. Sci.,* **35**, 752 (1994). Exhaustive Generation of Organic Isomers. 4. Acyclic Stereoisomers with One or More Chiral Carbon Atoms.

64. M. L. Contreras, G. M. Trevisiol, J. Alvarez, G. Arias, and R. Rozas, *J. Chem. Inf. Comput. Sci.,* **35**, 475 (1999). Exhaustive Generation of Organic Isomers. 5. Unsaturated Optical and Geometrical Stereoisomers and a New CIP Subrule.

65. M. L. Contreras, J. Alvarez, M. Riveros, G. Arias, and R. Rozas, *J. Chem. Inf. Comput. Sci.,* **41**, 964 (2001). Exhaustive Generation of Ogranic Isomers. 6. Stereoisomers Having Isolated and Spiro Cycles and New Extended N_Tuples.

66. I. Lukovits, *J. Chem. Inf. Comput. Sci.,* **39**, 563 (1999). Isomer Generation: Syntactic Rules for Detection of Isomorphism.

67. I. Lukovits, *J. Chem. Inf. Comput. Sci.,* **40**, 361 (2000). Isomer Generation: Semantic Rules for Detection of Isomorphism.

68. K. Balasubramanian, J. J. Kaufman, W. S. Koski, and A. T. Balaban, *J. Comput. Chem.,* **1**, 149 (1980). Graph Theoretical Characterization Computer Generation of Certain Carcinogenic Benzenoid Hydrocarbons and Identification of Bay Regions.

69. J. V. Knop, K. Szymanski, Z. Jericevi, and N. Trinajsticc, *J. Comput. Chem.,* **4**, 23 (1983). Computer Enumeration and Generation of Benzenoid Hydrocarbons and Identification of Bay Regions.

70. I. Stojmenovi, R. Toi, and R. Doroslovacki, *Proceedings of the Sixth Yougoslav Seminar on Graph Theory*, (1985). Generating and Counting Hexagonal Systems In Graph Theory.

71. W. J. He, W. C. He, Q. X. Wang, J. Brunvoll, and S. J. Cyvin, *Naturforsch.,* **43a**, 693 (1988). Supplement to Enumeration of Benzenoid and Coronoid Hydrocarbons.

72. S. Nikolic, N. Trinajsticc, J. V. Knop, W. R. Müller, and K. Szymanski, *J. Math. Chem.,* **4**, 357 (1990). On the Concept of the Weighted Spanning Tree of Dualist.

73. W. R. Müller, K. Szymanski, and J. V. Knop, *Croat. Chem. Acta,* **62**, 481 (1989). On Counting Polyhex Hydrocarbons.

74. W. R. Müller, K. Szymanski, J. V. Knop, S. Nikoli, and N. Trinajstic, *J. Comput. Chem.,* **11**, 223 (1990). On the Enumeration and Generation of Polyhex Hydrocarbons.

75. J. V. Knop, W. R. Müller, K. Szymanski, and N. Trinajstic, *J. Chem. Inf. Comput. Sci.,* **30**, 159 (1990). Use of Small Computers for Large Computations: Enumeration of Polyhex Hydrocarbons.

76. R. Tosic, D. Masulovic, I. Stojmenovi, J. Brunvoll, S. J. Cyvin, and B. N. Cyvin, *J. Chem. Inf. Comput. Sci.,* **35**, 181 (1995). Enumeration of Polyhex Hydrocarbons to H = 17.

77. G. Caporossi, and P. Hansen, *J. Chem. Inf. Comput. Sci.,* **38**, 610 (1998). Enumeration of Polyhex Hydrocarbons to H = 21.

78. G. Brinkmann, G. Caporossi, and P. Hansen, *Commun. Math. Chem. (MATCH),* **43**, 133 (2001). Numbers of Benzenoids and Fusenes.

79. M. Voge, A. J. Guttmann, and I. Jensen, *J. Chem. Inf. Comput. Sci.,* **42**, 456 (2002). On the Number of Benzenoid Hydrocarbons.

80. I. G. Enting, and A. J. Guttmann, *J. Phys. A,* **22**, 1371 (1989). Polygons on the Honeycomb Lattice.

81. J. de Vries, *Rendiconti Circolo Mat. Palermo,* **5**, 221 (1891). Sur Les Configurations Planes Dont Chaque Point Supporte Des Droites.

82. A. T. Balaban, *Revue Rounaine de Chimie,* **12**, 103 (1967). Valence-Isomerism of Cyclopolyenes (Erratum).

83. F. C. Bussemaker, S. Cobeljic, D. M. Cvetkovic, and J. J. Seidel, *J. Combin. Theory Ser. B.,* **23**, 234 (1977). Cubic Graphs on 14 Vertices.

84. B. D. McKay, and G. F. Royle, *Ars Combinatoria,* **21a**, 129 (1986). Constructing the Cubic Graphs on up to 20 Vertices.

85. G. Brinkmann, *J. Graph Theory,* **23**, 139 (1996). Fast Generation of Cubic Graphs.

86. M. Meringer, *J. Graph Theory,* **30**, 137 (1999). Fast Generation of Regular Graphs and Construction of Cages.

87. T. Grüner, R. Laue, and M. Meringer, in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Rutger University Press, 1997, pp.113-122. Algorithms for Group Actions: Homomorphism Principle and Orderly Generation Applied to Graphs.

88. X. Liu, and D. J. Klein, *J. Comput. Chem.,* **12**, 1265 (1991). Sixty-Atom Carbon Cages.

89. C.-H. Sah, *Croatica Chemica Acta,* **66**, 105 (1993). Combinatorial Construction of Fullerene Structures.

90. D. J. Klein, and X. Liu, *International Journal of Quantum Chemistry. Quantum Chemistry Symposium,* **28**, 501 (1994). Elemental Carbon Isomerism.

91. D. E. Manolopoulos, and P. W. Fowler, *Chem. Phys. Lett.,* **204**, 1 (1993). A Fullerene without a Spriral.

92. P. W. Fowler, T. Pisanski, A. Graovac, and J. Zerovnik, in *Discrete Mathematical Chemistry*, P. Hansen, P. W. Fowler and M. Zheng, Eds., American Mathematical Society, 2000, Vol. 51, pp. 175-188. A Generalized Ring Spiral Algorithm for Coding Fullerenes and Other Cubic Polyhedra.

93. P. W. Fowler, and D. E. Manolopoulos, *An Atlas of Fullerenes*, Oxford Univ. Press, Oxford, 1995.

94. G. Brinkmann, and A. W. Dress, *Journal of Algorithms,* **23**, 345 (1997). A Constructive Enumeration of Fullerenes.

95. G. Brinkmann, A. W. Dress, S. W. Perrey, and J. Stove, *Mathematical Programming,* **79**, 71 (1997). Two Applications of the Divide & Conquer Principle in the Molecular Sciences.

96. G. Brinkmann, and A. W. Dress, *Advances Applied Math.,* **21**, 473 (1998). Penthex Puzzles. A Reliable and Efficient Top-Down Approach to Fullerene-Structure Enumeration.

97. E. C. Kirby, and P. Pollack, *J. Chem. Inf. Comput. Sci.,* **38**, 66 (1998). How to Enumerate the Connectional Isomers of a Toroidal Polyhex Fullerene.

98. L. M. Masinter, N. S. Sridharan, J. Lederberg, and D. H. Smith, *J. Am. Chem. Soc.,* **96**, 7702 (1974). Applications of Artificial Intelligence for Chemical Inference. XII. Exhaustive Generation of Cyclic and Acyclic Isomers.

99. Y. Kudo, and S.-I. Sasaki, *J. Chem. Inf. Comput. Sci.,* **16**, 43 (1976). Principle for Exhaustive Enumeration of Unique Structures Consistent with Structural Information.

100. C. A. Shelley, T. R. Hays, M. E. Munk, and R. V. Roman, *Analytica Chimica Acta,* **103**, 121 (1978). An Approach to Automated Partial Structure Expansion.

101. I. P. Bangov, *J. Chem. Inf. Comput. Sci.,* **30**, 277 (1990). Computer-Assited Structure Generation For a Gross Formula. 3. Alleviation of the Combinatorial Problem.

102. J.-L. Faulon, *J. Chem. Inf. Comput. Sci.,* **32**, 338 (1992). On Using Graph-Equivalent Classes for the Structure Elucidation of Large Molecules.

103. V. Kvasnicka, and J. Pospichal, *J. Chem. Inf. Comput. Sci.,* **30**, 99 (1990). Canonical Indexing and Constructive Enumeration of Molecular Graphs.

104. V. Kvasnicka, and J. Pospichal, *J. Chem. Inf. Comput. Sci.,* **36**, 516 (1996). Simulated Annealing Construction of Molecular Graphs with Required Properties.

105. M. S. Molchanova, and N. S. Zefirov, *J. Chem. Inf. Comput. Sci.,* **38**, 8 (1998). Irredundant Generation of Isomeric Molecular Structures with Some Known Fragments.

106. M. S. Molchanova, V. V. Shcherbukhin, and N. S. Zefirov, *J. Chem. Inf. Comput. Sci.,* **36**, 888 (1996). Computer Generation of Molecular Structures by the Smog Program.

107. A. Kerber, R. Laue, and D. A. Moser, *Analytica Chimica Acta,* **235**, 2973 (1990). Structure Generator for Molecular Graphs.

108. C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, and T. Wieland, *Analytica Chemica Acta*, **141** (1995). Molgen+, a Generator of Connectivity Isomers and Stereoisomers for Molecular Structure Elucidation.

109. S. Bohanec, and J. Zupan, *J. Chem. Inf. Comput. Sci.,* **31**, 531 (1991). Structure Generation of Constitutional Isomers from Structural Fragments.

110. M. E. Munk, *J. Chem. Inf. Comput. Sci.,* **38**, 997 (1998). Computer-Based Structure Determination: Then and Now.

111. K. Funatsu, N. Miyabayashi, and S. I. Sasaki, *J. Chem. Inf. Comput. Sci.,* **28**, 18 (1988). Futher Developments of Structure Generation in the Automated Structure Elucidation System Chemics.

112. S. G. Molodtsov, *Commun. Math. Chem. (MATCH),* **30**, 203 (1994). Generation of Molecular Graphs with a Given Set of Nonoverlapping Fragments.

113. R. E. Carhart, D. H. Smith, N. A. B. Gray, J. G. Nourse, and C. Djerassi, *J. Org. Chem.,* **46**, 1708 (1981). Genoa: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures.

114. J. E. Dubois, M. Carabedian, and R. Ancian, *C. R. Acad. Sci. (Paris),* **290**, 369 (1980). Automatic Structural Elucidation by C-13 NMR - DARC-EPIOS Method - Search for a Discriminant Chemical Structure Displacement Relationship.

115. J. E. Dubois, M. Carabedian, and R. Ancian, *C. R. Acad. Sci. (Paris),* **290**, 369 (1980). Automatic Structural Elucidation by C-13 NMR - Darc-Epios Method - Description of Progressive Elucidation by Ordered Intersection of Substructures.

116. M. Will, W. Fachinger, and J. R. Richert, *J. Chem. Inf. Comput. Sci.,* **36**, 221 (1996). Fully Automated Structure Elucidation - a Spectroscopist's Dream Comes True.

117. A. Schrijver, *Integer Linear and Integer Programming*, Wiley & Sons, New York, 1986.

118. J.-L. Faulon, *J. Chem. Inf. Comput. Sci.,* **34**, 1204 (1994). Stochastic Generator of Chemical Structure. 1. Application to the Structure Elucidation of Large Molecules.

119. B. D. Christie, and M. E. Munk, *J. Chem. Inf. Comput. Sci.,* **28**, 87 (1988). Structure Generation by Reudction: A New Strategy for Computer-Assisted Structure Elucidation.

120. A. Korytko, K. P. Schulz, M. Madison, and M. E. Munk, *J. Chem. Inf. Comput. Sci.,* **43**, 1434 (2003). Houdini: A New Approach to Computer-Based Structure Generation.

121. K. P. Schulz, A. Korytko, and M. E. Munk, *J. Chem. Inf. Comput. Sci.,* **42**, 1447 (2003). Applications of a Houdini-Based Structure Elucidation System.

122. L. B. Kier, L. H. Hall, and J. W. Frazer, *J. Chem. Inf. Comput. Sci.,* **33**, 143 (1993). Design of Molecules from Quantitative Structure-Activity Relationship Models. 1. Information Transfer between Path and Vertex Degree Counts.

123. L. H. Hall, L. B. Kier, and J. W. Frazer, *J. Chem. Inf. Comput. Sci.,* **33**, 148 (1993). Design of Molecules from Quantitative Structure-Activity Relationship Models. 2. Derivation and Proof of Information Transfert Relating Equations.

124. L. H. Hall, R. S. Dailey, and L. B. Kier, *J. Chem. Inf. Comput. Sci.,* **33**, 598 (1993). Design of Molecules from Quantitative Structure-Activity Relationship Models. 3. Role of Higher Order Path Counts: Path 3.

125. L. B. Kier, and L. H. Hall, *Quant. Struct.-Act. Relat.,* **12**, 383 (1994). The Generation of Molecular Structures from a Graph-Based QSAR Equation.

126. L. H. Hall, and J. B. Fisk, *J. Chem. Inf. Comput. Sci.,* **34**, 1184 (1994). Degree Set Generation for Chemical Graphs.

127. M. I. Skvortsova, I. I. Baskin, O. L. Slovokhotova, V. A. Palyulin, and N. S. Zefirov, *J. Chem. Inf. Comput. Sci.,* **33**, 630 (1993). Inverse Problem in QSAR/QSPR Studies for the Case of Topological Indices Characterizing Molecular Shape (Kier Indices).

128. J.-L. Faulon, C. J. Churchwell, and D. P. Visco Jr., *J. Chem. Inf. Comput. Sci.,* **43**, 721 (2003). The Signature Molecular Descriptor. 2. Enumerating Molecules from Their Extended Valence Sequences.

129. J.-L. Faulon, D. P. Visco Jr., and R. S. Pophale, *J. Chem. Inf. Comput. Sci.,* **43**, 707 (2003). The Signature Molecular Descriptor. 1. Extended Valence Sequences and Topological Indices.

130. J. G. Nourse, *J. Am. Chem. Soc.,* **101**, 1210 (1979). The Configuration Symmetry Group and Its Application to Stereoisomer Generation, Specification, and Enumeration.

131. J. G. Nourse, R. E. Carhart, D. H. Smith, and C. Djerassi, *J. Am. Chem. Soc.,* **101**, 1216 (1979). Exhaustive Generation of Stereoisomers for Structure Elucidation.

132. H. Abe, H. Hayasaka, Y. Miyashita, and S. I. Sasaki, *J. Chem. Inf. Comput. Sci.,* **24**, 216 (1984). Generation of Stereoisomeric Structures Using Topological Information Alone.

133. T. Wieland, *J. Chem. Inf. Comput. Sci.,* **35**, 220 (1995). Enumeration, Generation, and Construction of Stereoisomers of High-Valence Stereocenters.

134. L. A. Zaltina, and M. E. Elyashberg, *Commun. Math. Chem. (MATCH),* **27**, 191 (1992). Generation of Stereoisomers and Their Spacial Models Corresponding to the Given Molecular Structure.

135. T. Wieland, A. Kerber, and R. Laue, *J. Chem. Inf. Comput. Sci.,* **36**, 413 (1996). Principles of the Generation of Constitutional and Configurational Isomers.

136. http://www.mathe2.uni-bayreuth.de/match/online/links.html

137. A. Nijenhuis, and H. S. Wilf, *Combinatorial Algorithms*, Academic Press, New York, 1978.

138. H. S. Wilf, *J. of Algorithms,* **5**, 247 (1984). The Uniform Selection of Free Trees.

139. J. D. Dixon, and H. S. Wilf, *J. of Algorithms,* **4**, 205 (1983). The Random Selection of Unlabeled Graphs.

140. N. C. Wormald, *SIAM J. Comput.,* **16**, 717 (1987). Generating Random Unlabeled Graphs.

141. G. C. Derringer, and R. L. Markham, *J. Appl. Polymer Sci.,* **30**, 4609 (1985). A Computer-Based Methodology for Matching Polymer Structure with Required Properties.

142. R. Nilakantan, N. Bauman, and R. Venkataraghavan, *J. Chem. Inf. Comput. Sci.,* **31**, 527 (1991). A Method for Automatic Generation of Novel Chemical Structures and Its Potential Applications to Drug Discovery.

143. N. Metropolis, and A. W. Rosenbluth, *J. Chem. Phys.,* **21**, 1087 (1953). Equation of State Calculation by Fast Computing Machines.

144. R. Judson, in *Reviews in Computational Chemistry*, K. B. Lipkowitz and D. B. Boyd, Eds., Wiley, 1997, Vol. 10, pp. 1-73. Genetic Algorithms and Their Use in Chemistry.

145. J.-L. Faulon, *J. Chem. Inf. Comput. Sci.,* **34**, 731 (1996). Stochastic Generator of Chemical Structure. 2. Using Simulated Annealing to Search the Space of Constitutional Isomers.

146. C. Steinbeck, *J. Chem. Inf. Comput. Sci.,* **41**, 1500 (2001). Seneca: A Platform-Independent, Distributed, and Parallel System for Computer-Assisted Structure Elucidation in Organic Chemistry.

147. A. Globus, J. Lawton, and T. Wipke, *Nanotechnology,* **10**, 290 (1999). Automatic Molecular Design Using Evolutionary Techniques.

148. V. Venkatasubramanian, K. Chan, and J. M. Caruthers, *J. Chem. Inf. Comput. Sci.,* **35**, 188 (1995). Evolutionary Design of Molecules with Desired Properties Using the Genetic Algorithm.

149. R. Sheridan, S. SanFeliciano, and S. Kearsley, *J. Molec. Graphics and Modelling,* **18**, 320 (2000). Designing Targeted Libraries with Genetic Algorithms.

150. R. P. Sheridan, S. G. SanFeliciano, and S. K. Kearsley, *J. Molec. Graphics and Modelling* , **18**, 320 (2000). Designing Targeted Libraries with Genetic Algorithms.

151. J. Meiler, and M. Will, *J. Chem. Inf. Comput. Sci.,* **41**, 1535 (2001). Automated Structure Elucidation of Organic Molecules from 13c NMR Spectra Using Genetic Algorithms and Neural Networks.

152. C.-W. Lam, *J. Math. Chem.,* **23**, 421 (1998). A Mathematical Relationship between the Number of Isomers of Alkenes and Alkynes: A Result Established from the Enumeration of Isomers of Alkenes from Alky Biradicals.

153. J. R. Dias, *J. Chem. Inf. Comput. Sci.,* **30**, 61 (1990). Benzenoid Series Having a Constant Number of Isomers.

154. S. J. Cyvin, *Chem. Phys. Lett.,* **181**, 431 (1991). Note on the Series of Fully Benzenoid Hydrocarbons with a Constant Number of Isomers.

155. S. J. Cyvin, and J. Brunvoll, *Chem. Phys. Lett.,* **176**, 413 (1991). Series of Benzenoid Hydrocarbons with a Constant Number of Isomers.

156. J. R. Dias, *Chem. Phys. Lett.,* **176**, 559 (1991). Enumeration of Benzenoid Series Having a Constant Number of Isomers.

157. J. R. Dias, *MATCH,* **26**, 87 (1991). Strictly Pericondensed Benzenoid Isomers.

158. S. J. Cyvin, B. N. Cyvin, and J. Brunvoll, *MATCH,* **26**, 63 (1991). Isomer Enumeration of Catafusenes, $C_{4n+2}H_{2n+4}$ Benzenoid and Helicenic Hydrocarbons.

159. J. Brunvoll, S. J. Cyvin, B. N. Cyvin, and I. Gutman, *MATCH,* **24**, 51 (1989). Essentially Disconnected Benzenoids: Distribution of K, the Number of Kekule Structures, in Benzenoid Hydrocarbons VIII.

160. B. N. Cyvin, Z. Fuji, G. Xiaofeng, J. Brunvoll, and S. J. Cyvin, *MATCH,* **29**, 143 (1993). On the Total Number of Polyhexes with Ten Hexagons.

161. J. R. Dias, *J. Chem. Inf. Comput. Sci.,* **31**, 89 (1991). Benzenoid Series Having a Constant Number of Isomers. 3. Total Resonant Sextet Benzenoids and Their Topological Characteristics.

162. J. R. Dias, *J. Chem. Inf. Comput. Sci.,* **30**, 53 (1990). Isomer Enumeration and Topological Characteristics of Benzenoid Quinones.

163. B. N. Cyvin, J. Brunvoll, C. Rong-si, and S. J. Cyvin, *MATCH,* **29**, 131 (1993). Coronenic Coronoids: A Course in Chemical Enumeration.

164. D. J. Klein, T. P. Zivkovic, and A. T. Balaban, *MATCH,* **29**, 107 (1993). The Fractal Family of Coro-[N]-Enes.

165. S. J. Cyvin, B. N. Cyvin, and J. Brunvoll, *MATCH,* **30**, 73 (1994). The Number of Pyrene Isomers Is Still Unknown.

166. P. W. Fowler, P. Hansen, and D. Stevanovic, *Comm. Math. Comp. Chem. (MATCH),* **48**, 37 (2003). A Note on the Smallest Eigenvalue of Fullerenes.

167. M. Yoshida, and E. Osawa, *Bull. Chem. Soc. Jpn.,* **68**, 2073 (1995). Formalized Drawing of Fullerene Nets. 1. Algorithm and Exhaustive Generation of Isomeric Structures.

168. http://cs.anu.edu.au/~bdm/plantri/fullgen-guide.txt

169. I. Novak, *J. Chem. Educ.,* **73**, 120 (1996). Chemical Enumeration with Mathematica.

170. J. R. Dias, *Chem. Phys. Lett.,* **185**, 10 (1991). Series of Fluorenoid/Fluoranthenoid Hydrocarbons Having a Constant Number of Isomers.

171. H. J. Luinge, *MATCH,* **27**, 175 (1992). AEGIS, a Structure Generation Program in Prolog.

172. S. G. Molodtsov, *MATCH,* **30**, 213 (1994). Computer-Aided Generation of Molecular Graphs.

173. B. N. Cyvin, J. Brunvoll, and S. J. Cyvin, *MATCH,* **33**, 35 (1996). Di-5-Catafusenes, a Subclass of Indacenoids.

174. J. Brunvoll, S. J. Cyvin, and B. N. Cyvin, *MATCH,* **34**, 91 (1996). Azulenoids.

175. B. N. Cyvin, J. Brunvoll, and S. J. Cyvin, *MATCH,* **34**, 109 (1996). Isomer Enumeration of Unbranched Catacondensed Polygonal Systems with Pentagons and Heptagons.

176. H. Dolhaine, H. Honig, and M. van Almsick, *MATCH,* **39**, 21 (1999). Sample Applications of an Algorithm for the Calculation of Isomers with More Than One Type of Achiral Substituent.

177. H. Dolhaine, and H. Honig, *MATCH,* **46**, 91 (2002). Full Isomer-Tables of Inositol-Oligomers up to Tetramers.

178. S. Davidson, *J. Chem. Inf. Comput. Sci.,* **42**, 147 (2002). Fast Generation of an Alkane-Series Dictionary Ordered by Side-Chain Complexity.

179. S. Bohanec, and J. Zupan, *MATCH,* **27**, 49 (1992). Structure Generator GEN.

180. R. Barone, F. Barberis, and M. Chanon, *MATCH,* **32**, 19 (1995). Exhaustive Generation of Organic Isomers from Base 2 and Base 4 Numbers.

181. C. Le Bret, *MATCH,* **41**, 79 (2000). Exhaustive Isomer Generation Using the Genetic Algorithm.

182. M. E. Elyashberg, *Russ. Chem. Rev.,* **68**, 525 (1999). Expert Systems for Structure Eluicdation of Organic Molecules by Spectral Methods.

183. J. B. Hendrickson, and C. A. Parks, *J. Chem. Inf. Comput. Sci.*, 101 (1991). Generation and Enumeration of Carbon Skeletons.

184. S. Y. Zhu, and J. P. Zhang, *J. Chem. Inf. Comput. Sci.,* **22**, 34 (1982). Exhaustive Generation of Structural Isomers for a Given Empirical Formula - a New Algorithm.

185. X. Shao, C. Wen-sheng, and M. Zhang, *Jisuanji Yu Yingyong Huaxue,* **15**, 169 (1998). Generation of Isomers of Organic Molecules Using Genetic Algorithms.

186. M. Badertscher, A. Korytko, K. P. Schulz, M. Madison, M. E. Munk, P. Portmann, M. Junghans, P. Fontana, and E. Pretsch, *Chemometrics and Intelligent Laboratory Systems,* **51**, 73 (2002). Assemble 2.0: A Structure Generator.

187. M. Carabedian, L. Dagane, and J. E. Dubois, *Anal. Chem.,* **60**, 2186 (1988). Elucidation by Progressive Intersection of Ordered Substructures from Carbon-13 Nuclear Magnetic Resonance.

188. R. Grund, A. Kerber, and R. Laue, *MATCH,* **27**, 87 (1992). MOLGEN, Ein Computeralgebra System Fur Die Konstruktion Molekularer Graphen.

189. M. E. Elyashberg, K. A. Blinov, A. J. Williams, E. R. Martirosian, and S. G. Molodtsov, *J. Nat. Prod.,* **65**, 693 (2002). Application of a New Expert System for the Structure Elucidation of Natural Products from Their 1D and 2D NMR Data.

190. T. Lindel, J. Junker, and M. Kock, *J. Molec. Mod.,* **3**, 364 (1997). Cocon: From NMR Correlation Data to Molecular Constitutions.

191. M. Will, and J. Richert, *J. Chem. Inf. Comput. Sci.,* **37**, 403 (1997). Specsolv - an Innovation at Work.

192. C. Hu, and L. Xu, *Fenxi Huaxue,* **20**, 643 (1992). Computer Automatic Structure Elucidation Expert System, Esesoc.

193. J. Hao, L. Xu, and C. Hu, *Science in China, Series B: Chemistry,* **43**, 503 (2000). Expert System for Elucidation of Structures of Organic Compounds (Esesoc) - Algorithm on Stereoisomer Generation.

194. B. D. Christie, and M. E. Munk, *J. Am. Chem. Soc.,* **113**, 3750 (1991). The Role of Two-Dimensional Nuclear Magnetic Resonance Spectroscopy in Computer-Enhanced Structure Elucidation.

195. C. Peng, S. Yuan, C. Zheng, Y. Hui, H. Wu, and K. Ma, *J. Chem. Inf. Comput. Sci.,* **34**, 814 (1994). Application of Expert System Cisoc-Ses to the Structure Elucidation of Complex Natural Products.

196. M. E. Elyashberg, E. R. Martirosian, Y. Z. Karasev, H. Thiele, and H. Somberg, *Analytica Chemica Acta,* **337**, 265 (1997). X-Pert: A User-Friendly Expert System for Molecular Structure Elucidation by Spectral Methods.

197. J.-L. Faulon, and P. G. Hatcher, *Energy and Fuels,* **8**, 402 (1994). Is There Any Order in the Structure of Lignin?

198. M. S. Diallo, A. Simpson, P. Gassman, J.-L. Faulon, J. J. H. Johnson, W. A. Goddard, and P. G. Hatcher, *Environ. Sci. & Technol,* **37**, 1783 (2003). 3-D Structural Modeling of Humic Acids through Experimental Characterization, Computer Assisted Structure Elucidation and Atomistic Simulations. 1. Clesea Soil Humic Acid.

199. M. S. Diallo, A. Strachan, J.-L. Faulon, and W. A. Goddard, *Petroleum Science and Technology, in press*, (2003). Properties of Petroleum Geomacromolecules through

Computer Assisted Structure Elucidation and Atomistic Simulations.1. Bulk Arabian Light Asphaltenes.

200. A. Williams, G. Martin, K. A. Blinov, and M. E. Elyashberg, in *44th Annual Meeting of the American Society of Pharmacognosy*, Chapel Hill, NC, 2003. All Good Things to Those Who Wait: Solving a Structure Computationally after 10 Years of Human Effort.

201. L. A. Thompson, and J. A. Ellman, *Chem. Rev.,* **96**, 555 (1996). Synthesis and Applications of Small Molecule Libraries.

202. E. M. Gortdon, M. A. Gallop, and D. V. Patel, *Acc. Chem. Res.,* **29**, 144 (1996). Strategy and Tactics in Combinatorial Organic Synthesis. Applications to Drug Discovery.

203. F. Balkenhohl, C. v. d. Bussche-Hunnefeld, A. Lansky, and C. Zechel, *Angew Chem. Int. Ed. Engl.,* **35**, 2289 (1996). Combinatorial Synthesis of Small Organic Molecules.

204. G. S. Sittampalam, S. D. Kahl, and W. P. Janzen, *Curr. Opin. Chem. Biol.,* **1**, 384 (1997). High-Throughput Screening: Advances in Assay Technologies.

205. K. R. Oldenburg, *Ann. Rep. Med. Chem,* **33**, 301 (1998). Current and Future Trends in High Throughput Screening for Drug Discovery.

206. R. Cramer, D. Patterson, R. Clark, F. Soltanshahi, and M. Lawless, *J. Chem. Inf. Comput. Sci.,* **38**, 1010 (1998). Virtual Compound Libraries: A New Approach to Decision Making in Molecular Discovery Research.

207. Y. C. Martin, *Perspect. Drug Disc. Des.,* **7**, 159 (1997). Challenges And Prospects For Computational Aids To Molecular Diversity.

208. D. C. Spellmeyer, J. M. Blaney, and E. M. Martin, in *Practical Application of Computer-Aided Drug Design*, P. S. Charifson, Ed., Dekker, New York, 1997, pp. 165-194. Computational Approaches to Chemical Libraries.

209. R. A. Lewis, S. D. Pickett, and D. E. Clark, in *Reviews in Computaional Chemistry*, K. B. Lipkowitz and D. B. Boyd, Eds., VCH Publishers, New York, 2000, Vol. 16, pp. 1-51. Computer-Aided Molecular Diversity Analysis and Combinatorial Library Design.

210. E. Kick, D. C. Roe, A. Skillman, G. Liu, T. Ewing, Y. Sun, I. Kuntz, and J. Ellman, *Chemistry & Biology,* **4**, 297 (1997). Structure-Based Design and Combinatorial Chemistry Yield Low Nanomolar Inhibitors of Cathepsin D.

211. D. C. Roe, Appplication and Development of Tools for Structure-Based Drug Design, University of California, San Francisco.

212. Y. Sun, T. J. A. Ewing, A. G. Skillman, and I. D. Kuntz, *J. Comput.-Aided Mol. Design,* **12**, 597 (1998). Combidock: Structure-Based Combinatorial Docking and Library Design.

213. M. Rarey, and T. Lengauer, *Perspect. Drug Disc. Des.,* **20**, 63 (2000). A Recursive Algorithm for Efficient Combinatorial Library Docking.

214. H. Bohm, D. Banner, and L. Weber, *J. Comput.-Aided Mol. Design,* **13**, 51 (1999). Combinatorial Docking and Combinatorial Chemistry: Design of Potent Non-Peptide Thrombin Inhibitors.

215. C. Murray, D. Clark, T. Auton, M. Firth, J. Li, R. Sykes, B. Waszkowycz, D. Westhead, and S. Young, *J. Comput.-Aided Mol. Design,* **11**, 193 (1997). Pro_Select: Combining Structure-Based Drug Design and Combinatorial Chemistry for Rapid Lead Discovery. 1. Technology.

216. P. Willett, J. Barnard, and G. Downs, *J. Chem. Inf. Comput. Sci.,* **38**, 983 (1998). Chemical Similarity Searching.

217. S. Teig, *J. Bio. Scr.,* **3**, 85 (1998). Informative Libraries Are More Useful Than Diverse Ones.

218. C. Lipinski, F. Lombardo, B. Dominy, and P. Feeney, *Advanced Drug Delivery Reviews,* **23**, 3 (1997). Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings.

219. J. Wang, and K. Ramnarayan, *J. Combinatorial Chemistry,* **1**, 524 (1999). Toward Designing Drug-Like Libraries: A Novel Computational Approach for Prediction of Drug Feasibility of Compounds.

220. J. Sadowski, and H. Kubinyi, *J. Med. Chem.,* **41**, 3325 (1998). A Scoring Scheme for Discriminating between Drugs and Nondrugs.

221. Ajay, W. Walters, and M. Murcko, *J. Med. Chem.,* **41**, 3314 (1998). Can We Learn to Distinguish between "Drug-Like" and "Nondrug-Like" Molecules?

222. V. Gillet, P. Willett, and J. Bradshaw, *J. Chem. Inf. Comput. Sci.,* **37**, 731 (1997). The Effectiveness Of Reactant Pools For Generating Structurally-Diverse Combinatorial Libraries.

223. E. Jamois, M. Hassan, and M. Waldman, *J. Chem. Inf. Comput. Sci.,* **40**, 63 (2000). Evaluation Of Reagent-Based And Product-Based Strategies In The Design Of Combinatorial Library Subsets.

224. R. Gugisch, A. Kerber, R. Laue, M. Meringer, and J. Weidinger, *Commun. Math. Chem. (MATCH),* 189 (2000). Molgen-Comb, a Software Package for Combinatorial Chemistry.

225. G. Downs, and J. Barnard, *J. Chem. Inf. Comput. Sci.,* **37**, 59 (1997). Techniques for Generating Descriptive Fingerprints in Combinatorial Libraries.

226. B. Leland, B. Christie, J. Nourse, D. Grier, R. Carhart, T. Maffett, S. Welford, and D. Smith, *J. Chem. Inf. Comput. Sci.,* **37**, 62 (1997). Managing the Combinatorial Explosion.

227. A. Leach, J. Bradshaw, D. Green, and M. Hann, *J. Chem. Inf. Comput. Sci.,* **39**, 1161 (1999). Implementation of a System for Reagent Selection and Library Enumeration, Profiling, and Design.

228. C. A. James, Daylight Theory Manual, Daylight, Chemical Information Systems Inc., Mission Viejo, CA.

229. V. Lobanov, and D. Agrafiotis, *Combinatorial Chemistry & High Throughput Screening,* **5**, 167 (2002). Scalable Methods for the Construction and Analysis of Virtual Combinatorial Libraries.

230. Moe Software, Chemical Computing Group, 1010 Sherbrook St. W., Suite 910, Montreal, Canada H3A 2R7.

231. R. Brown, and Y. Martin, *J. Med. Chem.,* **40**, 2304 (1997). Designing Combinatorial Library Mixtures Using a Genetic Algorithm.

232. V. Gillet, P. Willett, P. Fleming, and D. Green, *J. Molec. Graphics and Modelling,* **20**, 491 (2002). Designing Focused Libraries Using Moselect.

233. A. Good, and R. A. Lewis, *J. Med. Chem.,* **40**, 3926 (1997). New Methodology For Profiling Combinatorial Libraries and Screening Sets: Cleaning Up The Design Process With HARPICK.

234. M. Hassan, J. Bielawski, J. Hempel, and M. Waldman, *Molecular Diversity,* **2**, 64 (1996). Optimization and Visualization of Molecular Diversity of Combinatorial Libraries.

235. D. Agrafiotis, *J. Chem. Inf. Comput. Sci.,* **37**, 841 (1997). Stochastic Algorithms for Maximizing Molecular Diversity.

236. V. Gillet, *J. Comput.-Aided Mol. Design,* **16**, 371 (2002). Reactant- and Product-Based Approaches to the Design of Combinatorial Libraries.

237. M. P. Beavers, and X. Chen, *J. Molec. Graphics and Modelling,* **20**, 463 (2002). Structure-Based Combinatorial Library Design: Methodologies And Applications.

238. T. Haque, A. Skillman, C. Lee, H. Habashita, I. Gluzman, T. Ewing, D. Goldberg, I. Kuntz, and J. Ellman, *J. Med. Chem.*, **42**, 1428 (1999). Potent, Low-Molecular-Weight Non-Peptide Inhibitors of Malarial Aspartyl Protease Plasmepsin II.

239. L. B. Kier, and L. H. Hall, *Quant. Struct.-Act. Relat.,* **12**, 383 (1993). The Generation of Molecular Structure for a Graph-Based Equation.

240. R. Bruggemann, S. Pudenz, L. Carlsen, P. B. Sorensen, M. Thomsen, and R. K. Mishra, *SAR and QSAR in Envir. Res.,* **11**, 473 (2001). The Use of Hasse Diagrams as a Potential Approach for Inverse QSAR.

241. S. Garg, and L. E. K. Achenie, *Biotechnol. Prog.,* **17**, 412 (2001). Mathematical Programming Assisted Drug Design for Nonclassical Antifolates.

242. M. I. Skvortsova, K. S. Fedyaev, V. A. Palyulin, and N. S. Zefirov, *Internet Electron. J. Mol. Des.,* **2**, 70 (2003). Molecular Design of Chemical Compounds with Prescribed Properties from QSAR Models Containing the Hosoya Index.

243. D. P. Visco, R. S. Pophale, M. D. Rintoul, and J.-L. Faulon, *J. Molecular Graphics and Modeling,* **20**, 429 (2002). Developing a Methodology for an Inverse Quantitative Structure Activity Relationship Using the Signature Molecular Descriptor.

244. C. J. Churchwell, M. D. Rintoul, S. Martin, D. P. Visco Jr., A. Kotu, R. S. Larson, L. O. Sillerud, D. C. Brown, and J.-L. Faulon, *J. Molecular Graphics & Modelling,* **22**, 263

(2004). The Signature Molecular Descriptor. 3. Inverse Quantitative Structure-Activity Relationship of ICAM-1 Inhibitory Peptides.

245. L. A. Goldberg, and M. Jerrum, *SIAM J. Comput.,* **29**, 834 (1999). Randomly Sampling Molecules.

246. R. S. Pearlman, *Chem. Des. Auto. News,* **2**, 1 (1987). Rapid Generation of High Quality Approximate 3D Molecular Structures.

247. J. Gasteiger, C. Rudolph, and J. Sadowski, *Tetrahedron Comp. Method.,* **3**, 537 (1990). Automatic Generation of 3D-Atomic Coordinates for Organic Molecules.

248. J. Sadowski, and J. Gasteiger, *Chem. Rev.,* **93**, 2567 (1993). From Atoms and Bonds to Three-Dimensional Atomic Coordinates: Automatic Model Builders.

249. J. Gasteiger, J. Sadowski, J. Schuur, P. Selzer, L. Steinhauer, and V. Steinhauer, *J. Chem. Inf. Comput. Sci.,* **36**, 1030 (1996). Chemical Information in 3D-Space.

DISTRIBUTION:

Department of Chemistry
MSC03 2060
1 University of New Mexico
Albuquerque, NM 87131-0001

Department of Computer Science
MSC01 11301
1 University of New Mexico
Albuquerque, NM 87131-0001

Donald P. Visco
Tennessee Technological University
Department of Chemical Engineering
Box 5013
Cookeville, TN

1	MS	0321	Bill Camp, 9200
1		1110	David Womble, 9210
1		0310	M. D. Rintoul, 9212
1		0310	W. Mike Brown, 9212
1		0318	George Davidson, 9212
1		0310	Shawn S. Martin, 9212
1		0310	Steve Plimpton, 9212
1		1111	Bruce Hendrickson, 9215
1		1110	Bob Carr, 9215
1		1110	Bill Hart, 9215
1		1110	Amy Johnson, 9215
1		1110	Cynthia Philips, 9215
1		0885	Grant S. Heffelfinger, 1802
10		9951	Diana Roe, 8130
10		9951	Jean-Loup Faulon, 9212
3		9018	Central Technical Files, 8945-1
1		0899	Technical Library, 9616
1		9021	Classification Office, 8511 for Technical Library, MS 0899, 9616 DOE/OSTI via URL

This page intentionally left blank.