STRUCTURE ENUMERATION AND SAMPLING

MARKUS MERINGER

To appear in Handbook of Chemoinformatics Algorithms

Chemical structure enumeration and sampling have been studied by mathematicians, computer scientists and chemists for quite a long time. Given a molecular formula plus, optionally, a list of structural constraints, the typical questions are: (1) How many isomers exist? (2) Which are they? And, especially if (2) cannot be answered completely: (3) How to get a sample?

In this chapter we describe algorithms for solving these problems. The techniques are based on the representation of chemical compounds as molecular graphs (see Chapter 2), i.e. they are mainly applied to constitutional isomers. The major problem is that *in silico* molecular graphs have to be represented as labeled structures, while in chemical compounds, the atoms are not labeled. The mathematical concept for approaching this problem is to consider orbits of labeled molecular graphs under the operation of the symmetric group. We have to solve the so–called *isomorphism problem.*

According to our introductory questions, we distinguish several disciplines: counting, enumerating and sampling isomers. While counting only delivers the number of isomers, the remaining disciplines refer to constructive methods. Enumeration typically encompasses exhaustive and non–redundant methods, while sampling typically lacks these characteristics. However, sampling methods are sometimes better suited to solve real–world problems.

There is a wide range of applications where counting, enumeration and sampling techniques are helpful or even essential. Some of these applications are closely linked to other chapters of this book. Counting techniques deliver pure *chemical information*, they can help to estimate or even determine sizes of chemical databases or compound libraries obtained from combinatorial chemistry.

Constructive methods are essential to *structure elucidation* systems, see Chapter 9. They are used to generate structures that fulfill structural restrictions obtained from spectroscopy in a pre–generation step, while in a post–generation step virtual spectra of the generated structures can be computed and compared with the measured data in order to determine, which of them achieves the best fit.

Date: submitted September 10, 2008; revised January 25, 2009 and April 13, 2009.

Other applications use structure enumeration algorithms in order to produce candidate structures for *virtual screening*, see Chapter 5. *Structure–activity* and *structure–property relationships*, as introduced in Chapter 6, can be used in combination with structure enumeration or sampling as rudimentary approaches towards inverse QSAR (see Chapter 10) and *de novo design* algorithms often have their roots in conventional structure generation.

The non-quantitative aspects of reaction network generation (Chapter 11) are also based on methods similar to those used for isomer enumeration.

1. Isomer counting

Counting means that only the number of structures is calculated, the structures themselves are not produced by the algorithm. The most powerful counting technique available to chemists is Pólya's theorem [1], see also [2, 3]. There are, of course, various predecessors, e.g. a paper by Lunn and Senior [4], who were the first to note that group theory plays a role, and a paper by Redfield [5] that contained even better results. However, Pólya's paper gave rise to the development of a whole theory that is nowadays called *Pólya's Theory of Counting.* Typical applications are counting of *permutational isomers* and *acyclic compounds.*

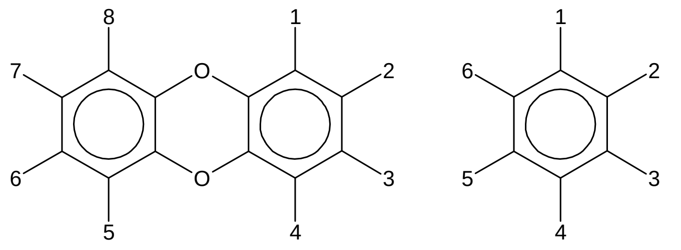
1.1. Counting permutational isomers.

Pólya's approach to the enumeration of molecules with a given molecular formula is to subdivide the molecule in question into a *skeleton* and a set of *univalent substituents.* It leads to the following challenge: Evaluate the set of essentially different distributions of the substituents over the sites of the skeleton with respect to the given *symmetry group* of the skeleton.

In mathematical terms, the symmetry group G acts on the set of mappings m^n from the n sites of the skeleton onto the m available substituents. The set of orbits under this group operation, $m^n//G$, is in one to one relation with the different constitutions.

If the skeleton shows no symmetries, i.e. if G is of order 1, then it is clear that there are m^n different substitutions. Note, that m^n has two different meanings, once for denoting the set of mappings and once for its cardinality. If the order of G is larger than 1, the situation is more interesting.

The resulting isomers are called *permutational* or *substitutional* isomers. For example the 22 permutational isomers of dioxin (tetrachlorodibenzo-p-dioxin) are the essentially different distributions of four hydrogen and four chlorine atoms over the eight sites of the skeleton depicted on the left:



Counting these isomers is described in detail in Kerber's comprehensive book [6] on finite group actions. In this section we will discuss the example of permutational isomers of dichlorobenzene, which are based on the benzene skeleton sketched on the right.

First we will try to describe Pólya's approach in general. The symmetry group G of the skeleton with respect to the n binding sites is required as input. The procedure works with the topological symmetry group as well as with the geometrical symmetry group. Of course, the results might differ. For ways to compute a molecule's symmetry group see Chapter 2. A suitable data structure for representing groups is described by Sims [7].

The reader should be familiar with the cycle notation of permutations, which is briefly described in Example 1.2 below. At this point, it is useful to know that every permutation has a unique decomposition into disjoint cycles. For more details on permutations and cycles, the reader is referred to [6].

The *cycle index* of a permutation $g \in G$ is a monomial in variables z_k. It is defined as

$$Z(g) = \prod_{k=1}^{n} {z_k}^{c_k(g)}, \tag{1.1}$$

where $c_k(g)$ is the number of cycles of g having length k. The cycle index $Z(G)$ of G is the averaged sum of cycle indices of group elements:

$$Z(G) = \frac{1}{|G|} \sum_{g \in G} \prod_{k=1}^{n} {z_k}^{c_k(g)}. \tag{1.2}$$

In order to obtain a *counting series* from the cycle index, a so-called *generating function* has to be inserted. Having $m > 1$ different chemical elements to choose from, a suitable generating function is $y_1 + ... + y_m$. Inserting the generation function into the cycle index means that every occurrence of z_k in $Z(G)$ is replaced by ${y_1}^k + ... + {y_m}^k$. If $m = 2$ the easier generation function $1 + x$ can be used instead, and during insertion z_k is replaced by $1 + x^k$.

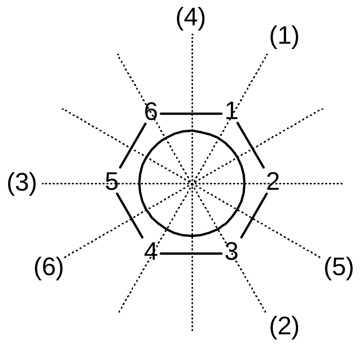
The coefficient of the monomial $\prod_{i=1}^{m} y_i^{j_i}$ of the counting series equals the number of isomers with j_i substituents of type i. In the case where we have just two different elements to substitute, say H and Cl, the coefficient of x^j equals the number of isomers with j hydrogen atoms. To summarize, we can formulate the following

Algorithm 1.1. *Counting permutational isomers by Pólya's theorem*

(1) Calculate the cycle index of the skeleton with respect to the binding sites.
(2) Insert the generation function into the cycle index.
(3) Evaluate the coefficients of the counting series.

This formal description of how to solve the counting problem in general needs to be illustrated by an example. Counting $C_6H_4Cl_2$ constitutions with a benzene skeleton will be explained step by step.

Example 1.2. Counting isomers of dichlorobenzene
Below we see the benzene skeleton together with its symmetry axes (1),...,(6).



Besides six axis reflections, the benzene skeleton allows some more symmetry operations, namely five proper rotations. Table 1 lists all the symmetry operations and the according permutations of benzene's symmetry group D_{6h}. E denotes the identity, $C_i^{+/-}$ represents a rotation by $360/i$ degrees, where the sign describes the direction of the rotation, and $\sigma_v^{(j)}$ represents a reflection at axis (j).

Permutations are given in two notations. The list representation might appear more straightforward to the reader, because for a permutation π the i–th component in the list simply defines the image of i, i.e. the list representation of π is $[\pi(1), \ldots, \pi(n)]$.

The cycle notation is a little more difficult to understand, but gives direct access to the cycle index, which is needed to compute counting series. The cycle representation consists of 1 to n cycles, which are enclosed by round brackets. Each cycle itself consists of 1 to n elements. Cycles of only one element (i) show that i is fixed under the permutation, i.e. $\pi(i) = i$. Sometimes such cycles are even suppressed in cycle notations. Cycles $(i_1, \ldots, i_l)$ with more than one element indicate that i_1 is mapped onto i_2, i_2 is mapped onto i_3 and so on. The length l of the cycle is determined by the minimum number of applications of π that map i again onto i, i.e. $l = \min\{h : \pi^h(i) = i\}$. In particular this

Operation	List-representation	Cycle-representation	Cycle-index
E	[1 2 3 4 5 6]	(1)(2)(3)(4)(5)(6)	${z_1}^6$
C_6^+	[2 3 4 5 6 1]	(1 2 3 4 5 6)	${z_6}^1$
C_6^-	[6 1 2 3 4 5]	(6 5 4 3 2 1)	${z_6}^1$
C_3^+	[3 4 5 6 1 2]	(1 3 5)(2 4 6)	${z_3}^2$
C_3^-	[5 6 1 2 3 4]	(5 3 1)(6 4 2)	${z_3}^2$
C_2	[4 5 6 1 2 3]	(1 4)(2 5)(3 6)	${z_2}^3$
$\sigma_v^{(1)}$	[1 6 5 4 3 2]	(1)(4)(2 6)(3 5)	${z_1}^2{z_2}^2$
$\sigma_v^{(2)}$	[5 4 3 2 1 6]	(3)(6)(1 5)(2 4)	${z_1}^2{z_2}^2$
$\sigma_v^{(3)}$	[3 2 1 6 5 4]	(2)(5)(1 3)(4 6)	${z_1}^2{z_2}^2$
$\sigma_v^{(4)}$	[6 5 4 3 2 1]	(1 6)(2 5)(3 4)	${z_2}^3$
$\sigma_v^{(5)}$	[4 3 2 1 6 5]	(1 4)(2 3)(5 6)	${z_2}^3$
$\sigma_v^{(6)}$	[2 1 6 5 4 3]	(1 2)(3 6)(4 5)	${z_2}^3$

TABLE 1. Permutations of the automorphism group D_{6h} of benzene

means that the last element of the cycle, i_l, is mapped onto i_1, and generally the cycle of i_1 can be written as $(i_1, \pi(i_1), \pi^2(i_1), \dots \pi^{l-1}(i_1))$.

Finally, the cycle indices of the elements of the automorphism group can be derived directly from the cycle notations using Equation 1.1. This results in the cycle index

$$Z(D_{6h}) = \frac{1}{12}({z_1}^6 + 4{z_2}^3 + 2{z_3}^2 + 2{z_6}^1 + 3{z_1}^2{z_2}^2)$$

for benzene's automorphism group D_{6h}. Inserting the generating function $1 + x$ leads to the counting series

$$\begin{aligned} C(D_{6h}) &= \frac{1}{12}\big((1+x)^6 + 4(1+x^2)^3 + 2(1+x^3)^2 \\ &\quad + 2(1+x^6) + 3(1+x)^2(1+x^2)^2\big) \\ &= 1 + x + 3x^2 + 3x^3 + 3x^4 + x^5 + x^6 \end{aligned}$$

The coefficient of x^i in the counting series indicates the number of isomers with i hydrogen and $n - i$ chlorine atoms. Thus we obtain the number of isomers of benzene (1 isomer according to coefficient 1 of x^0), chlorobenzene (1 isomer according to coefficient 1 of x^1), dichlorobenzene (3 isomers according to coefficient 3 of x^2) and so on. This sums up to 13 different substitutions of the benzene skeleton with H and Cl. These 13 compounds are shown in Figure 1.

But note that a counting series itself gives no hint to the structures of the counted isomers, i.e. as soon as there is more than one isomer found, Pólya's theorem does not show how to attach the substituents

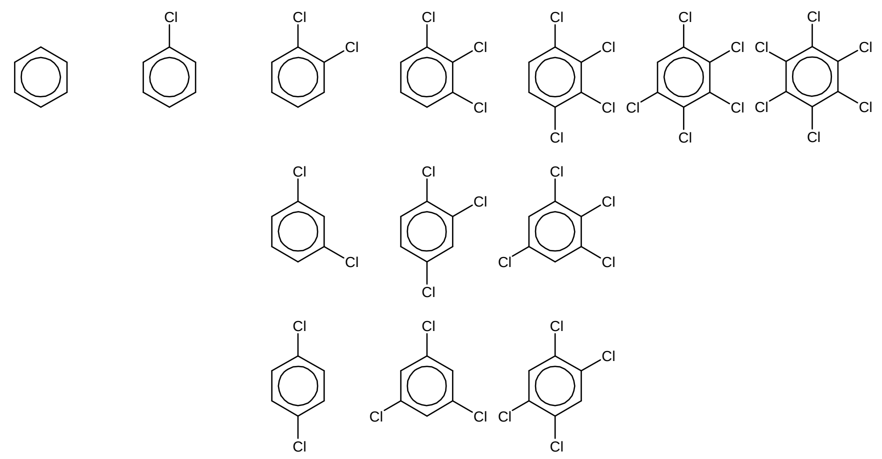


FIGURE 1. The 13 different substitutions of a benzene skeleton with H and C

to the skeleton in order to obtain all isomers. For this purpose we need constructive methods based on the principles of double cosets developed by Ruch, Klein et al. [8, 9, 10].

Examples 1.3. Cycle indices and counting series
In the following we list cycle indices of several benzenoid hydrocarbons, together with their counting series obtained by substituting $1 + x$.

- Naphthalene: $Z(D_{2h}) = \frac{1}{4}({z_1}^8 + 3{z_2}^4)$, $C(x) = 1 + 2x + 10x^2 + 14x^3 + 22x^4 + 14x^5 + 10x^6 + 2x^7 + x^8$.
- Anthracene: $Z(D_{2h}) = \frac{1}{12}(z_1^{10} + z_1^2 z_2^4 + 2z_2^5)$, $C(x) = 1 + 3x + 15x^2 + 32x^3 + 60x^4 + 66x^5 + 60x^6 + 32x^7 + 15x^8 + 3x^9 + x^{10}$.
- Phenanthrene: $Z(C_{2v}) = \frac{1}{2}({z_1}^{10} + {z_2}^5)$, $C(x) = 1 + 5x + 25x^2 + 60x^3 + 110x^4 + 126x^5 + 110x^6 + 60x^7 + 25x^8 + 5x^9 + x^{10}$.
- Tetracene: $Z(D_{2h}) = \frac{1}{4}(z_1^{12}+2z_2^6)$, $C(x) = 1+3x+21x^2+55x^3+135x^4+198x^5+236x^6+198x^7+125x^8+55x^9+21x^{10}+3x^{11}+x^{12}$.
- Triphenylene: $Z(D_{3h}) = \frac{1}{6}(z_1^2 + 2z_2^6 + 2z_3^4)$, $C(x) = 1 + 2x + 14x^2 + 38x^3 + 90x^4 + 132x^5 + 166x^6 + 132x^7 + 90x^8 + 38x^9 + 14x^{10} + 2x^{11} + x^{12}$.

We obtain the same cycle index for naphthalene as for the introductory sample of dioxin. We see that the monomial x^4 has the coefficient 22, i.e. there are 22 isomers of dioxin.

It follows from Formulas 1.1 and 1.2 that the total number of different substitutions with respect to G can be computed as

$$|m^n//G| = \frac{1}{|G|}\sum_{g\in G} m^{c(g)}, \tag{1.3}$$

where $c(g)$ denotes the number of cycles of g. We will use this later in Section 4.2 for counting constituents of combinatorial libraries, which is closely related to counting permutational isomers.

Van Almsick et al. [11] developed a software tool that calculates the number of permutational isomers using Pólya's approach. Computer algebra systems, such as commercial implementations Mathematica and Maple, or the open source system SYMMETRICA [12] are also able to conduct computations following Pólya's theory, however without any special adaptions to chemistry. The computation of numbers of permutational isomers using SYMMETRICA is available online at `symmetrica.uni-bayreuth.de/perm_iso.html` (accessibility checked January 2009).

1.2. **Counting isomers of acyclic structures and other compound classes.** Besides permutational isomers, counting series for several other compound classes have been discovered in the past. However, in contrast to permutational isomers these cannot be produced using a well defined algorithm. It were rather individual ideas that led to these counting series. Counting series are known especially for the most prominent acyclic compound classes. Most of them were derived by applying Pólya's theorem in a recursive manner, i.e. counting series themselves were used as generating functions.

Alkyl groups have the form $-\mathsf{C_nH_{2n+1}}$. They can be interpreted as rooted trees on n nodes, where the root is the carbon atom with the free valence. Let $A_n(x)$ denote the counting series for alkyl groups having n atoms. There is a recursive formula

$$A_n(x) = 1 + \frac{1}{6}x[A_{n-1}{}^3(x) + 3A_{n-1}(x)A_{n-1}(x^2) + 2A_{n-1}(x^3)] \tag{1.4}$$

starting with $A_0(x) = 1$. For $n \to \infty$ the counting series for alkyl groups is often written as

$$A(x) = \sum_{n=0}^{\infty} A_n x^n$$

with certain coefficients calculated from the recursive Equation 1.4. The first terms are

$$A(x) = 1+x+x^2+2x^3+4x^4+8x^5+17x^6+39x^7+89x^8+211x^9+507x^{10}+\dots$$

Based on this recursive approach, several counting series for other acyclic compound classes have been formulated by Read [13]:

- Primary alcohols: $\mathsf{R-CH_2-OH}$ with an alkyl group R: $xA(x)$
- Secondary alcohols: $\mathsf{R^1-CH(R^2)-OH}$ with two alkyl groups $\mathsf{R^1}$ and $\mathsf{R^2}$: $\frac{1}{2}x[A^2(x) - 2A(x) + A(x^2)]$.
- Tertiary alcohols: $\mathsf{R^1-C(R^2)(R^3)-OH}$ with alkyl groups $\mathsf{R^1, R^2}$ and $\mathsf{R^3}$: $\frac{1}{6}x[A^3(x) - 3A^2(x) + 3A(x)A(x^2) - 3A(x^2) + 2A(x^3)]$.
- Aldehydes and ketones: $\mathsf{R^1-C(=O)-R^2}$ with alkyl groups or hydrogen atoms $\mathsf{R^1}$ and $\mathsf{R^2}$: $\frac{1}{2}x[A^2(x) + A(x^2)]$.
- Alkynes: $\mathsf{R^1-C\equiv C-R^2}$ with alkyl groups or hydrogen atoms $\mathsf{R^1}$ and $\mathsf{R^2}$: $\frac{1}{2}x^2[A^2(x) + A(x^2)]$.

- Esters: $\mathsf{R^1 - C(= O) - O - R^2}$ with alkyl groups $\mathsf{R^1}$ and $\mathsf{R^2}$ where $\mathsf{R^1}$ can also be a hydrogen atom: $xA(x)[A(x) - 1]$.

Perhaps the most important counting series for acyclic compounds is the one for alkanes, i.e. compounds with formula $\mathsf{C_nH_{2n+2}}$. It has been determined as

$$\begin{aligned} a(x) &= \frac{1}{24}x[A^4(x) + 6A^2(x)A(x^2) + 3A^2(x^2) + 8A(x)A(x^3) + 6A(x^4)] \\ &- \frac{1}{2}[(A(x))^2 - A(x^2) + 1], \end{aligned}$$

and the first terms are

$$a(x) = 1 + x + x^2 + x^3 + 2x^4 + 3x^5 + 5x^6 + 9x^7 + 18x^8 + 35x^9 + 75x^{10} + \dots$$

Other compound classes for which several counting series are known are benzenoids and polyhex hydrocarbons. The review of Faulon et al. [14] offers an extensive overview on these counting series and on how they were deduced.

Although there is a counting series known for simple graphs on n nodes, no general counting series for molecular graphs with given molecular formula has been found yet. An approach for counting cubic graphs is presented [15]. The relationship between cubic and molecular graphs might not be very obvious at first sight, but will become clearer in Subsection 2.1. Recently, another small step towards a more universal counting series was found for hydroxyl ethers [16], i.e. isomers with molecular formula $\mathsf{C_iH_{2i+2}O_j}$.

Up till now, the only way to calculate the number of isomers belonging to an arbitrary molecular formula is to use structure generators. Structure generators not only calculate the number of isomers, but deliver the structures themselves as output. On the other hand, counting series are always a good choice to prove the correctness of new structure generator results. In the next section we will get to know the algorithmic concepts underlying past and present structure generators.

2. Isomer enumeration: deterministic structure generation

The construction of all constitutional isomers having the same molecular formula has a long history which will and can not be reported in detail here. Just as the representation of chemical compounds as graphs was one of the roots of graph theory, their generation was one of the challenges for the development of construction algorithms for computers.

A prominent starting point is the well known DENDRAL system [17], the development of which began already in the middle and late sixties of the last century. DENDRAL (short for DENRric ALgorithm) was developed for the automated structure elucidation of organic compounds by mass spectrometry (MS). For that purpose DENDRAL was endowed with an isomer generator that was able to process structural constraints obtained from MS (especially the molecular formula) and from other spectroscopic methods, in particular NMR.

DENDRAL is described in many computer science books as the first expert system. Moreover, it can be considered as one of the roots of chemoinformatics. Interestingly, even the NASA was among the founders of this pioneering project, with the ambitious intention to supply future Mars missions with such software, to enable analysis and interpretation of MS samples onboard a space probe and to broadcast only identified structural formulas back to earth instead of huge GC/MS data sets.

2.1. **Early cyclic and acyclic structure generators.**

At first, only acyclic structures could be constructed until there was a breakthrough in the early seventies when a decomposition of the given molecular formula into those of cyclic substructures was found. Cyclic substructures had to be combined by bridges to get molecules with the prescribed molecular formula. All possible decompositions of this kind could be determined by appropriate mathematical theorems prior to constructing these cyclic substructures.

2.1.1. *Acyclic structure generators.*

Henze and Blair [18] used the fact that a unique centroid can be found in any chemical tree for the enumeration of alkanes as early as the 1930's. The unique *centroid* is the starting point for a canonical labeling of the tree, following simple rules of precedence of the constituent radicals according to their composition and topological structure. An unambiguous notational system was established by Lederberg [19].

However, the existence of a unique (bi)centroid in a tree on n nodes had already been formulated a century earlier in Jordan's theorem [20]:

- For odd $n = 2k + 1$, there exists a unique node, called *centroid*, such that all incident subtrees have at most k nodes.
- For even $n = 2k$ there exists either
  - a unique node such that all incident subtrees have less than k nodes, or
  - a unique edge, called *bicentroid* or centroid edge, such that the incident subtrees have exactly k nodes.

This theorem shows a recursive way to generate trees. A tree on $n = 2k + 1$ nodes is composed from one node (the centroid, having degree d) and d rooted trees with less than k nodes in such a way that the sum of nodes is $n - 1$. In terms of acyclic chemical graphs (without multiple bonds) one will have to loop over all different atoms as centroid, partition the remaining atoms into subsets according to the centroid's valency, and then iterate this procedure on the subsets (with the small difference that now rooted trees have to be built). The iteration ends when no more partitioning is possible. The case of odd numbers of nodes can be processed similarly, with the variation that two atoms have to be chosen for a bicentroid. Reference [21] offers pseudo code for such an algorithm applied to alkanes.

An implementation with respect to general chemical trees was part of the DENDRAL system. Results of this acyclic generator have been published in [22].

2.1.2. *Cyclic structure generator.* Approaching the challenge of cyclic structures, Lederberg introduced a series of simplification steps that finally (apart from certain exceptions) showed a mapping from cyclic structures on certain classes of cubic graphs [23].

These initial ideas developed into a structure generator described by Masinter et al. [24, 25], which was the first generator that covered both acyclic and cyclic structures. The fundamental ideas of this structure generator will be described below. First, some terminology is required.

Chapter 2 introduced molecular graphs as representations of chemical compounds. In Figure 2 we see such a representation **1** of a substituted piperazine. The *chemical graph* **2** ignores hydrogen. The symbol U is used in the composition to denote the number of unsaturations. The number of unsaturations u is computed from the molecular formula as follows:

$$u = \frac{1}{2}\left(2 + \sum_{i=1}^{k}(i-2)a_i\right), \tag{2.1}$$

where a_i denotes the number of atoms of valence i and k is the maximum valence of the composition.

An atom of a chemical graph is called *cyclic*, if it lies on a cycle (or ring); otherwise it is called *acyclic*. This way a chemical graph can be separated into cyclic and acyclic parts. Connected components of the chemical graph induced by the cyclic atoms are called *superatoms*. Graph–theoretically, a superatom is a connected isthmus–free multigraph (short cif–graph), i.e. with no edge whose deletion would disconnect the graph. The number of free valences of the superatom is determined by the number of connections to atoms outside the superatom. The chemical graph **2** is composed of the superatom **3**, having 16 free valencies, and two acyclic carbon atoms.

The *ciliated skeleton* **4** is obtained from **3** by stripping the element symbols. A further step of abstraction is the deletion of free valences, resulting in the *cyclic skeleton* **5**. Finally, if chains of bivalent nodes are reduced to edges we obtain the *vertex graph* **6**.

Going into the reverse direction, starting from **6**, two alternative cyclic graphs that can be obtained are **7** and **8**. **9** and **10** are alternative cilitated skeletons that can be built from **5**.

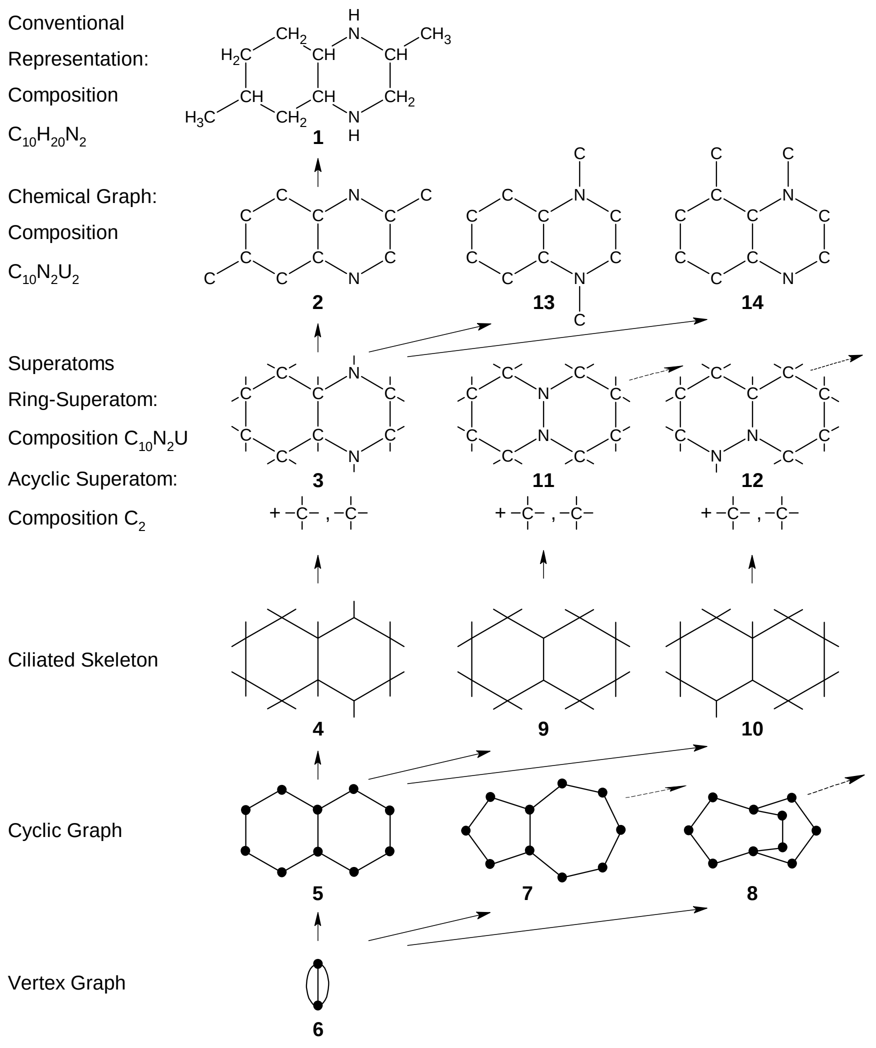


FIGURE 2. Examples of abstraction and refinement steps used in generation of cyclic structures

In this particular example, the valencies of nodes in **9** and **10** allow only unique superatoms **11** and **12**, respectively. If, for instance, 4–valent sulfur or 3–valent phosphorus would also be part of the composition, more than one superatom per ciliated skeleton would be possible.

The scheme of abstraction and specification steps between molecular graphs and vertex graphs above described indicates already a strategy for a generation algorithm, roughly following the *Divide and Conquer* principle. The algorithm consists of a sequence of partitioning steps starting from the set of atoms defined by a molecular formula that leads to the selection of vertex graphs from a catalog. A sequence of consecutive labeling steps finally reconstructs all molecular graphs that arise from a vertex graph. A more detailed description of the algorithm as outlined in [25] reads as follows:

Algorithm 2.1. *DENDRAL Isomer Generation*

(1) Determine all distinct allowable partitions of a given degree sequence V into atoms and superatom sets with assigned free valences. These partitions are based on the unsaturation of V.
(2) For each superatom set, determine all the distinct allowable allocations of the free valences to the atoms of the set.
(3) For each such free valence allocation, determine recursively the allowable sets of atoms remaining after the deletion of the bivalent atoms and the pruning of any resulting loops. This recursion is done until:
    (a) the remaining bivalent atoms in any cif–graph based on the set must all be on edges, or
    (b) one of two special cases is encountered.
(4) For each such set of atoms, if condition (a) terminates the recursion, look up in the catalog all the cif–graphs based on the nonbivalent atoms in the set and for each such graph, label the edges with the bivalent atoms. If condition (b) terminates the recursion, directly write down the allowable graphs.
(5) For each such graph, recursively label the atoms with loops and the loops and edges with bivalent atoms.
(6) For each graph so obtained, label the atoms with the free valences.
(7) For each set of atoms and superatoms obtained as above, use the tree generator to construct all the non–isomorphic connected multigraphs based on these atoms and superatoms.

This algorithm uses several subroutines which cannot be described in detail here. The *superatom partitioner* (Step 1), the *free valence partitioner* (2), the *loop–bivalent partitioner* (3) with the definition of the special cases (b), the look-up routine from the catalog (4), the *loop–bivalent labeler* (5) and the *free–valence labeler* are subject to [25] and the references cited therein. Especially for the labeling steps, see [26, 27, 28].

Example 2.2. Superatom partitioner
Step (1) of Algorithm 2.1 will be illustrated here. Table 2 shows the results of the superatom partitioner for $\mathsf{C_6H_8}$. Firstly, hydrogens are replaced by unsaturations U. According to Equation 2.1, $\mathsf{C_6H_8}$ has $u = 3$ unsaturations. A total of eleven allowed partitions of up to three superatoms are obtained.

According to the terminology introduced in Figure 2, the results of Step (5) are cyclic graphs, at Step (6) cilitated skeletons are obtained, and Step (7) delivers chemical graphs. Step (7) is also described in [24]. However, some words on the treatment of superatoms are appropriate.

In the final step, superatoms require some special treatment in the tree generator. If a superatom A has k free valences, then in forming

Partition	Superatompots	Superatompot 1	2	3	Remaining pot
1	1	C_6U_3			
2	1	C_5U_3			C
3	1	C_4U_3			C_2
4	1	C_3U_3			C_3
5	2	C_4U_2	C_2U		
6	2	C_3U_2	C_2U		C
7	2	C_2U_2	C_2U		C_2
8	2	C_4U	C_2U_2		
9	2	C_3U	C_2U_2		C
10	2	C_3U_2	C_3U		
11	3	C_2U	C_2U	C_2U	

Table 2. Allowed partitions of C_6U_3 into superatom pots and remaining pot

molecular structures which include A, A behaves differently from an atom of valence k. The difference in forming structures including A and those including an atom of valence k is the following: the k free valences on an atom of valence k are, as edge endpoints in a graph, indistinguishable, i.e. the free valences on the atom admit as symmetry group the group S_k, the full permutation group on k objects. However, the k free valences on the superatom A are usually distinguishable from a symmetry viewpoint, so the free valences on A will, in general, admit only a subgroup of S_k.

The structure generator outlined here has become popular under the name CONGEN (short for CONstained GENerator) and was used within the DENDRAL project until it was finally replaced by the advanced generator GENOA [29] (short for GENeration with Overlapping Atoms).

From today's point of view it is remarkable that a project like DENDRAL could be successfully realized. Computers were slow at that time and extremely limited in memory. Programming languages were still on a low level and software engineering was hardly recognized as a new technological discipline. However, mathematically it was state-of-the-art. But the various partitioning and labeling steps implicate a problem: it is difficult to process structural constraints efficiently. Efficiency means, that constraints can already be tested during structure generation, help to reduce intermediate results and speed up the enumeration process. Among others, this feature will be subject of the next subsection.

2.2. **Orderly generation.** There was a development by Read [30] and Faradzev [31, 32] who both presented the technique of orderly generation independently in 1978. In this technique an artificial ordering is imposed on the set of graphs that are to be generated, such that the smallest representative of a given isomorphism type always contains a subgraph that is the smallest representative of its isomorphism type. Thus, only smallest representatives have to be extended and the results have to be tested for being smallest again.

This approach allowed avoidance of pairwise isomorphism testing and keeping long lists of graphs in memory for comparison. An advantage compared with the DENDRAL generators is that orderly generation does not require any catalog of elemental graphs.

2.2.1. *Enumerating labeled graphs.* The principles underlying orderly generation are best explained using simple graphs. Let γ and γ' be simple graphs on n nodes. Nodes are labeled with numbers from 1 to n. There is an order on edges of such graphs defined as follows: for edges $e = (x, y)$, $e' = (x', y')$ with $x < y$, $x' < y'$, e is less than e', if and only if $x < x'$, or $x = x'$ and $y < y'$. This can be expressed more precisely in mathematical terms:

$$e < e' \;:\Longleftrightarrow\; x < x' \;\vee\; (x = x' \;\wedge\; y < y').$$

This induces a *lexicographical order* on the set of graphs on n nodes. Let $e_1, ..., e_t$ be the edges of γ and $e'_1, ..., e'_{t'}$ those of γ' sorted in the above order, i.e. $e_1 < ... < e_t$ and $e'_1 < ... < e'_{t'}$. Then γ is less than γ', if and only if there exists an index i with $e_i < e'_i$ and $e_j = e'_j$ for all $j < i$, or $t < t'$ and $e_j = e'_j$ for all $j \leq t$. Again, this can be expressed more conveniently using mathematical notation:

$$\begin{aligned} \gamma < \gamma' \quad :\Longleftrightarrow \quad & (\exists i < \min\{t, t'\} : e_i < e'_i \wedge \forall j < i : e_j = e'_j) \\ & \vee (t < t' \wedge \forall j \leq t : e_j = e'_j). \end{aligned}$$

As a first application, this order shows a way to construct labeled structures. We can define an algorithm that constructs labeled simple graphs according to this order.

Algorithm 2.3. *Labeled Enumeration* (γ)

```
(1) Output $\gamma$
(2) For each edge $e > \max\{e' \in \gamma\}$ do in ascending order of $e$
        Call Labeled Enumeration $(\gamma \cup \{e\})$
```

Example 2.4. Labeled graphs on three nodes

Let us have a brief look at the minimalistic example of $n = 3$ nodes. Figure 3 shows the way edges are inserted during recursive calls of *Labeled Enumeration.* During the first call with the empty graph $\{\}$ edges $(1, 2)$, $(1, 3)$ and $(2, 3)$ are used for augmentation. In the second call with graph $\{(1, 2)\}$ as the argument, edges $(1, 3)$ and $(2, 3)$ are

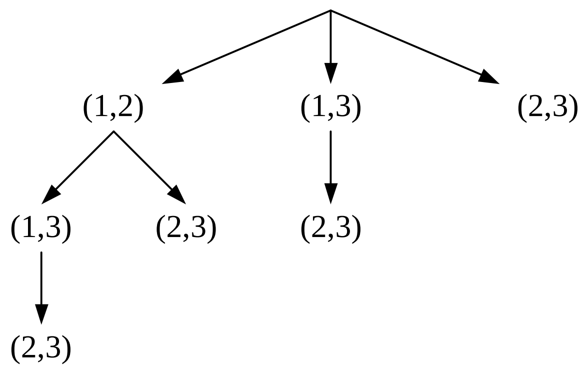


FIGURE 3. Generating tree for labeled graphs on three nodes

considered, and so on. Thus graphs are written to the output in the following order:

$\{\}$, $\{(1,2)\}$, $\{(1,2),(1,3)\}$, $\{(1,2),(1,3),(2,3)\}$, $\{(1,2),(2,3)\}$, $\{(1,3)\}$, $\{(1,3),(2,3)\}$, $\{(2,3)\}$.

It is easy to check that this is the lexicographical order as introduced above.

2.2.2. *Enumerating unlabeled graphs.* Beyond the construction sequence the ordering on the set of graphs provides a canonical form. Selecting the minimal orbit representative shows a way to avoid producing isomorphic duplicates. A graph γ is defined *canonical*, if it is *minimal* in its orbit. In mathematical terms:

$$\forall \pi \in S_n : \ \gamma \leq \gamma^{\pi}.$$

Algorithm 2.3 can be upgraded to generate minimal orbit representatives only by modifying Step (1):

Algorithm 2.5. *Unlabeled Enumeration* (γ)

```
(1) If $\gamma$ is minimal in its orbit then
        Output $\gamma$
(2) For each edge $e > \max\{e' \in \gamma\}$ do in ascending order of $e$
        Call Unlabeled Enumeration ($\gamma \cup \{e\}$)
```

However, this algorithm has to check all of the $2^{n(n-1)/2}$ labeled graphs on n nodes for canonicity. The main finding of Read [30] and Faradzev [31, 32] was, that every minimal orbit representative with q edges has a minimal subgraph with $q-1$ edges. Thus, non–minimal intermediates do not have to be considered for further augmentation. Using this knowledge, Algorithm 2.5 can be improved to

Algorithm 2.6. *Orderly Enumeration* (γ)

```
(1) If $\gamma$ is not minimal in its orbit then
        Return
(2) Output $\gamma$
```

```
(3) For each edge $e > \max\{e' \in \gamma\}$ do in ascending order of $e$
        Call *Orderly Enumeration* $(\gamma \cup \{e\})$
```

Example 2.7. Unlabeled graphs on three nodes
Continuing Example 2.4, we notice that there are four unlabeled graphs on three nodes. They have zero to three edges. The minimal orbit representatives are

$$\{\}, \{(1,2)\}, \{(1,2),(1,3)\}, \{(1,2),(1,3),(2,3)\}.$$

Comparing Algorithms 2.5 and 2.6, one canonicity test could be saved using the latter: graph $\{(1,3)\}$ would be recognized as non–minimal, and its augmentation $\{(1,3),(2,3)\}$ would not have to be considered. Of course, for increasing n the improvement in Algorithm 2.6 leads to much bigger gains in speed.

2.2.3. *Introducing constraints.* Typically one is not interested in enumerating all graphs, but just certain subsets, often denoted as *classes of graphs.* Such a class of graphs is characterized by one or more *constraint* or *restriction.* In mathematical terms a constraint is a mapping R from the set of graphs on n nodes onto the set of boolean values $\{true,\ false\}$, which is symmetry invariant:

$$\forall \pi \in S_n : \ R(\gamma) = R(\gamma^\pi).$$

A graph γ fulfills R, if $R(\gamma) = true$. Otherwise γ violates the constraint. A constraint R is called *consistent* if the violation of a graph γ to R implies that every augmentation γ' of γ violates R:

$$R(\gamma) = false \ \wedge \ \gamma \subset \gamma' \quad \Longrightarrow \quad R(\gamma') = false.$$

Examples of consistent constraints are an upper number of edges, a minimal cycle size or graph–theoretical planarity. On the other hand, the presence or absence of a certain subgraph or a maximum ring size are examples for inconsistent constraints (the precise definition of these terms would require a section on its own).

Consistent constraints can be incorporated into generating algorithms in a way that structure enumeration is accelerated. Such restrictions can be checked after each insertion of a new edge, and help to prune the generating tree. Inconsistent constraints are more problematic. Testing these constraints is only useful, when a graph is completed. Completeness itself is also described by constraints. As to generating constitutional isomers, completeness is typically defined by a given degree sequence.

Algorithm 2.8. *Orderly Enumeration with Constraints* (γ)

```
(1) If $\gamma$ is not minimal in its orbit then
        Return
(2) If $\gamma$ violates any consistent constraint then
        Return
(3) If $\gamma$ fulfills all inconsistent constraints then
```

```
            Output $\gamma$
    (4) For each edge $e > \max\{e' \in \gamma\}$ do in ascending order of $e$
            Call *Orderly Enumeration With Constraints* $(\gamma \cup \{e\})$
```

2.2.4. *Variations and refinements.* There are several variations and refinements possible that might, depending on the type of constraints, lead to a considerable speedup.

- Testing completeness is typically cheaper than other constraints like presence and absence of substructures. Thus these more expensive inconsistent constraints should be tested after completeness has been confirmed.
- Testing inconsistent constraints is often cheaper than testing canonicity. Thus it can be useful to process step (2) before step (1).

In general the sequence of tests is affected by two strategies:

- Process cheap tests first, i.e. tests that consume least computation time.
- Process selective tests first, i.e. tests that eliminate most intermediates.

Those tests that fulfill both criteria should surely be processed first, and such that fulfill none of them should be executed last. However, for expensive tests that are very selective and cheap tests with low selectivity, one has to find a trade–off.

Going back to Algorithm 2.8, step (2) is often replaced by a cheaper criterion that only tests a necessary condition for canonicity, so-called *semi–canonicity.* Without going into details this criterion only checks for transpositions τ if $\gamma \leq \gamma^{\tau}$. For a more detailed description see [33] or [34]. The full canonicity test will be delayed until the graph is completed.

If some candidate solution then turns out not to be canonical, a so called *learning criterion* provides a necessary condition for the canonicity of the lexicographic successors. The earliest extension step is determined where non–minimality could have been detected in the generation procedure. Applying this criterion will further prune the generating tree. Details on this criterion can also be found in [33] and [34].

2.2.5. *From simple graphs to molecular graphs.* Now that we have learnt the principles of orderly generation, it is about time to adapt them to molecular graphs. In contrast to simple graphs, edges of molecular graphs have a *bond multiplicity* (or *bond order*). It is convenient to use the lexicographical order on the adjacency matrix (or equivalently on the connectivity stack) as construction sequence. Objects with maximal connectivity stack are defined as canonical orbit representatives. This definition of canonicity is backward compatible in the following

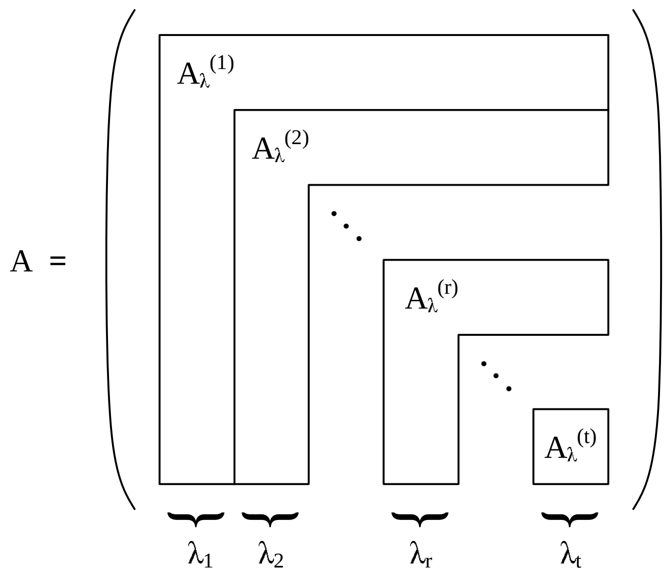


FIGURE 4. Adjacency matrix with block structure as used in Algorithm 2.9

sense: a minimal simple graphs as defined in Subsection 2.2.2 has the maximum connectivity stack in its orbit and vice versa.

Nodes of molecular graphs are colored by *element symbols*. Hydrogen atoms are typically treated implicitly, i.e. they are not represented by nodes, but instead each non–hydrogen atom has a *hydrogen count* as attribute. Further attributes of atoms are the sum of *remaining valencies*, i.e. those not bonded to hydrogen, *charges* and *unpaired electrons*. These attributes impose invariants on the set of atoms. Additionally, the *bond order distribution* of bonds incident with an atom can be used as invariant.

The combination of these attributes defines the atom state. Before starting to fill the adjacency matrix A, the atom states are assigned to rows (and columns) of A. If the number of atoms of each state cannot be deduced directly from the input, all possible distributions of atom states are generated and filling the adjacency matrix is repeated for each atom state distribution.

The assignment of atom states to rows and columns of the adjacency matrix introduces a block structure as depicted in Figure 4. Each block belongs to one of the t different atom types; λ_r equals the number of atoms of a state r.

As a first gain of this block structure no longer all $n!$ permutations of the full symmetric group S_n have to be checked during the canonicity test. Only the $\prod_{i=1}^{t} \lambda_i!$ permutations that respect the block structure have to be considered. This reduces the computational costs for canonicity testing immensely.

Algorithm 2.9 is taken from [33] and shows how the structure generator underlying MOLGEN (short for MOLecular structure GENerator), version 3.5 [35, 36] fills the adjacency matrix. Filling matrix blocks (steps 3 and 4) is iterated with testing canonicity for matrix blocks (step 5). For canonicity testing of block r only permutations from the formerly calculated automorphism group Aut^{r-1} of blocks $1, ..., r-1$ have to be taken into account.

Algorithm 2.9. *MOLGEN Orderly Enumeration*

```
(1) Start: set $r := 0$ and goto (3).
(2) Stop criterion: if $r = 0$ stop; else goto (4).
(3) Maximum filling: fill block $A^{(r)}$ (depending on $A^{(1)}, ..., A^{(n-1)}$)
    in lexicographically maximal manner so that $A^{(r)}$ fulfills the
    desired matrix properties (regarding atom states and consistent
    constraints).
    If no such filling exists then set $r := r - 1$ and goto (2); else
    goto (5).
(4) Next smaller filling: fill block $A^{(r)}$ (depending on $A^{(1)}, ..., A^{(n-1)}$)
    in lexicographically next smaller manner so that $A^{(r)}$ fulfills the
    desired matrix properties (regarding atom states and consistent
    constraints).
    If no such filling exists then set $r := r - 1$ and goto (2); else
    goto (5).
(5) Test canonicity: if $\forall \pi \in Aut^{(r-1)}(A) : A^{(r)} \geq A^{(r)}\pi$, then
        if $r = t$ (canonical matrix complete) then
          (a) if constraints are fulfilled then output $A$
          (b) goto (4)
        else determine $Aut^{(r)}(A)$, set $r := r + 1$ and goto (3).
    else goto (4).
```

This algorithm uses two subroutines, the filling of a matrix block and the canonicity test of a matrix block. Filling a matrix block is called in two different situations: In Step (3) block $A^{(r)}$ is initially filled in maximal manner. When Step (4) is called, block $A^{(r)}$ had already been filled before, and now the next smaller filling is produced. Due to their huge technical overhead, these subroutines will not be described in detail here. The reader is referred to the original publication [33]. However, this book comprises the principles of these subroutines. Canonical labeling has been introduced in Chapter 2. Filling a matrix block is done in lexicographically descending order, which is similar to constructing labeled graphs as introduced at the beginning of this subsection.

2.3. **Beyond orderly generation.** Of course, other principles can be combined with orderly generation. For instance the above–mentioned MOLGEN 3.5 allows definition of *macroatoms*. These are substructures that are treated during orderly generation as a special atom type and are expanded whenever a canonical matrix is complete. Double coset representatives are used to avoid isomorphic duplicates. This principle is already known from the construction of permutational isomers and from the treatment of superatoms during tree generation in the DENDRAL generator. In mathematics, this method of joining partial structures without producing isomorphic duplicates is known as *gluing lemma* [37, 38].

These two references [37, 38] also describe the *principle of homomorphisms.* A homomorphism is a simplification of a structure, which maps isomorphic objects onto isomorphic simplified ones. The simplification from molecular graphs to multigraphs by removing element symbols, or from multigraphs to simple graphs by forgetting bond multiplicities are examples of homomorphisms. Indeed, the DENDRAL strategy already relied on these simplification steps, only the general principle had not been worked out. In [39], this approach of simplifying by homomorphisms has been pushed to an extreme by constructing graphs with a prescribed degree sequence from regular graphs as the most simple graphs. It turned out that for huge numbers of nodes n such a generator is much faster than orderly generation only. However, for small n, that still allow generation of full lists of graphs, the generator accelerated by homomorphisms was not able to keep up with ordinary orderly generation.

Another variation of orderly generation is also worth mentioning: McKay's *enumeration by canonical construction path* [40] restricts extensions to those structures where the new edges are taken from a certain orbit of the automorphism group.

Speed plays an important role in structure enumeration, but only few theoretical results about the computational complexity are known. Goldberg's work [41] proves that the results in orderly enumeration can be computed with polynomial delay and a paper of Luks [42] shows that isomorphism testing of molecular graphs can be done in polynomial time.

A new approach named *constrained generation* [43] pays attention to the fact that isomer generators in structure elucidation typically aim at small numbers of solutions. For this reason, the ability to generate labeled structures that fulfill long lists of constraints becomes more important than efficient isomorphism avoidance. This generator has no fixed sequence of filling the adjacency matrix. Instead a heuristic method has to decide which alternative makes best use of the actual constraints. It only has to be guaranteed that each isomorphism type is constructed at least once. Its canonical representation is then stored in a hash table. If it is new, it will be written to the output, otherwise it is a duplicate. Although giving up all the expertise from orderly generation, gluing lemma and homomorphism principle looks like a step backwards, this approach, implemented in MOLGEN 4.0 [44] currently appears to be the best suited solution for application in structure elucidation. It is being used in chemical and pharmaceutical companies (where results typically are not disclosed to the public domain), as well as in public research institutions (see for example [45]).

Of course not all generation algorithms and implementations can be discussed in detail here. At least the most popular ones such as CHEMICS [46], ASSEMBLE [47, 48], as well as [49, 50, 51] are worth being cited. Number 27 of the journal MATCH is completely devoted to this topic. Faulon's review [14] also contains a large section about this topic. Free online access to MOLGEN 3.5 and the new MOLGEN 5.0 are available at `unimolis.uni-bayreuth.de/molgen` and `molgen.de`, respectively (accessibility checked January 2009).

3. Isomer sampling: stochastic structure generation

Due to the *combinatorial explosion* of numbers of constitutions with increasing numbers of atoms, it is often impossible to generate all molecular graphs belonging to a given molecular formula. Alternative methods are required, especially if no structural constraints are available, for instance if statistical statements on structural or physico–chemical properties of isomers of a certain molecular formula have to be made. Sampling techniques help to tackle such problems. A frequent requirement is a uniform probability distribution for all isomorphism types.

3.1. Uniformly distributed random sampling.

To explain the basic principle of uniformly distributed random sampling, we will again start with simple graphs. Labeled simple graphs on n nodes can be sampled with uniform distribution by simply choosing each pair of nodes with probability 0.5 as an edge.

However, the different isomorphism types have different numbers of labeled structures, thus other methods are required for uniformly distributed random sampling for unlabeled structures. Dixon and Wilf [52] solved the problem as follows.

Firstly, they choose a permutation at random from S_n and next a graph is constructed at random that is fixed by this permutation. The details of this procedure are described below.

Algorithm 3.1. *Sampling unlabeled graphs uniformly at random*

```
(1) Select a permutation $\pi \in S_n$ at random.
(2) Compute $\pi^* \in S_{\binom{n}{2}}$ corresponding to $\pi$.
(3) For each cycle of $\pi^*$ select a boolean value at random.
(4) Output the graph composed by edges of cycles with value true.
```

The operation of S_n on the nodes of graphs induces an operation of $S_{\binom{n}{2}}$ on the edges of graphs. $\pi^* \in S_{\binom{n}{2}}$ in step (2) is defined as

$$\pi^*\left((i,j)\right) := \left(\pi(i), \pi(j)\right).$$

This is the key to generate a random graph fixed by π. A graph constructed this way is drawn randomly with uniform distribution from all unlabeled graphs on n nodes.

Example 3.2. Unlabeled simple graphs on six nodes
Having selected $\pi = [3\ 4\ 5\ 6\ 1\ 2] = (1\ 3\ 5)(2\ 4\ 6)$ at random, the corresponding permutation in $S_{\binom{n}{2}}$ is

$$\begin{aligned}\pi^* &= ((1,2)\ (3,4)\ (5,6))\,((1,3)\ (3,5)\ (1,5))\\ &\quad ((1,4)\ (3,6)\ (2,5))\,((1,6)\ (2,3)\ (4,5))\,((2,4)\ (4,6)\ (2,6))\,.\end{aligned}$$

Random values *true* for the cycles

$$((1,2)\ (3,4)\ (5,6))\,\text{and}\,((1,6)\ (2,3)\ (4,5))$$

would lead to the graph with edgeset

$$\{(1,2),(1,6),(2,3),(3,4),(4,5),(5,6)\},$$

the cycle graph on six nodes.

The Dixon–Wilf technique was later expanded by Wormald [53] to sample regular graphs. An extension to molecular graphs with given molecular formula was developed by Goldberg and Jerum [54]. 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 by adding trees and chains of trees (vertices of degree one or two). This strategy is similar to the processing in DENDRAL, where once cyclic substructures were generated, and connections representing the acyclic parts are added afterward (see Subsection 2.1).

3.2. **Monte Carlo and simulated annealing.** Uniformly distributed random sampling is appropriate to calculate average properties of compounds from specific compound classes, but is 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 (MC) and simulated annealing (SA) or genetic algorithms (GA) are more suitable.

Here, structures are optimized with respect to a certain target property. Any mapping from the constitutional space onto real numbers can be used as target property. Of course this mapping must be invariant with respect to atom numbering. Topological indices, group contribution calculations or potential energy are prominent examples used by Faulon in [55].

Algorithm 3.3 is extracted from [55] and outlines the principle of MC/SA. In each annealing step a molecular graph, represented by its adjacency matrix A, is given a small displacement in order to obtain a new structure, represented by A'. If the new structure is better with respect to the target property, it is accepted for the next annealing step. Otherwise it could still be accepted depending on a random decision guided by a so-called annealing schedule. The coefficient kT calculated in (g) is typically dependent on an initial coefficient, the current step number and the total number of scheduled annealing steps. Indeed the annealing schedule is the only difference between SA and MC algorithms. This procedure is repeated until a given number of annealing steps were carried out.

Algorithm 3.3. *Simulated Annealing*

```
(1) Generate an initial $A$ using a deterministic technique.
(2) For each SA step
    (a) Choose four distinct atoms $x_1, y_1, x_2, y_2$ randomly.
    (b) Set $A' :=$ Displacement $(x_1, y_1, x_2, y_2)$.
    (c) If $A'$ does not meet the chemical constraints goto (a).
    (d) Compute the cost function $e(A')$.
    (e) $\Delta e := e(A') - e(A)$.
    (f) $RN :=$ random number between 0 and 1.
    (g) Compute the coefficient $kT$ according to the
        annealing schedule.
    (h) If $\Delta e < 0$ or $RN < exp(-\Delta e/kT)$ then $A := A'$ and
        Output $A$.
```

Subroutine *Displacement* (x_1, y_1, x_2, y_2)

```
(1) Initialize $A' := A$,
    $a_{11} := A(x_1, y_1)$, $a_{12} := A(x_1, y_2)$,
    $a_{21} := A(x_2, y_1)$, $a_{22} := A(x_2, y_2)$.
(2) Choose $b_{11} \neq a_{11}$ at random so that
    $b_{11} \geq \max(0, a_{11} - a_{22}, a_{11} + a_{12} - 3, a_{11} + a_{21} - 3)$ and
    $b_{11} \leq \min(3, a_{11} + a_{12}, a_{11} + a_{21}, a_{11} - a_{22} + 3)$.
(3) Set $A'(x_1, y_1) := b_{11}$,
    $A'(x_1, y_2) := a_{11} + a_{12} - b_{11}$,
    $A'(x_2, y_1) := a_{11} + a_{21} - b_{11}$,
    $A'(x_2, y_2) := a_{22} - a_{11} + b_{11}$.
(4) Return $A'$.
```

The crucial step in this procedure is the random displacement, which can be regarded as transformation of a molecular graph in such a way that another isomer is obtained. Random displacements are implemented by modifying bond orders [56]. This includes creation of bonds in case a bond order is changed from zero to a positive value and deletion of bonds if the bond order is set to zero during the modification. Because isomers must have the same total number of bonds, when a bond order is increased, another bond order must be decreased. Hence, such a transformation implies the selection of at least two bonds, or four atoms.

The *bond order switch* is described in subroutine *Displacement* of Algorithm 3.3. Numbers of randomly selected atoms are parameter values

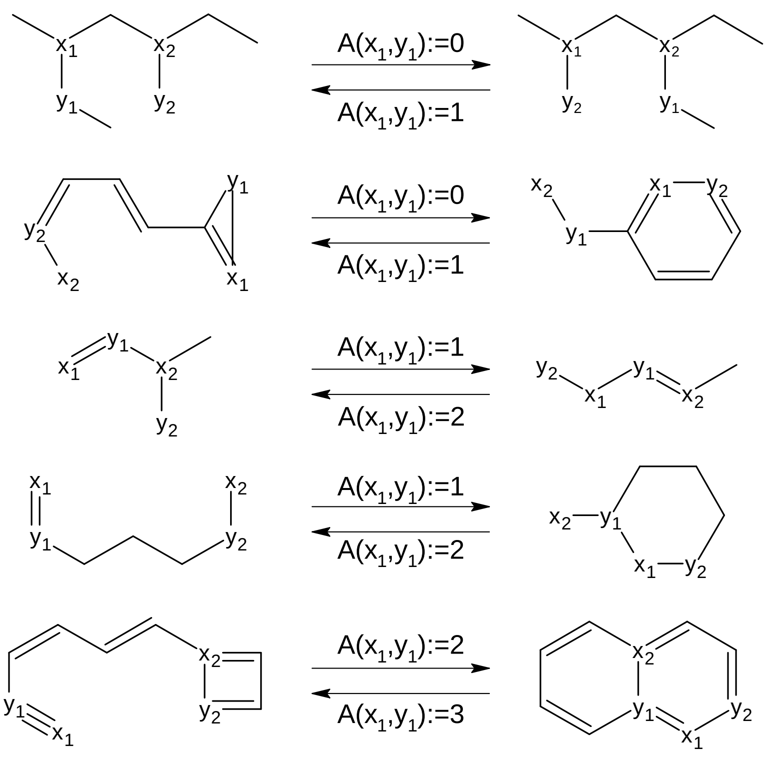


FIGURE 5. Examples of random displacements as used in Monte Carlo and simulated annealing algorithms

for this subroutine. The inequations in Step (2) reflect the fact that bond orders range from zero to three. The new bond orders assigned in Step (3) maintain the atom's valencies. In [55] it has been shown by computer experiments that all possible constitutional isomers of a given molecular formula can be reached using this bond order switch.

Example 3.4. Bond order switch
Figure 5 shows several examples of such random displacements. The new bond orders assigned to (x_1, y_1) are sketched by arrows labeled with the change in the adjacency matrix. In the upper two bond switches a bond between x_1 and y_1 is deleted and created in reverse direction. In the third and fourth bond switch the bond order changes from two to one and vice versa. The lower bond switch shows a change from triple to double bond between x_1 and y_1.

3.3. **Genetic algorithms.** Another type of stochastic structure generators are based on the technique of genetic algorithms (GA). Genetic algorithms try to simulate principles from biological evolution, such as *inheritance*, *mutation*, *crossover* (or *recombination*) and *selection.* Except for recombination, these principles have already been used in MC/SA algorithms. However, terminology is taken from biology's evolutionary theory.

A fitness function serves for the selection of structures that fit a problem specific target property well. The connectivity stack can be used as *genetic code*. The two types of structure manipulations, mutation and recombination can be seen as operations on the genetic code. Mutations can be defined like random displacements known from MC/SA. Crossover involves two parent structures and at positions where their genetic codes differ a random decision determines which parent's information should be passed to the child structure.

Algorithm 3.5 shows the construction of a new generation of structures as described in Meiler's work [57, 58]. This study was devoted to structure elucidation of small organic compounds by means of ^{13}C NMR spectra. The root–mean–square deviation between the experimental chemical shifts and the predicted chemical shifts obtained by an artificial neural network served as fitness function.

Algorithm 3.5. *Genetic Algorithm (construction of a new generation)*

```
(1) Set $i := 0$.
(2) While the number of populations $i < n$
    (a) Set $j := 0$.
    (b) While the number of molecules $j < m$
        (i) Select first parent structure at random according to
            a probability based on the fitness function.
        (ii) If random decision for recombination is $true$ then
                Select second parent and perform recombination
            else goto (iv).
        (iii) If random decision for mutation is $false$ then
                goto (v).
        (iv) Perform mutation.
        (v) If molecule is new then increase $j$ by 1.
    (c) Calculate fitting values of the molecules of the child population.
    (d) Replace the $l$ worst molecules of the child population by
        the $l$ best parents.
    (e) Increase $i$ by 1.
```

Each generation consists of a predefined number of populations n, where each population comprises m molecules. In Step (i) the first parent for recombination is chosen at random. The probability distribution for this random selection is based on the values of the fitness function in the parent population. This guarantees that better structures have higher probability to hand down their genetic information to child structures. In the next step it is decided randomly if recombination or mutation should take place. In the case of recombination a second parent is chosen, and the recombination is carried out. Otherwise the parent structure is directly passed to mutation in Step (iv). After recombination, the child structure can also still be subject to mutation,

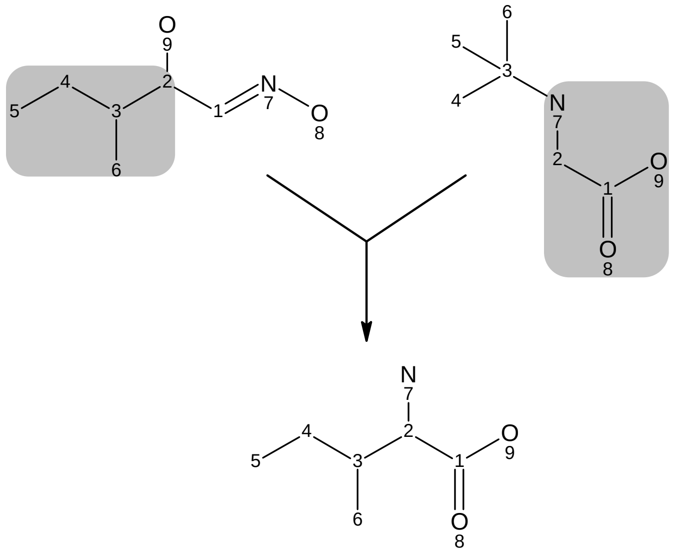


Figure 6. Example of a recombination as used in genetic algorithms

again based on a random decision. After these structure manipulation steps, fitting values of the child population are calculated. Step (d) finally prevents loosing good solutions already obtained in the parent population, via the user–parameter l.

While mutation was already known from the previous subsection, recombination is a special feature of the GA. From the chemoinformatics point of view it is interesting how this is applied to constitutional isomers. For pairs of atoms (x_i, y_i) that have the same bond order in both parent structures A and B, the child structure C also obtains this bond order for (x_i, y_i). If the bond orders differ in the parent structures, the corresponding bond order in the child structure is selected randomly from one of the parent structures.

Example 3.6. Recombination
Figure 6 depicts two parent structures A (left) and B (right) and a resulting child structure C (bottom). The parts contributed by the parent structures are highlighted grey. However the process of recombination can be better understood when looking at the genetic codes of the involved structures. Remember, the genetic codes are the upper left triangles of the adjacency matrices written in one row. The different rows of the adjacency matrices are separated by blanks in the following representation:

```
A :  10000200  1000001  101000  10000  0000  000  10  0
          ↓↓↓      ↓ ↓                            ↓
C :  10000021  1000100  101000  10000  0000  000  00  0
               ↑         ↑ ↑    ↑
B :  10000021  0000100  111100  00000  0000  000  00  0
```

We see that the genetic codes of the parent structures differ in ten positions. Arrows mark the positions in the parent structures that have to be changed in order to obtain the child structure.

But in contrast to the bond order switch introduced in Subsection 3.2 it could happen that the child structure is not necessarily chemically valid. Structures have to be checked after recombination and further bond orders might have to be changed. Another approach for recombination, which avoids this deficiency, is proposed in [59].

The introduced MC/SA algorithm and the GA do not intend to avoid isomorphic duplicates. However, since these algorithms typically aim on producing small numbers of constitutions that fit the target property well, they could easily be upgraded to avoid duplicates by using a canonical labeling algorithm and a hash map.

4. Beyond isomer enumeration

Beyond generation of isomers, the methods described in the first two sections of this chapter can be applied to generate extensive parts of the chemical space or to count and construct combinatorial libraries. In the following we outline the adaptations in algorithms required for these purposes.

4.1. **Virtual chemical space.** There are several reasons to generate large parts of the chemical space by means of computers. In [60, 61] the authors generate and examine a virtual chemical space of small molecules (up to 11 non–hydrogen atoms, elements $\mathsf{C}, \mathsf{H}, \mathsf{N}, \mathsf{O}, \mathsf{F}$) with respect to ring systems, stereochemistry, compound classes, physico–chemical properties, as well as drug– and lead–likeness.

The algorithm starts with the construction of simple graphs, subsequently introduces multiple bonds and element symbols, and ends with the generation of stereo isomers. More precisely the algorithm's pseudo code looks as follows:

Algorithm 4.1. *Virtual chemical universe up to 11 atoms*

```
(1) Generate all simple connected graphs on up to 11 vertices with
    vertex valency up to 4 (corresponding to saturated hydrocarbons).
(2) Selection of graphs with moderate ring strain using topological
    criteria and molecular mechanics that eliminate
    (a) graphs containing one or more nodes in two small (three–
        or four–membered) rings,
    (b) graphs containing a ring system with a tetravalent bridge-
        head in a small ring,
    (c) all graph–theoretically non–planar graphs,
    (d) graphs containing highly distorted centers not identified by
        topology, but with an adapted MM2 force field.
```

(3) Introduction of multiple bonds and elements $\mathsf{C}, \mathsf{H}, \mathsf{N}, \mathsf{O}, \mathsf{F}$:
  (a) Determine symmetry of the simple graphs.
  (b) Introduce double and triple bonds combinatorially with respect to symmetry in order to avoid duplicates. Bridgehead double bonds, triple bonds in rings smaller than nine and allenes are excluded as considered potentially problematic for synthesis.
  (c) Determine symmetry of the multigraphs.
  (d) Introduce element symbols combinatorially with respect to symmetry and under consideration of valency rules.

(4) Filtering for chemical stability, tautomeric and aromatic duplicates:
  (a) Structures with unstable functional groups are identified by substructure search and removed.
  (b) Tautomeric and aromatic duplicates are removed.

(5) Stereoisomer generation.

Step (1) was executed using GENG which is part of the freely available NAUTY system [40, 62]. This resulted in 843,335 simple, connected graphs. The three topological selection steps (2) (a)–(c) reduced the number of graphs from 843,335 to 16,009, the molecular–mechanics–based procedure eliminated another 283 graphs leaving a final set of 15,726 graphs. The introduction of multiple bonds in Step (3) (a) and (b) resulted in 276,220 multigraphs, and the introduction of element symbols in (3) (c) and (d) led to 1,720,329,902 molecular graphs. For the symmetry perception in Steps (3) (a) and (c) an algorithm described in [63] was used, which is based on the methods of [64], but introduces additional atomic invariants. Note that we see two typical applications of the homomorphism principle here. After removing unstable structures in Step (4) (a), 27,681,431 structures remained. Keeping only the most probable tautomer in Step (4) (b) resulted in 26,434,571 structures. Stereo isomers in Step (5) were constructed by an implementation of [65] and led to 110,979,507 configurations.

Another approach to generate molecular structures as SMILES has been proposed in [66]. The software GENSMI is based on the commercial toolkit of Daylight Chemical Information Systems, Inc. The aim of this project was to provide structures for the search for new drug candidates that involves virtual screening by evaluating protein-ligand interactions.

A study to point out the discrepancy between compounds registered in structural databases and the number of mathematically possible compounds has been published in [67]. Mathematically possible compounds consisting of $\mathsf{C}, \mathsf{H}, \mathsf{N}, \mathsf{O}$ and having a mass ≤ 150 Da were generated exhaustively and were compared with the Beilstein registry and the NIST '98 mass spectral library. As expected, it turned out that the spectral library contains only a small fraction of the compounds in the Beilstein registry, which itself represents only an even smaller fraction of the mathematically possible compounds (ratio 1:11:404976). This result emphasizes the need for structure generation software in the fields of structure elucidation and drug discovery.

The algorithm used for the generation of the chemical space in [67] is mainly based on the structure generator MOLGEN 3.5 [35, 36] which was fed with all possible molecular formulas:

Algorithm 4.2. *Molecules in silico up to 150 Da*

```
For each mass $m$ between 1 and 150 do
    for each graphical molecular formula $f$ with mass $m$ do
        generate all molecular graphs with this molecular formula $f$ using MOLGEN.
```

The program run resulted in 1405 valid molecular formulas. In this context a molecular formula is denoted being *valid*, if it belongs to at least one connected molecular graph of an organic compounds (i.e. at least one C), and where the involved elements appear with standard valencies (4, 1, 3, 2 for $\mathsf{C}, \mathsf{H}, \mathsf{N}, \mathsf{O}$ respectively). Structure generation finally resulted in 3,699,858,517 non–isomorphic molecular graphs. Detailed tables that list molecular formulas by mass and constitutions by molecular formula, as well as the numbers of structures in the above mentioned databases are included in [68].

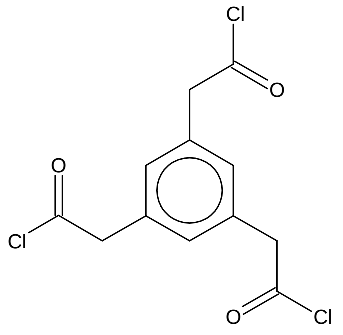
4.2. **Combinatorial libraries.** During the past decades, combinatorial chemistry has become an appropriate method to synthesize huge libraries of new compounds for biochemical screening. Especially with the development of high throughput screening technology and better bioassays, this method has gained much attraction in the drug development workflow. In order to reduce costs it is useful to plan such experiments using computer programs. Depending on the stage in the drug discovery process, combinatorial chemistry experiments may follow different strategies. In early stages, say during lead discovery, one may want to produce libraries with a high structural diversity. In later stages, for instance during lead optimization, pharmaceutical and medicinal chemists are rather interested in focused libraries.

For any of these purposes it is useful to generate the library compounds at first in silico, and apply appropriate tools in order to calculate parameters representing diversity, or virtual screening methods in order to optimize the experiment into a direction where most promising hits can be expected.

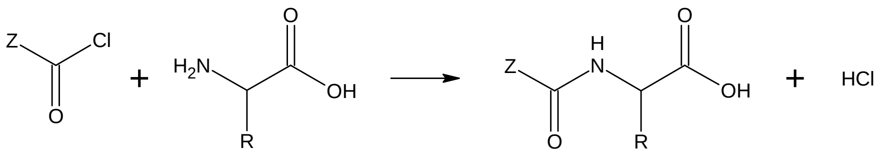
4.2.1. *Counting combinatorial libraries.* Analogously to permutational isomers we will at first show how to calculate sizes of combinatorial libraries. Most combinatorial chemistry experiments can be reduced to the situation where building blocks from a pool of substituents are connected to a central molecule, a so-called core structure with n active sites. Even reactions with multiple reaction steps, as for instance Ugi's seven component reaction can be processed this way.

If the core structure shows no symmetry with respect to the active sites, the size of the library is simply the product $\prod_{i=1}^{n} a_i$, where a_i denotes the numbers of possible substituents for active site i. However, if the central molecule shows symmetries, the situation is more complicated. But it can be solved with the methods from Subsection 1.1, as illustrated in the example below.

Example 4.3. Amidation of benzene trisacetylcychloride
As an example, we consider the exhaustive amidation of benzene trisacetylcychloride as central molecule:



Different amino acids are attached to this central molecule as shown below. An acyl chloride group reacts with an amino group in α position to the carboxyl group:



Altogether there are m^3 possible attachments of m amino acid molecules to the central molecule. But with respect to the central molecule's automorphism group, the essentially different attachments are obtained as orbits of the operation of the automorphism group applied to the set of m^3 mappings.

We face a similar situation as in Subsection 1.1. The topological automorphism group of the central molecule D_{3h}, has six permutations, the identity (1)(2)(3), three reflections (1 2)(3), (1 3)(2), (1)(2 3), and two rotations (1 2 3), (1 3 2). According to Equation 1.3 the number of orbits is

$$|m^3//D_{3h}| = \frac{1}{6}\left(m^3 + 3m^2 + 2m\right).$$

For $m = 20$ amino acids this makes a library size of 1,540 compounds, while ignoring symmetry would lead to $20^3 = 8,000$ possible attachments. As experienced earlier in Subsection 1.1, we can also apply

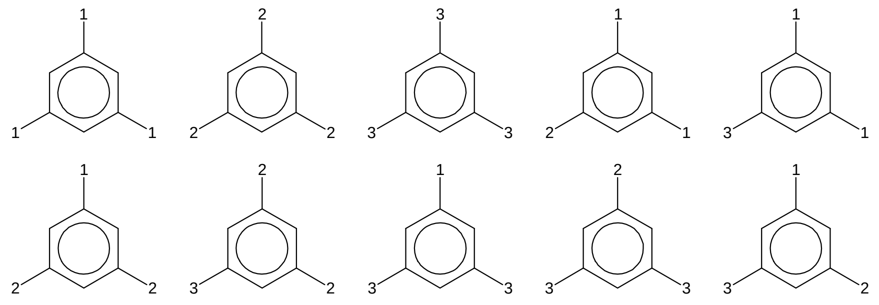
Pólya's Theorem. We obtain the cycle index

$$Z(D_{3h}) = \frac{1}{6}\left({z_1}^3 + 3z_1z_2 + 2z_3\right)$$

and for instance if we want to examine the situation of three different amino acids as possible substituents, we replace z_k by $\sum_{i=1}^{3} y_i^k$ and obtain

$$\begin{aligned} C(D_{3h}) &= \frac{1}{6}\left(\left(\sum_{i=1}^{3} y_i\right)^3 + \left(\sum_{i=1}^{3} y_i\right)\left(\sum_{i=1}^{3} {y_i}^2\right) + \sum_{i=1}^{3} {y_i}^3\right) \\ &= {y_1}^3 + {y_2}^3 + {y_3}^3 + {y_1}^2y_2 + {y_1}^2y_3 \\ &\quad + y_1{y_2}^2 + {y_2}^2y_3 + y_1{y_3}^2 + y_2{y_3}^2 + y_1y_2y_3. \end{aligned}$$

The coefficient of the monomial ${y_1}^{j_1}{y_2}^{j_2}{y_3}^{j_3}$ gives the number of library molecules where amino acid i has been attached j_i times. In this example every monomial occurs with coefficient 1. Using this knowledge, it is quite easy to construct all the library members by hand:



4.2.2. *Generating combinatorial libraries.* In general it is not possible to construct library members directly from counting series. In order to solve this problem we need to apply the relationship between permutational isomers and double cosets introduced in [8, 10]. It says that the library members created from j_1 building blocks M_1, ..., j_m building blocks M_m are in one–to–one correspondence to the set of double cosets

$$G \setminus S_n / S_{j_1} \oplus \ldots \oplus S_{j_m},$$

where G denotes the automorphism group of the central molecule and S_n, S_{j_1}, ... ,S_{j_m} the full symmetric groups of order n, j_1,...,j_m, respectively. One method to generate double cosets is using subgroup ladders [69]. An implementation MOLGEN–COMB [70], which is specialized to applications in combinatorial chemistry, uses orderly generation for the construction of double cosets.

A completely different approach to generate combinatorial libraries and reaction networks in general is to execute all possible reactions using a backtracking strategy and to filter duplicate products using a canonical labeling algorithm. [71] and [72] are just two references that describe such an approach. Chapter 11 will discuss the generation of reaction networks in more detail.

Many of the well known molecular modeling packages contain modules to generate combinatorial libraries, for instance CombiLibMaker (package: SYBYL, company: Tripos), LibraryMaker (BenchWare, Tripos), Analog Builder (Cerius^2, Accelrys), Afferent (company: MDL) or Structure Designer (ACD). For more detailed information on the features of these tools the reader is referred to the product information published by the various companies.

Acknowledgement. The author would like to thank Emma Schymanski for carefully proof–reading the manuscript.

References

[1] G. Pólya. *Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen.* Acta Mathematica, 68:145–253, 1937.

[2] G. Pólya and R. C. Read. *Combinatorial Enumeration of Groups, Graphs and Chemical Compounds.* Springer, New York, 1987.

[3] A. T. Balaban. *Enumeration of isomers.* In D. Bonchev and D. H. Rouvray, editors, *Chemical Graph Theory. Introduction and Fundamentals*, pages 177–234. Abacus Press – Gordon and Breach, New York, 1991.

[4] A. C. Lunn and J. K. Senior. *Isomerism and Configuration.* J. Phys. Chem., 33:1027–1079, 1929.

[5] J. H. Redfield. *The theory of group–reduced distributions.* Amer. J. Math., 49:433–455, 1927.

[6] A. Kerber. *Algebraic Combinatorics via Finite Group Actions.* Wissenschaftsverlag, Berlin, Heidelberg, New York, 2nd edition, 1999.

[7] C. C. Sims. *Computation with permutation groups.* In S. R. Petrick, editor, *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, pages 23–28, New York, 1971.

[8] E. Ruch, W. Hässelbarth, and B. Richter. *Doppelnebenklassen als Klassenbegriff und Nomenklaturprinzip für Isomere und ihre Abzählung.* Theor. Chim. Acta, 19:288–300, 1970.

[9] W. Hässelbarth, E. Ruch, D. J. Klein, and T. H. Seligman. *Bilateral classes.* J. Math. Phys., 21:951–953, 1980.

[10] E. Ruch and D. J. Klein. *Double Cosets in Chemistry and Physics.* Theor. Chim. Acta, 63:447–472, 1983.

[11] M. van Almsick, H. Dolhaine, and H. Hönig. *Efficient Algorithms to Enumerate Isomers and Diamutamers with More Than One Type of Substituent.* J. Chem. Inf. Comput. Sci., 40:956–966, 2000.

[12] A. Kerber, A. Kohnert, and A. Lascoux. *SYMMETRICA, an object oriented computer-algebra system for the symmetric group.* J. Symb. Comput., 14:195–203, 1992.

[13] R. C. Read. *The Enumeration of Locally Restricted Graphs II.* J. London Math. Soc., 35:344, 1960.

[14] J.-L. Faulon, D. Visco Jr., and D. Roe. *Enumerating Molecules.* In K. Lipkowitz, editor, *Reviews in Computational Chemistry*, volume 21, pages 209–286. Wiley–VCH, New York, 2005.

[15] R. W. Robinson and W. C. Wormald. *Numbers of Cubic Graphs.* J. Graph Theory, 7:436–467, 1983.

[16] J. Wang, R. Li, and S. Wang. *Enumeration of Isomers of Acyclic Saturated Hydroxyl Ethers.* J. Math. Chem., 33:171–179, 2003.

[17] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project.* McGraw–Hill Book, New York, NY, USA, 1980.

[18] H. R. Henze and C. M. Blair. *The number of structurally isomeric alcohols of the methanol series.* J. Am. Chem. Soc., 53:3077–3085, 1931.

[19] J. Lederberg. *DENDRAL-64, A System for Computer Construction, Enumeration and Notation of Organic Molecules as Tree Structures and Cyclic Graphs.* Interim Rpt. NASA CR–57029, National Aeronautics and Space Administration, 1964.

[20] C. Jordan. *Sur les assemblages des lignes.* Reine Angew. Math., 70:185–190, 1869.

[21] R. Aringhieri, P. Hansen, and F. Malucelli. *Chemical Trees Enumeration Algorithms.* Tech. Rpt. TR–99–09, Università di Pisa, 1999.

[22] J. Lederberg, G. L. Sutherland, B. G. Buchanan, E. A. Feigenbaum, A. V. Robertson, A. M. Duffield, and C. Djerassi. *Applications of artificial intelligence for chemical inference. I. Number of possible organic compounds. Acyclic structures containing carbon, hydrogen, oxygen, and nitrogen.* J. Am. Chem. Soc., 91:2973–2976, 1969.

[23] J. Lederberg. *Topological Mapping of Organic Molecules.* Proc. Natl. Acad. Sci., 53:134–139, 1965.

[24] L. M. Masinter, N. S. Sridharan, J. Lederberg, and D. H. Smith. *Applications of artificial intelligence for chemical inference. XII. Exhaustive generation of cyclic and acyclic isomers.* J. Am. Chem. Soc., 96:7702–7714, 1974.

[25] H. Brown and L. Masinter. *An algorithm for the construction of the graphs of organic molecules.* Tech. Rpt. STAN–CS–73–361, Computer Science Dept., Stanford Univ., 1973.

[26] H. Brown, L. Masinter, and L. Hjelmeland. *Constructive graph labeling using double cosets.* Tech. Rpt. STAN–CS–72–318, Computer Science Dept., Stanford Univ., 1972.

[27] H. Brown, L. Hjelmeland, and L. Masinter. *Constructive graph labeling using double cosets.* Discr. Math., 7:1–30, 1974.

[28] L. M. Masinter, N. S. Sridharan, R. E. Carhart, and D. H. Smith. *Applications of artificial intelligence for chemical inference. XIII. Labeling of objects having symmetry.* J. Am. Chem. Soc., 96:7714–7723, 1974.

[29] R. E. Carhart, D. H. Smith, N. A. B. Gray, J. G. Nourse, and C. Djerassi. *GENOA: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures.* J. Org. Chem., 46:1708–1718, 1981.

[30] R. C. Read. *Everyone a Winner*, volume 2 of *Annals of Discrete Mathematics*, pages 107–120. North–Holland Publishing Company, 1978.

[31] I. A. Faradzhev. *Generation of Nonisomorphic Graphs with a Given Degree Sequence*, pages 11–19. Algorithmic Studies in Combinatorics. NAUKA, Moscow, Russia, 1978. In Russian.

[32] I. A. Faradzhev. *Constructive Enumeration of Combinatorial Objects.* Problèmes Combinatoires et Théorie des Graphes, 260:131–135, 1978. (Colloq. Internat. CNRS, University of Orsay, Orsay 1976).

[33] R. Grund. *Construction of Molecular Graphs with Given Hybridizations and Non–overlapping Fragments.* Bayreuther Mathematische Schriften, 49:1–113, 1995. In German.

[34] M. Meringer. *Fast Generation of Regular Graphs and Construction of Cages.* J. Graph Theory, 30:137–146, 1999.

[35] C. Benecke, R. Grund, R. Hohberger, A. Kerber, R. Laue, and T. Wieland. *MOLGEN+, a Generator of Connectivity Isomers and Stereoisomers for Molecular Structure Elucidation.* Anal. Chim. Acta, 314:141–147, 1995.

[36] C. Benecke, T. Grüner, A. Kerber, R. Laue, and T. Wieland. *Molecular Structure Generation with MOLGEN, new Features and Future Developments.* Fresenius J. Anal. Chem., 358:23–32, 1997.

[37] R. Laue. *Construction of Combinatorial Objects — A Tutorial.* Bayreuther Mathematische Schriften, 43:53–96, 1993.

[38] R. Kerber, A. Laue. *Group Actions, Double Cosets, and Homomorphisms: Unifying Concepts for the Constructive Theory of Discrete Structures.* Acta Appl. Math., 52:63–90, 1998.

[39] T. Grüner, R. Laue, and M. Meringer. *Algorithms for Group Actions: Homomorphism Principle and Orderly Generation Applied to Graphs*, volume 28 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 113–122. American Mathematical Society, 1996.

[40] B. D. McKay. *Isomorph–Free Exhaustive Generation.* J. Algorithms, 26:306–324, 1998.

[41] L. A. Goldberg. *Efficient algorithms for listing unlabeled graphs.* J. Algorithms, 13:128–143, 1992.

[42] E. Luks. *Isomorphism of Graphs of Bounded Valence Can be Tested in Polynomial Time.* J. Comput. Syst. Sci., 25:42–65, 1982.

[43] R. Laue, T. Grüner, M. Meringer, and A. Kerber. *Constrained Generation of Molecular Graphs*, volume 69 of *DIMACS Series in Discrete Mathematics And Theoretical Computer Science: Graphs and Discovery*, pages 319–332. American Mathematical Society, 2005.

[44] T. Grüner, A. Kerber, R. Laue, and M. Meringer. *MOLGEN 4.0.* MATCH Commun. Math. Comput. Chem., 37:205–208, 1998.

[45] E. L. Schymanski, C. Meinert, M. Meringer, and W. Brack. *The use of MS classifiers and structure generation to assist in the identification of unknowns in effect-directed analysis.* Anal. Chim. Acta, 615:136–147, 2008.

[46] K. Funatsu, N. Miyabayaski, and S. Sasaki. *Further Development of Structure Generation in the Automated Structure Elucidation System CHEMICS.* J. Chem. Inf. Comput. Sci., 28:18–28, 1988.

[47] V. Kvasnicka and J. Pospichal. *Canonical indexing and the constructive enumeration of molecular graphs.* J. Chem. Inf. Comput. Sci., 30:99–105, 1990.

[48] M. Badertscher, A. Korytko, K.-P. Schulz, M. Madison, M. E. Munk, P. Portman, M. Junghans, P. Fontana, and E. Pretsch. *Assemble 2.0: A Structure Generator.* Chemom. Intel. Lab. Syst., 51:73–79, 2000.

[49] M. E. Elyashberg, E. R. Martirosian, Y. Z. Karasev, H. Thiele, and H. Somberg. *X–PERT: A User Friendly Expert System for Molecular Structure Elucidation by Spectral Methods.* Anal. Chim. Acta, 337:265–286, 1997.

[50] M. S. Molchanova and N. S. Zefirov. *Irredundant Generation of Isomeric Molecular Structures with Some Known Fragments.* J. Chem. Inf. Comput. Sci., 38:8–22, 1998.

[51] S. G. Molodtsov. *The Generation of Molecular Graphs with Obligatory, Forbidden and Desirable Fragments.* MATCH Commun. Math. Comput. Chem., 37:157–162, 1998.

[52] J. D. Dixon and H. S. Wilf. *The Random Selection of Unlabeled Graphs.* J. Algorithms, 4:205–213, 1983.

[53] N. C. Wormald. *Generating Random Unlabeled Graphs.* SIAM J. Comput., 16:717–727, 1987.

[54] L. A. Goldberg and M. Jerrum. *Randomly Sampling Molecules.* SIAM J. Comput., 29:834–853, 1999.

[55] J.-L. Faulon. *Stochastic Generator of Chemical Structure. 2. Using Simulated Annealing To Search the Space of Constitutional Isomers.* J. Chem. Inf. Comput. Sci., 36:731–740, 1996.

[56] J.-L. Faulon. *Isomorphism, Automorphism Partitioning, and Canonical Labeling Can Be Solved in Polynomial-Time for Molecular Graphs.* J. Chem. Inf. Comput. Sci., 38:432–444, 1998.

[57] J. Meiler and M. Will. *Automated Structure Elucidation of Organic Molecules from 13C NMR Spectra Using Genetic Algorithms and Neural Networks.* J. Chem. Inf. Comput. Sci., 41(6):1535–1546, 2001.

[58] M. Will and J. Meiler. *Genius: A genetic algorithm for automated structure elucidation from 13C NMR spectra.* J. Am. Chem. Soc., 124:1868–1870, 2002.

[59] A. Globus, J. Lawton, and T. Wipke. *Automatic Molecular Design Using Evolutionary Techniques.* Nanotechnology, 10:290–299, 1999.

[60] T. Fink, H. Bruggesser, and J.-L. Reymond. *Virtual Exploration of the Small-Molecule Chemical Universe below 160 Daltons.* Angew. Chem. Internat. Ed., 44(10):1504–1508, 2005.

[61] T. Fink and J. L. Reymond. *Virtual Exploration of the Chemical Universe up to 11 Atoms of C, N, O, F: Assembly of 26.4 Million Structures (110.9 Million Stereoisomers) and Analysis for New Ring Systems, Stereochemistry, Physicochemical Properties, Compound Classes, and Drug Discovery.* J. Chem. Inf. Model., 47:342–353, 2007.

[62] B. D. McKay. *Nauty user's guide (version 1.5).* Tech. Rpt. TR–CS–90–02, Dept. Computer Science, Austral. Nat. Univ., 1990.

[63] S. Bohanec and M. Perdih. *Symmetry of Chemical Structures: A Novel Method of Graph Automorphism Group Determination.* J. Chem. Inf. Comput. Sci., 33:719–726, 1993.

[64] D. Weininger, A. Weininger, and J. L. Weininger. *SMILES. 2. Algorithm for Generation of Unique SMILES Notation.* J. Chem. Inf. Comput. Sci., 29:97–101, 1989.

[65] J. G. Nourse, R. E. Carhart, D. H. Smith, and C. Djerassi. *Exhaustive Generation of Stereoisomers for Structure Elucidation.* J. Am. Chem. Soc., 101:1216–1223, 1979.

[66] T. I. Oprea and J. M. Blaney. *Cheminformatics Approaches to Fragment–based Lead Discovery*, volume 34 of *Methods and Principles in Medicinal Chemistry: Fragment-based Approaches in Drug Discovery*, chapter 5, pages 91–111. Wiley–VCH, 2006.

[67] A. Kerber, R. Laue, M. Meringer, and C. Rücker. *Molecules in Silico: Potential versus Known Organic Compounds.* MATCH Commun. Math. Comput. Chem., 54:301–312, 2005.

[68] M. Meringer. *Mathematical Models for Combinatorial Chemistry and Molecular Structure Elucidation.* Logos–Verlag Berlin, 2004. In German.

[69] B. Schmalz. *Verwendung von Untergruppenleitern zur Bestimmung von Doppelnebenklassen.* Bayreuther Mathematische Schriften, 31:109–143, 1993.

[70] R. Gugisch, A. Kerber, R. Laue, M. Meringer, and J. Weidinger. *MOLGEN–COMB, a Software Package for Combinatorial Chemistry.* MATCH Commun. Math. Comput. Chem., 41:189–203, 2000.

[71] G. Benkö, C. Flamm, and F. Stadler. *A Graph–Based Toy Model of Chemistry.* J. Chem. Inf. Comput. Sci., 43:1085–1093, 2003.

[72] A. Kerber, R. Laue, M. Meringer, and C. Rücker. *Molecules in Silico: A Graph Description of Chemical Reactions.* J. Chem. Inf. Model., 47:805–817, 2007.