Constrained Generation of Molecular Graphs

R. Laue, Th. Grüner, M. Meringer, A. Kerber
Lehrstuhl II für Mathematik (Informatik)
Universität Bayreuth

February 12, 2003

Abstract

The generation of molecular graphs by computer programs has undergone some changes. The development is reported with focus on various mathematical methods that are created and employed in this process. No attempt has been made to explicitely state and prove the theorems but this overview contains hints to the relevant literature. In particular, a new generator MOLGEN 4.0 is described that aims at highest flexibility in using constraints during the generation process and, thus, meeting the needs of practical applications.

keywords: Graph generation, isomorphism problem, double cosets.

1 Introduction

This article describes the background and the most important new issues of our generator for chemical graphs, MOLGEN 4.0. A more detailed version, written in German, can be found in [10].

The construction of all isomers having the same brutto formula has a long history which we will and can not report in detail. Many construction algorithms are reported in the different volumes of the journal *MATCH*, at least number 27 [16] is completely devoted to this problem. There are some points that we hold to be remarkable. Just as the representation of chemical molecules as graphs was one of the roots of graph theory, their construction was one of the challenges for the development of construction algorithms for computers. A prominent starting point was the well known DENDRAL system [20], the development of which started already in the middle and late sixties of the last century. At first, only treelike structures could be constructed until there was a breaktrough in the early seventies when a decomposition of the given brutto formula into those of cyclic substructures was found which had to be combined by bridges to get molecules with the prescribed brutto formula. All possible decompositions of this

kind could be determined by appropriate mathematical theorems prior to constructing these cyclic substructures. The DENDRAL paradigm for this was *plan, generate, test*. The cyclic structures were built after the planning phase, which is the described decomposition, from a database of cyclic structures. Their combination needed some group theory, in particular the computation of double coset representatives [32]. The handling of permutation groups in a computer was developed by C. Sims [33] shortly before to prove the existence of some sporadic simple groups. These could be used to determine permutations that were smallest with respect to some ordering in their double coset. In a final step the constructed isomers were tested against some constraints coming from spectral information. This application of double cosets came up about the same time when in theoretical chemistry Ruch, Klein et al. [30],[15],[31] described the different ways to place ligands onto a skeleton of a molecule by double cosets. So, DENDRAL was the state of the art at that time. In mathematics, the double coset technique was already mentioned by Ph. Hall [14] in 1939 in a proposal for the construction of soluble groups. It is still of general importance in construction algorithms [19].

Computers at that time were small and slow and so it was important to focus on speed and efficient use of memory. Theoretical computer science then dealt with complexity results and found graph isomorphism very challenging. The generation of isomers of course was only up to isomorphism but this problem had to be solved for practical purposes. There were some results from theoreticians like Luks showing that the problem could be solved in polynomial time for chemical graphs, [21], since they have a bounded degree of the vertices. There was another development by R. Read [29] and I. Faradzev [6, 7] who independently presented at the same time, in 1978, the technique of orderly generation. In this technique an artificial ordering is imposed on the set of graphs that have 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. The ordering can be chosen so that a given subgraph can be prescribed as being smallest [3]. This approach allowed to avoid pairwise isomorphism testing and keeping long lists of graphs in memory for comparison. It is still a popular technique for many construction problems, e.g. [25]. Though the running times show that orderly generation is fast it was only in 1992 that L. A. Goldberg [9], by adding a choice rule, obtained a proof that the results could be computed with polynomial delay. The choice rule was to only extend the already known smaller graph by a new vertex that has maximal degree in the resulting graph.

The isomorphism problem itself was of importance for building up databases of chemical structures. So, there was a big discussion about unique namings of molecules. The Morgan algorithm [27] determined a unique number characterizing the isomorphism type of the chemical structure. The chemical abstract service relies on such an approach. The actual determination of smallest representatives was improved by B. D. McKay in his Nauty [22] program. He refined

the obvious rules of classifying vertices first by their degrees and some invariants coming from the number of neighbors at certain distance and then changing the sequence of vertices only within these classes. Choosing some vertex of a class as being the first imposes a new structural information that can be used to split classes again. Then the number of possibilities of staying within the classes while renumbering the vertices is drastically reduced. A thorough perfection of this idea made the Nauty program the standard algorithm.

In our approach to generate all isomers having the same brutto formula in our MOLGEN project, we make use of nearly all of these ideas. In addition we systematically use homomorphisms to find fast generations up to isomorphism. The basic idea is that a structure preserving simplification, called a homomorphism, of a structure should map isomorphic objects onto isomorphic simplified ones. Even the isomorphism should be transported onto a corresponding isomorphism of the simpler objects.

Formally, a group G acts on two sets of objects Ω_1 and Ω_2 such that the orbits of G are just the isomorphism types of objects. A surjective mapping $\sigma : \Omega_1 \longrightarrow \Omega_2$ is required to be compatible with the two actions which means that for all $\omega \in \Omega_1$ and all $g \in G$ we have $\sigma(\omega^g) = \sigma(\omega)^g$. This σ is called a homomorphism of the action of G.

Then from a set Γ_2 of representatives from the orbits of G on Ω_2 and their stabilizers in G we obtain a set Γ_1 of representatives from the orbits of G on Ω_1. The computation consists in computing for each $\omega \in \Gamma_2$ the preimage set $\sigma^{-1}(\omega)$ and then computing a set of representatives from the orbits of the stabilizer G_ω on $\sigma^{-1}(\omega)$. These new representatives form Γ_1.

This approach considers the preimages of only one object from each orbit on Ω_2. So, only a small fraction of all preimage sets from that orbit is used. Also, the group acting is reduced to a subgroup that usually also consists of only a small fraction of all group elements. Very often, such a stabilizer is trivial such that no group action has to be taken into account at all in these cases. That means that all elements in such a preimage set are pairwise non-isomorphic. Then one even needs not list these elements but only employs an unrank function for producing an object with a prescribed ranking number on demand.

As an example consider simple graphs as simplifications of multigraphs, the simplification being just to forget the edge multiplicities. Then all renamings of the vertices are compatible with this simplification. If we want to obtain the representatives of the isomorphism types of multigraphs we may first determine a set of representatives Γ_2 of the isomorphism types of simple graphs and their automorphism groups which are the stabilizers in the full symmetric group on the set of vertices. Then the edge set of each simple graph γ from Γ_2 is mapped into the set of of multiplicities. The orbits of the automorphism group of γ on the corresponding set of mappings then yield the isomorphism types of multigraphs that simplify to γ. As nearly all simple graphs have a trivial automorphism group in all these cases all multigraphs resulting from them have different isomorphism types.

If this technique is applied in several steps, enormous numbers of objects can be classified up to isomorphism.

In a generation of chemical graphs, for example the following simplifications can be employed:

- forgetting atom names,
- forgetting bond multiplicities,
- forgetting vertices of degree one,
- replacing vertices of degree 2 by an edge connecting their neighbors directly,
- omitting bridges.

The DENDRAL strategy relied on these simplification steps, only the principle was not generally worked out, see a more detailed description in [17]. We have tried to push this approach of simplifying by homomorphisms to an extreme by constructing graphs with a prescribed degree sequence from regular graphs as the most simple graphs [11]. This required a decomposition strategy different from that of DENDRAL, allowing to iterate the decomposition in an unbounded number of steps. Actually, like in the approach of Goldberg, we singled out vertices of a fixed degree for making extensions. We decomposed a graph into the subgraph spanned by the vertices of the fixed degree, say the largest degree, and that spanned by the remaining vertices. The resulting possible degree sequences of the subgraphs could be determined by using Gale and Ryser's theorem for the existence of 0/1-matrices with given row and column sums. This theorem allows to find out whether two degree sequences of vertex disjoint subgraphs can result from deleting edges of a graph with the given degree sequence prior to constructing the graphs. The rule to take all vertices of a fixed degree implies that the restriction to the subgraphs is compatible with isomorphisms. The resulting algorithm was able in many cases to *determine* up to 10^{30} graphs with a given degree formula on 50 vertices *up to isomorphism* within 10 seconds on a Pentium II computer. The computation was stopped because the full number of isomorphism types could never have been reached even with this computation speed. Of course these graphs were not really constructed but an unrank function was set up which immediately could output any of the determined graphs on demand, needing only its rank in the solution space.

The idea of restricting extensions to those where the new elements are taken from a certain orbit of the automorphism group was then extracted by B. D. McKay [23] and transported to his generation strategies. A very nice description of this approach is given by G. Brinkmann in [2].

Nearly independent of these algorithmic developments chemists built their own software to solve the problem of generating isomers to a given brutto formula and further constraints. The most notable systems are CHEMICS [8], ASSEMBLE [28], [18].

These systems usually could not compete with the speed of the mathematical software and often were neither complete nor redundancy free, with the exception of [4] and [26]. They involved a lot of chemical knowledge and had the big advantage that the users were not overloaded with masses of results. Of course the mathematician complains that he does not know on which grounds the selection is done, often relying on the current content of a database. But chemists found just those results that they expected and did not have to look at too many mathematically possible solutions that seemed to be of no relevance to chemistry.

The point of this approach is that it meets the needs of the ordinary user to get only very few results, favorably only the one structure that he was experimenting with. We should be aware of the fact that also our mathematical colleagues prefer to look at objects that have some unusual properties and are happy if they find that such a thing is unique, like the big Witt design [34]. So, the above success in generating astronomical numbers of objects in surprisingly short time on a relatively small machine does not really meet the need or taste of the scientist who originally came up with the problem. The actual set of objects that can be generated in a short time often cannot be tested for further constraints after their construction. Our experience with the MOLGEN project led us to the conclusion that we better listen to the customers and allow a lot of constraints to be exploited already during the construction.

A first attempt allowed to specify several important constraints that could be tested during the generation and others had to be tested after the complete object was available. The focus still was laid on the construction speed. A fixed generation strategy used the above explained techniques of simplifying by homomorphisms, using double coset representatives for gluing partial objects together, and orderly generation. An important point was made by R. Grund [13] who delayed isomorphism testing several steps and preferred to test simpler properties first. Only the survivors then had to undergo this elaborate step. If some candidate solution then turns out not to be canonical, the earliest extension step is determined where this could have been detected in the generation procedure. All other candidates that also are extensions of this partial object then also cannot be canonical and are rejected. Since there were usually by far too many objects after all constraints had been tested, the user could interrupt the generation, look at a visualization of the solutions obtained so far. Often, the user finds obvious unwanted properties in many of these objects. So, new additional constraints can be directly drawn from the previous output for the next run of the generator.

This chemical graph generator by chemists still was considered too much oriented towards the mathematical goals. We had to admit by far more types of constraints to meet the information that comes out of different kinds of spectra, in particular NMR-spectra. Since the constraints vary a lot, depending on the actual application, we needed a new paradigm of a construction strategy, the *constraint driven generation* [10]. We needed a very flexible way of constructing objects, oriented towards exploiting the actual set of given constraints as early as possible in the various construction steps. So, in each step we could compare a set of possible strategies

to follow for extending the present partial object. This idea already appeared in a paper by B.D. McKay, W. Myrvold, and J. Nadon [24] on the construction of $(3, 9)$-cages. These authors extend a graph by adding a new edge at a best position, speeding up the generation this way. Our goal instead is not ultimate speed but highest flexibility. A heuristic method has to decide which alternative makes best use of the actual constraints. Then that choice is pursued until in a next step this procedure is iterated. As before, testing for isomorphism is delayed according to the expectation that only few results will survive the cheaper tests for other constraints. In comparison to the previous version, the generation often is much slower. This may mean that still some better understanding is needed, since the approach by McKay, Myrvold, and Nadon just gained speed from the new flexibility. On the other hand, this approach is able to take into account many more types of constraints than our previous generators, like ionisations, subunits, neighborhood restrictions, hybridizations, number of signals for C-atoms in a NMR-spectrum, and various ring structures. In addition, an important advantage concerning all kinds of subgraph restrictions is that intervals may be prescribed for the corresponding parameters instead of fixed values.

Besides the flexibility to adopt to various constraints, the software is easily extendible to further tests of constraints. The new constraints should be of a type comparable to the already existing ones. A simple call to the test function should return yes or no for the actual partial object in order to proceed with the extension or to stop. It still has to be decided where best to perform the new test, that is where to call the new test function.

Of course, a broader mixture of methods might be faster still with the same ability to use many constraints. So, it will be a challenge to import the above sketched faster methods into the process of adopting the generation strategy to the set of actually imposed constraints.

2 Generation under Constraints

There are two choices for the generating strategy, orderly generation and constraint generation. Among the approaches sketched in the introduction, the orderly generation is most easily adopted to some constraints. The reason for its flexibility is that one can freely define new orderings on the set of graphs that have to be generated. So, the entries of an adjacency matrix may be arranged in any sequence for a lexicographical comparison. We define the sequence so that building up the chemical graphs by adding entries to a partially filled adjacency matrix fills the places in decreasing positions. Then the smaller subgraphs corresponding to the already filled matrix are lexicographically smaller than the extensions obtained by adding new entries. We may thus first fill in prescribed subgraphs or run through a small number of choices corresponding to some prescribed constraint. The choice of that constraint is done after some heuristics. After each insertion of an entry further constraints may be tested for compatibility.

Also, at certain steps a minimality test has to be made according to the orderly generation strategy.

The second approach is to run via backtracking through the extension steps and, in each step, test the candidate partial structures against the given constraints. Of course, the constraints are defined on the full structures. So, we have to restrict constraints to substructures which often is the most difficult problem. Only few candidates are supposed to survive the various tests and only these are transformed into canonical form by an approach similar to B.D. McKay's. We give a list of the presently implemented constraints in the last section. Here we want to show examples of their implementation via testing restrictions of constraints.

A *prescribed substructure* may be only partly known. So, the vertices of a subgraph may be labelled by a set of candidate atom names and an edge may be labelled by a set of possible multiplicities. An embedding is legal if the actual labels in the embedded structure occur in the set of possibles labels at that place.

There are a lot of special cases ranging from exact specifications of neighborhoods of C-atoms and assignments of numbers of H-atoms bonded to certain atoms to very loose distance information. From NMR-spectra often information can be obtained on atoms having a certain small distance from a subgraph. So, certain pairs of vertices may have a distance within specified intervals.

A classical information on subgraphs is that the subgraph must be contained in the graph. Here we have to distinguish between non-overlapping substructures, known as macroatoms, and those that may overlap. In addition, if there are free bondings connecting the vertices of the subgraph to the vertices of the graph then it has to be specified whether these bondings may be added within the subgraph, the case of *open subgraphs*, or only connect to vertices outside the subgraph, the case of *closed subgraphs*.

Prescribed substructures may be inserted into the adjacency matrix first. This is easy for macroatoms. For possibly overlapping prescribed substructures at least one can be inserted, a largest one is chosen for that. Then possible overlapping can be analysed by searching for isomorphic common substructures. This is done in a breadth first search by comparing neighborhoods of atoms. Necessary conditions for overlapping can be used to detect non-overlapping parts. The handling of many possibly overlapping parts is by running depth first search through the solution space. Then, further constraints may reduce the number of actually searched alternatives.

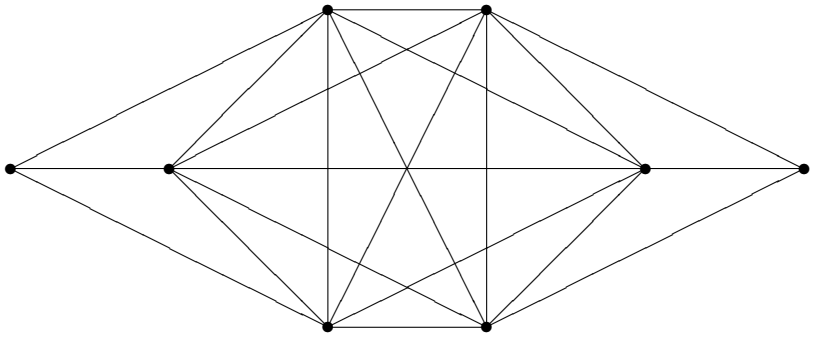
After having inserted a subgraph, the atom names and known neighboring H-atoms, and possibly prescribed hybridizations, are assigned to some atoms. This will not be unique, creating disjoint subproblems for the choices is necessary.

Now new vertices and edges are added by filling free places of the adjacency matrix. Different from earlier generators, this generator has no fixed sequence of places to continue with. The next

places to fill will belong to a new vertex that extends a partially known prescribed substructure. Its row is filled while testing against further constraints. Here the known partial structure and the new entry are handed over to test functions that may interdict that extension. So, all kinds of tests can be called here.

We want to discuss in more detail a restriction that usually comes from some NMR-spectra. There C-atoms that are in the same orbit of the automorphism group will yield the same peaks in the spectrum. The restriction then comes from the fact that there are fewer peaks than atoms of the corresponding type. So, the number of orbits may be prescribed but the automorphism group is not known. The idea which allows to exploit this information is to check during the generation process how many atoms of that type must lie in pairwise different orbits because they have some non-isomorphic neighborhoods. A graph is formed where a pair of vertices of the candidate structure that have isomorphic neighborhoods is represented by an edge. The possible orbits of the automorphism group correspond to cliques. An orbit partition then corresponds to a covering of the graph with disjoint cliques. The restriction that only few orbits exist then demands that a covering by only few cliques must exist. Not all such coverings by cliques have the same number of cliques as can be seen from the example shown. So, backtracking is used to find these coverings. The existence of such a covering is only a necessary condition. So, the final results still have to be analyzed whether they really have the prescribed number of orbits. The automorphism group is determined anyway as a byproduct of the computation of the canonical form.

Clique Covering



Two different coverings by cliques:

- Middle 6-clique and the two remaining 1-cliques
- Left half and right half both are 4-cliques

In this example, the first covering has 3 components while the second one has only 2 components.

This constraint turns out to reduce the number of candidate structures very effectively. It also can be checked early in the generation process. This can be deduced from the reduction of the computation time when prescribing smaller numbers of orbits. Mathematicians may want to use this constraint in order to find regular graphs with a transitive automorphism group without specifying the group. So, for example, there is only one connected regular graph of degree 4 on 26 points with a transitive automorphism group and of girth 6. It is obtained in less than a second on a PC.

3 MOLGEN 4.0

The MOLGEN project now runs since 16 years. There have been several previous versions, which are used worldwide in chemical industry and many university institutes. One of them was awarded the German-Austrian university software award in 1993 for Chemical software. While the previous version which was optimized with respect to generation speed had graphical user interfaces for many platforms, a GUI for the present new version MOLGEN 4.0 is only available for UNIX stations. The obvious reason is that we focus on research and do not have enough manpower to steadily meet all changing requirements. The research was supported by the German ministery BMBF in various projects. We are grateful for that steady funding making it possible to experience practical needs with a university software product.

The following constraints can be used in the generator MOLGEN 4.0:

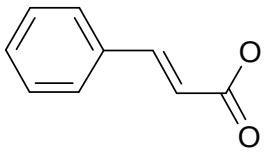
- brutto formula with intervals for the number of atoms of each type (obligatory)
- specific atom types having a name, a valency, a charge, a radical position
- non-overlapping fragments
- possibly overlapping prescribed fragments
- forbidden fragments
- surroundings of fragments in various ways
- subunits (i.e. connected substructures given by a molecular formula)
- distribution of H-atoms
- hybridization
- number of cycles of various lengths

- number of bondings of a fixed multiplicity
- number of ^{13}C-NMR signals

The following table shows by an example how the size of the set of structures returned by MOLGEN is reduced by stepwise adding constraints. We start defining only the brutto formula. CPU times refer to a PC P4, 2.53GHz running Windows 2000.

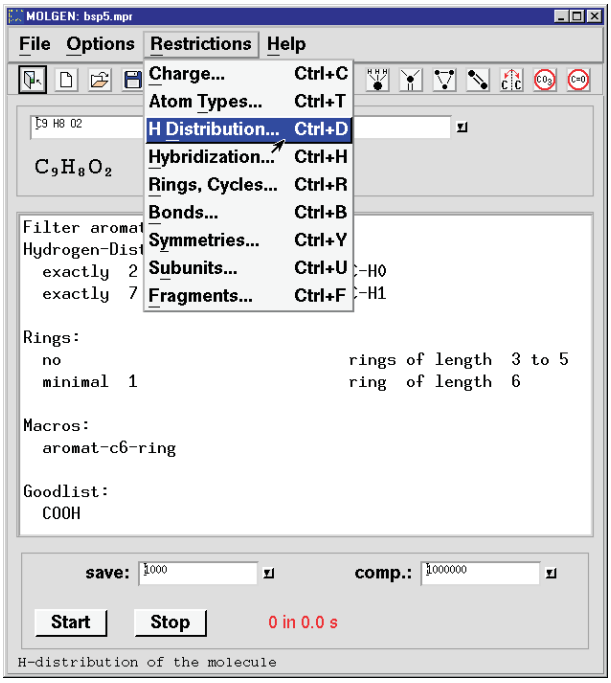
Constraints		Structures	CPU time
Brutto formula:	$C_9H_8O_2$	9,989,568	2689.9 s
H distribution:	2× C		
	7× CH	71,799	22.8 s
Rings:	no rings of length 3–5		
	at least one 6-ring	557	2.9 s
Macro:	aromatic 6-ring	31	0.1 s
Goodlist:	COOH	1	0.1 s

At the last step there remains only one structure that fulfills all the restrictions, the so called cinnamic acid:

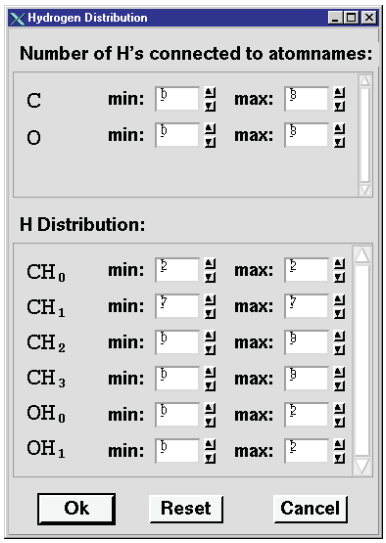


An even more sophisticated example which also uses hybridizations and subunits is published in [12]. We add some screen shots of the graphical user interface of MOLGEN 4.0 illustrating how the constraints of the above example are specified. The MOLGEN main window shows the brutto formula as well as a summary of all constraints from the individual constraint windows.

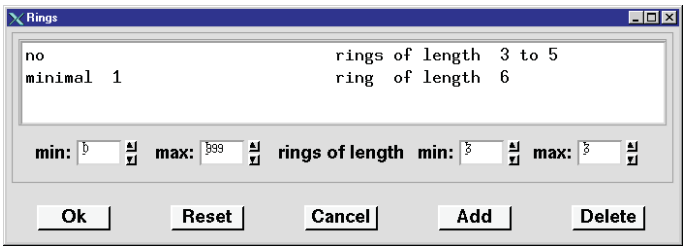
For the above example the MOLGEN main window looks like this:



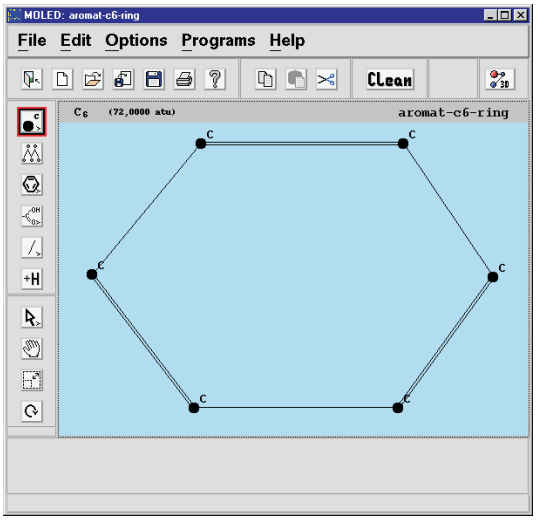
The constraints are defined in dialogues that can be reached from the main window by menus or the toolbar. Here we have the dialogue window for the hydrogen distribution

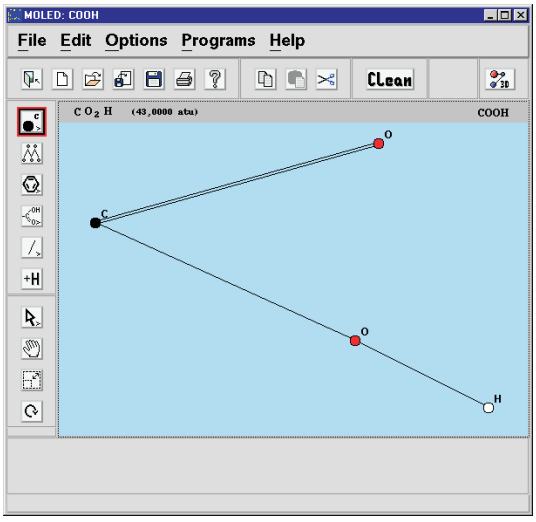


and the ring restrictions

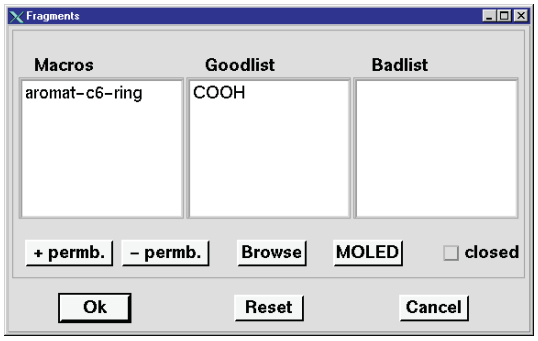


For the specification of fragments you can use the graphical editor MOLED, which is included in the package:

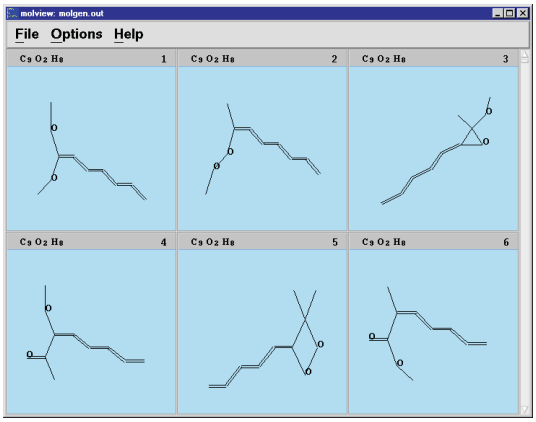


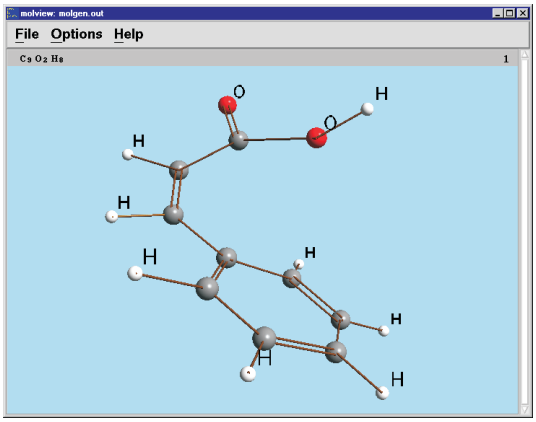


Of course you can use any other chemical structure editor, which is able to export structures in MDL molfile format. Having drawn your fragments, you can express constraints concerning the fragments. You can specify prescribed fragments as (non-overlapping) macros or by inserting them in the goodlist. Forbidden fragments must be inserted into the badlist:



The output of our structure generator, MOLGEN is displayed by a molecule viewer, which offers two- and three-dimensional visualization:





On the left there are some arbitrary isomers of $C_9H_8O_2$, which were generated without any constraints. On the right picture we see again the cinnamic acid, this time with a three dimensional layout obtained by an MM2 force field optimization[1], which is also included. A more detailed description of all MOLGEN 4.0 features is available online at `www.molgen.de`.

Acknowledgement
We are grateful to the German Federal Ministery BMBF and the Deutsche Forschungsgemeinschaft for financial support of our applied and theoretical research.

References

[1] N.L. Allinger,: MM2. A Hydrocarbon Force Field Utilizing V_1 and V_2 Torsional Terms. *J. Am. Chem. Soc.* **99** (1977), 8127–8134.

[2] G. Brinkmann: Isomorphism rejection in structure generation programs. *Discrete Mathematical Chemistry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **51** (2000), 25–38.

[3] C.J. Colbourn, R.C. Read: Orderly algorithms for generating restricted classes of graphs. *Journal of Graph Theory* **3** (1979), 187-195.

[4] M.E. Elyashberg, E.R. Martirosian, Y.Z. Karasev, H. Thiele, H. Somberg: X-pert: A user friendly expert system for molecular structure elucidation by spectral methods. *Analytica Chemica Acta* **337** (1997), 265-286.

[5] M.E. Elyashberg, Y.Z. Karasev, E.R. Martirosian, H. Thiele, H. Somberg: Expert systems as a tool for the molecular structure elucidation by spectral methods. Strategies of solution to the problems. *Analytica Chemica Acta* **348** (1997), 443-463.

[6] I.A. Faradzhev: Generation of nonisomorphic graphs with a given degree sequence(russian). *In Algorithmic Studies in Combinatorics* NAUKA Moscow (1978), 11–19.

[7] I.A. Faradzhev: Constructive enumeration of combinatorial objects. *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay 1976)* **260** (1978), 131-135.

[8] K. Funatsu, N. Miyabayaski, S. Sasaki: Further development of structure generationin the automated structure elucidation system CHEMICS. *J. Chem. Inf. Comput. Sci.* **28** (1988), 18-28.

[9] L.A. Goldberg: Efficient algorithms for listing unlabeled graphs. *J. Algorithms* **13** (1992), 128-143.

[10] T. Grüner: Strategien zur Konstruktion diskreter Strukturen und ihre Anwendung auf molekulare Graphen. *MATCH* **39** (1999), 39-126.

[11] T. Grüner, R. Laue, M. Meringer: Algorithms for group actions applied to graph generation. *Groups and Computation II, Workshop on Groups and Computation*, june7–10, 1995,113–123, L. Finkelstein, W. M. Kantor, ed., *DIMACS* **28**, AMS 1997.

[12] T. Grüner, A. Kerber, R. Laue, M. Meringer: MOLGEN 4.0. *MATCH* **37** (1998), 205-208.

[13] R. Grund: Konstruktion schlichter Graphen mit gegebener Gradpartition. *Bayreuther Math. Schr.***44** (1993).

[14] P. Hall: On groups of automorphisms. *J. reine angew. Math.* **182** (1940), 194-204, esp. 199.

[15] W. Hässelbarth, E. Ruch, D.L. Klein, T.H. Seligman: Bilateral classes. *J. Math. Phys.* **21** (180), 951-953.

[16] A. Kerber,*editor* : Generatoren für molekulare Graphen, Nomenklaturfragen. *MATCH* **27** (1992).

[17] A. Kerber, R. Laue: Group actions, double cosets, and homomorphisms: unifying concepts for the constructive theory of discrete structures. *Acta Applicanda Mathematicae* **52**(1998), 63–90. Conference proceedings of the Kaloujnine memorial conference held near Dresden 1994.

[18] V. Kvasnicka, J. Pospichal: Canonical indexing and the constructive enumeration of molecular graphs *J. Chem. Computer Science* **30** (1990), 99–105.

[19] R. Laue: Construction of combinatorial objects - A tutorial. *Bayreuther Math. Schr.* **43** (1993), 53–96.

[20] R.K. Lindsay, B.G. Buchanan, E.A. Feigenbaum, J. Lederberg: Applications of artificial intelligence for organic chemistry: The Dendral Project. *McGraw-Hill, New York* (1980).

[21] E. Luks: Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comp. Sys. Sci.* **25** (1982), 42-65.

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

[23] B.D. Mckay: Isomorph-free exhaustive generation. *J Algorithms* **26** (1998), 306-324.

[24] B.D. Mckay, W. Myrvold, J. Nadon: Fast backtracking principles applied to find new cages. *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (1998), 188–191.

[25] M. Meringer: Fast Generation of Regular Graphs and Construction of Cages. *Journal of Graph Theory* **30** (1999), 137–146.

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

[27] H. L. Morgan.hs: The Generation of Unique Machine Description for Chemical Structures - A Technique Developed at Chemical Abstracts Services. *J. Chem. Doc.* **5** (1965), 107-113.

[28] M.E. Munk: ASSEMBLE: User manual. Version April 1995.

[29] R.C. Read: Every one a winner. *Annals of Discrete Math.* **2** (1978), 107-120.

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

[31] E. Ruch, D.J. Klein Double cosets in chemistry and physics. *Theor. Chim. Acta* **63** (1983), 447-472.

[32] B. Schmalz: Verwendung von Untergruppenleitern zur Bestimmung von Doppelnebenklassen. *Bayreuther Math. Schr.* **31** (1990), 109-143.

[33] C.C. Sims: Computation with permutation groups. *Proc. Sec. Symp. Symb. Alg. Manip.* S.R. Petrick, editor ACM (1979), 23-28.

[34] E. Witt: Die 5-fach transitiven Gruppen von Mathieu. *Abh. Math. Seminar Univ. Hamburg* **12** (1938), 256-264.