J. Transparent Formal Methods for Reducing the Combinatorial Wealth of Conceivable Solutions to a Chemical Problem—Computer-Assisted Elucidation of Complex Reaction Mechanisms. *Anal. Chim. Acta* **1990**, *235*, 155-161.

(2) (a) Corey, E. J.; Long, A. K.; Rubenstein, S. D. Computer-Assisted Analysis in Organic Synthesis. *Science* **1985**, *228*, 408-418. (b) Gund, P.; Grabowski, E. J. J.; Hoff, D. R.; Smith, G. M.; Andose, J. D.; Rhodes, J. B.; Wipke, W. T. Computer-Assisted Synthetic Analysis at Merck. *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 88-93.

(3) Gasteiger, J.; Hutchings, M. G.; Christoph, B.; Gann, L.; Hiller, C.; Löw, P.; Marsili, M.; Saller, H.; Yuki, K. A New Treatment of Chemical Reactivity: Development of EROS, an Expert System for Reaction Prediction and Synthesis Design. *Top. Curr. Chem.* **1987**, *137*, 19-73.

(4) Schubert, W. ASSOR—Allgemeines Simulationssystem Organischer Reaktionen. *Informal Commun. Math. Chem. (Match)* **1979**, *6*, 213-255.

(5) (a) Hendrickson, J. B. A General Protocol for Systematic Synthesis Design. *Top. Curr. Chem.* **1976**, *62*, 49-172. (b) Hendrickson, J. B.; Grier, D. L.; Toczko, A. G. A Logic-Based Program for Synthesis Design. *J. Am. Chem. Soc.* **1985**, *107*, 5228-5238. (c) Hendrickson, J. B. Approaching the Logic of Synthesis Design. *Acc. Chem. Res.* **1986**, *19*, 274-281. (d) Hendrickson, J. B.; Huang, P. Multiple Construction in Synthesis Design. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 145-151. (e) Hendrickson, J. B.; Toczko, A. G. SYNGEN Program for Synthesis Design: Basic Computing Techniques. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 137-145.

(6) Ugi, I.; Bauer, J.; Brandt, J.; Friedrich, J.; Gasteiger, J.; Jochum, C.; Schubert, W.; Dugundji, J. Computer Programs for the Deductive Solution of Chemical Problems on the Basis of a Mathematical Model—A Systematic Bilateral Approach to Reaction Pathways. In *Computational Methods in Chemistry*; Bargon, J., Ed.; Plenum: New York, 1980; pp 275-300.

(7) Wipke, W. T.; Rogers, D. Artificial Intelligence in Organic Synthesis. SST: Starting Material Selection Strategies. An Application of Superstructure Search. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 71-81.

(8) (a) Ugi, I.; Gillespie, P. D. Stoffbilanz-erhaltende Synthesewege und semi-empirische Syntheseplanung mittels elektronischer Datenverarbeitung. *Angew. Chem.* **1971**, *83*, 982-985; *Angew. Chem., Int. Ed. Engl.* **1971**, *10*, 915-918. (b) Ugi, I.; Gillespie, P. D. Beschreibung chemischer Systeme und ihrer Umwandlungen durch BE-Matrizen und ihre Transformationen. *Angew. Chem.* **1971**, *83*, 980-981; *Angew. Chem., Int. Ed. Engl.* **1971**, *10*, 914-915. (c) Dugundji, J.; Ugi, I. An Algebraic Model of Constitutional Chemistry as a Basis for Chemical Computer Programs. *Top. Curr. Chem.* **1973**, *39*, 19-64.

(9) Bauer, J.; Fontain, E.; Forstmeyer, D.; Ugi, I. Interactive Generation of Organic Reactions by IGOR2 and the PC-assisted Discovery of a New Reaction. *Tetrahedron Comput. Methodol.* **1988**, *1*, 129-132.

(10) Bauer, J. IGOR2: A PC-Program for Generating New Reactions and Molecular Structures. *Tetrahedron Comput. Methodol.* **1990**, in press.

(11) Zefirov, N. S. An Approach to Systemization and Design of Organic Reactions. *Acc. Chem. Res.* **1987**, *20*, 237-243.

(12) (a) Koca, J.; Kratochvil, M.; Kunz, M.; Kvasnicka, V. Mathematical Model of Organic Chemistry. VI. Valence States of Atoms and Their Conversions. *Collect. Czech. Chem. Commun.* **1984**, *49*, 1247-1261. (b) Koca, J.; Kratochvil, M.; Kvasnicka, V.; Matyska, L.; Pospichal, J. Synthon Model of Organic Chemistry and Synthesis Design. *Lect. Notes Chem.* **1989**, *51*, 78-84.

(13) Schubert, W.; Ugi, I. Constitutional Symmetry and Unique Descriptors of Molecules. *J. Am. Chem. Soc.* **1978**, *100*, 37-41.

(14) (a) Augelmann, G.; Fritz, H.; Rihs, G.; Streith, J. An Unexpected Addition Product of Nitrosobenzene with Pyran-2-thione. *J. Chem. Soc., Chem. Commun.* **1982**, 1112-1113. (b) Wochner, M.; Brandt, J.; v. Scholley, A.; Ugi, I. Chemical Similarity, Chemical Distance, and its Exact Determination. *Chimia* **1988**, *42*, 217-225. (c) Defoin, A.; Augelmann, G.; Fritz, H.; Geffroy, G.; Schmidlin, C.; Streith, J. Cycloaddition of 2*H*-Pyran-2-thiones with Nitroso Derivatives. An Unexpected Cycloaddition-Rearrangement Reaction. *Helv. Chim. Acta* **1985**, *68*, 1998-2014. (d) Defoin, A.; Geffroy, G.; Le Nouen, D.; Spileers, D.; Streith, J. Cascade Reactions. A Simple One-Pot Synthesis of the Mitomycin Skeleton. *Helv. Chim. Acta* **1989**, *72*, 1199-1215.

Generation and Enumeration of Carbon Skeletons

JAMES B. HENDRICKSON* and CAMDEN A. PARKS

Department of Chemistry, Brandeis University, Waltham, Massachusetts 02254-9110

Received July 9, 1990

Two different programs were written to enumerate all possible carbon skeletons, i.e., saturated hydrocarbons. These enumerations were carried out fully to 11 carbon atoms and for limited numbers of rings to 16 carbon atoms. The results of the two programs were consistent with each other as well as with such previous enumerations as were available. A program to show graphically the stored skeletons was also prepared. The skeletons of commercially available starting materials are a minute subset of the total. Various ways of classifying the data into subsets are illustrated as are superstructure searches from seed skeletons.

INTRODUCTION

Enumeration of structural isomers of organic compounds is a problem which has fascinated chemists and mathematicians for over a century. The first attempts at solutions concentrated on the development of mathematical formulas which would give the numbers of isomers. This line of work began with Cayley[1] in 1874 and was later taken up by Henze and Blair,[2] Pôlya,[3] and Read.[4] The development of the computer allowed a different approach, which involved the generation and subsequent counting of graphs representing the isomers. Knop[5] used this method to enumerate the alkanes as well as the substituted alkanes. Masinter's program[6a] generates all compounds for a given empirical formula, and Read[7] developed a program which generated all graphs for up to 10 vertices.

Our interest in the problem arose from the construction of the SYNGEN program for synthesis design.[8] This logic-centered program aims to find all shortest, most efficient synthetic paths to a given target structure. In order to simplify the vast synthesis tree of possibilities, SYNGEN first determines all convergent assemblies for the carbon skeleton of the target from four starting skeletons, and these must be contained in a catalogue of actual starting materials. Having established a base catalogue of about 6000 commercial compounds, we questioned what proportion of all possible carbon skeletons were actually represented in this catalogue. Since we had developed an efficient canonical numbering system for rapid computer comparison of skeletons,[9] we undertook to create a program to generate all possible carbon skeletons, to act as an "ideal catalogue" of all starting skeletons. When our work was essentially complete, a paper appeared from Kvasnička and Pospichal,[10] who generated all connected graphs of up to eight vertices.

Our interest is to find all carbon skeletons. A carbon skeleton is the connected framework of a compound which has been stripped of all π-bonds and all non-carbon atoms; any chirality which may be present in this framework is ignored. Defined in this manner, skeletons are equivalent to connected topological graphs of degree no greater than four. This will be a subset of the enumerations of Read[7] and Kvasnička and Pospichal,[10] since these include graphs of higher degree. It will also be a subset of the Masinter enumeration,[6a] which includes double- and triple-bond isomers of the saturated polycyclics. We hoped to extend our enumeration to the 16-carbon size limit of the SYNGEN catalogue as well as to produce graphic output of the skeletons produced.

All computer programs that generate structural isomers of chemical compounds must generate all the nonredundant isomers in a reasonable amount of time. This requires that there be a method of recognizing redundancy and that the total number of graphs which must be checked for redundancy be reduced as much as possible. The usual method of checking redundancy involves the use of a canonical representation of the isomer. There are two ways in which this can be done.

The first way is to generate the canonical representation of a given isomer and see whether that representation has already been generated. This usually involves storing all previously generated unique representations, searching that set to see whether the new representation appears there, and, if it does not, adding it to the set of unique representations. The main difficulty with this method is that the data sets quickly become so large that they cannot be stored in dynamic memory and so must be stored on disk. Searching the data sets then requires a very large number of time-consuming disk access operations. If an attempt is made to reduce this search time by ordering the data set, every new entry requires a rearrangement of the data set and so still a great deal of disk access time.

The second way to use canonical representations to identify redundancy involves checking each newly generated isomer first to see if it is already canonically labeled. If the algorithm used to generate the isomer representations never generates any representation more than once, the program can simply throw out any representation which is not already in canonical form. In this way many (in principle, $n! - 1$) generated representations of any given skeleton need never be passed to storage; only the canonical labelings are saved. This method has the advantage that it only requires one disk access operation for each unique isomer, making it a great deal faster than the first method as well as making it practical to put large data sets on tape.

Our canonical representation method corresponds to the maximal upper-right triangle of the adjacency matrix.[9] For a graph of n vertices, there are $n!$ different labelings, or numberings, of the graph, each of which has associated with it an adjacency matrix. Each adjacency matrix can, without loss of information, be represented by the binary string formed by concatenating the rows of the upper-right triangle of the matrix. If each such binary string is interpreted as an integer, there is only one integer which is the maximum among all binary strings representing a particular skeleton. The string corresponding to this maximum integer is the maximal string, and the adjacency matrix for this string is the maximal matrix; these constitute a unique canonical numbering.[11]

In this paper we describe two programs for generating and enumerating carbon skeletons. The SKEL_GEN program is the main one and the faster of the two; it follows the second procedure above. The algorithm is designed to create rigorously all possible adjacency matrices by generating all combinations of one/zero entries for each successive row. However, many such combinations are not compatible with maximal numbering and so are deleted before the full matrix is created. When a full matrix has been generated, the program checks to see whether it corresponds to the unique maximal matrix, and if it does, it is stored.

The second program (SKEL_GROW) uses previously generated skeletons of $(n - 1)$ carbons as "seeds" to produce all skeletons of n carbons. To each skeleton of $(n - 1)$ carbons is added one carbon which may be attached to one or more (up to four) carbons of the original skeleton. Each new skeleton so generated is then maximized, and the canonical result is then checked for redundancy with all previously created skeletons of n atoms. The SKEL_GROW program uses the first method described above for the checking of redundancy and so is not very efficient; it was written only to provide a check for the first program (SKEL_GEN). These programs are described in more detail under Experimental Section.

Table I. Skeleton Generation Statistics

n[a]	max r[a]	no. of skels produced	no. of unique skeletons	CPU time[12] (hh:mm:ss)
3		2	2	00:00:01
4		6	6	00:00:02
5		28	21	00:00:02
6		170	78	00:00:04
7		1 380	353	00:00:25
8		13 455	1 929	00:04:08
9		151 459	12 207	00:51:24
10		1 951 711	89 402	12:07:11
11		28 660 233	739 335	192:44:50
12	4	5 392 555	278 229	50:48:36
13	3	6 007 038	288 151	69:25:21
14	3	27 712 814	968 260	361:53:45
15	2	15 360 051	534 493	235:32:38
16	1	4 733 890	170 988	82:32:00

[a] The number of carbons is equal to n, the number of rings to r.

RESULTS

The rapid growth of the number of skeletons is apparent in Table I, which also records the times required for their generation. The increase in numbers on going from n to $n + 1$ atoms is roughly a factor of $(n - 1)$ to $(n - 2)$ up to the maximum size fully generated, i.e., 11 atoms. Above 11 atoms the numbers and times became prohibitive and we restricted the generation to a maximum number of rings. This was done to keep with our focus on the totals as representing chemically viable carbon skeletons, since many higher polycyclics would be impossibly strained when considered as real carbon skeleton molecules. The success of the SKEL_GEN program in reducing the redundancy may be seen in the two columns which show how many skeletons were generated and the final number of unique skeletons actually found.

There are several ways to break down and classify these skeletons into smaller groups. The total number of unique skeletons found is broken down by numbers of rings in Table II, which shows qualitatively the expected distribution, peaking at or just above $n/2$ rings. The familiar enumeration of the acyclic skeletons is shown in the first column and acts as a check on the accuracy of the method, being in agreement with the alkane enumerations previously derived.[2,4,5,6a,6b,13] The totals for the cases of $n \leq 6$ agree with those in Harary's text,[14] and the monocyclic totals for $n \leq 8$ agree with those in section 64 of Pólya's paper.[3] We have since used Pólya's method to enumerate the monocyclics and bicyclics up to 50 atoms,[15] and these numbers agree with those of Table II. Finally, the two different programs (SKEL_GROW and SKEL_GEN) used here both produced the same totals up to 11 atoms; SKEL_GROW proved too inefficient for general enumeration above that size but was used in some special contexts described in the next section.

For comparison with the tabulation of all possible skeletons in Table II is the collection of 555 skeletons, shown in Table III, which were extracted from the 6000 compounds of the starting material catalogue used in the SYNGEN program. This catalogue is limited to 16 skeletal atoms and is clearly a minute subset of the total possible. The catalogue contains the most common of the approximately 24 000 compounds in the 1989 Aldrich catalogue. Even if all the 18 000 compounds not in the SYNGEN catalogue were skeletally nonisomorphic, the whole

Table II. Distributions of Carbon Skeletons

	r												
n	0	1	2	3	4	5	6	7	8	9	10	11	12
3	1	1											
4	2	2	1	1									
5	3	5	5	4	2	1	1						
6	5	12	17	18	14	8	3	1					
7	9	29	56	79	79	59	31	9	2				
8	18	73	182	326	430	427	298	134	35	6			
9	35	185	573	1 278	2 161	2 768	2 616	1 714	707	154	16		
10	75	475	1 792	4 875	10 162	16 461	20 346	18 436	11 477	4 399	845	59	
11	159	1 231	5 533	17 978	45 282	90 111	140 605	167 703	146 428	87 191	31 409	5 440	265
12	355	3 232	16 977	64 720	192 945	*a*							
13	802	8 506	51 652	227 842	*a*								
14	1 858	22 565	156 291	787 546	*a*								
15	4 347	60 077	470 069	*a*									
16	10 359	160 629	*a*										

[a] Maximum number of rings was restricted.

Table III. Distributions of Carbon Skeletons from SYNGEN Catalogue

	r				
n	0	1	2	3	4
3	1	1			
4	2	2			
5	3	3			
6	5	6	1		
7	9	7	3		1
8	12	16	3	1	
9	16	23	6		
10	14	34	16	3	
11	13	32	19	2	
12	10	30	27	4	
13	9	28	30	3	
14	4	16	22	9	
15	2	14	26	12	
16	2	7	26	19	1

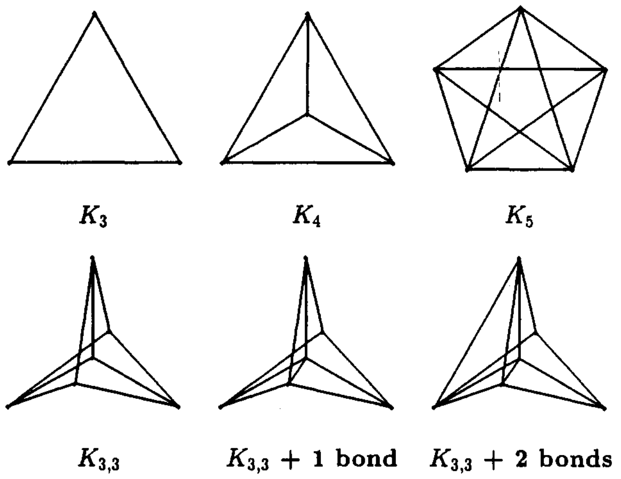

Figure 1. Generated K_n graphs and $K_{3,3}$ "seed" graphs.

catalogue of this supplier could be represented by fewer than 19 000 skeletons. The variety of catalogue skeletons drops abruptly above tricyclic. The number of possible skeletons of fewer than 4 rings and up to 16 carbons is estimated to be about 15 million, so the whole Aldrich catalogue uses less than about 0.13% of the possible skeletons with $n \leq 16$ and $r \leq 4$.

However, a significant proportion of the generated skeletons must consist of physically nonviable molecular skeletons, especially those with many rings. The maximum numbers of rings are found in the K_n graphs, of which K_3 (cyclopropane) and substituted K_4 (tetrahedranes) are known but K_5 is impossible on five carbons; see Figure 1. The higher K_n skeletons violate the carbon valence limit and so the rows for $n \geq 6$ in Table II are truncated to fewer rings than required[16] for K_n; even so, it is apparent that most of the higher polycyclics will be grossly unstable as molecules. A method for elimination of some of these from the table is explored below.

Table IV. Distribution of 8-Carbon Acyclics

	N_4		
N_3	0	1	2
0	1	3	1
1	4	3	
2	5		
3	1		

Table V. Distribution of 8-Carbon Monocyclics

	N_4		
N_3	0	1	2
0	1	6	4
1	5	17	1
2	18	7	
3	12		
4	2		

Table VI. Numbers of Monocyclic Skeletons

ring size	no. of skeletons	ring size	no. of skeletons
3	83 571	10	1 612
4	69 429	11	534
5	43 291	12	169
6	29 391	13	39
7	16 307	14	11
8	8 771	15	2
9	3 894	16	1

A further breakdown of the large sums in Table II may be sought in the numbers of tertiary and quaternary atoms in the skeleton (in graph terms, vertices of degrees 3 and 4, respectively). The numbers of primary and secondary atoms may be derived from these. If n is the number of atoms, r the number of rings, and N_v the numbers of atoms with valence (or degree) of v, then:

$$N_1 = 2N_4 + N_3 - 2(r - 1)$$
$$N_2 = n + 2(r - 1) - 3N_4 - 2N_3$$

Such a breakdown of the totals is shown in Tables IV–V for the 18 acyclic and 73 monocyclic 8-atom skeletons and for the 10 359 acyclic 16-atom skeletons. A sampling of one skeleton from each matrix entry is shown for the 8-atom acyclics and monocyclics in Figure 2, as drawn by the DRAW_SKEL program.

Of the 257 022 monocyclic skeletons in Table II, the frequency of ring sizes is collected in Table VI. The most common are three-membered rings since they offer the most

Table VII. Classification of Bicyclics by α

	α						
n	0	1	2	3	4	5	6
5		1	3	1			
6	1	2	11	3			
7	4	9	32	11			
8	20	26	100	35	1		
9	76	87	294	113	3		
10	288	257	881	351	15		
11	1 005	787	2 590	1 093	57	1	
12	3 433	2 322	7 639	3 348	231	4	
13	11 324	6 891	22 344	10 218	853	22	
14	36 712	20 160	65 278	30 906	3 131	103	1
15	116 809	58 939	189 832	93 005	11 007	473	4

Table VII. 10-Carbon Separated Bicyclics

	ρ_2				
ρ_1	3	4	5	6	7
3	120	102	29	7	1
4		20	7	1	
5			1		

Table IX. 10-Carbon Spiro Bicyclics

	ρ_2					
ρ_1	3	4	5	6	7	8
3	55	88	39	18	4	1
4		25	18	5	1	
5			2	1		

Table X. 10-Carbon Fused (α = 2) Bicyclics

	ρ_2						
ρ_1	3	4	5	6	7	8	9
3	201	233	155	66	25	5	1
4		78	66	23	5	1	
5			15	5	1		
6				1			

opportunity for articulation in the $n - 3$ acyclic carbons which remain. The numbers for the several largest rings are easily confirmed by hand. The monocyclics may also be further classified by the frequency of tertiary and quaternary atoms.

The bicyclic ring systems may be classified according to the two ring sizes ($\rho_1 \leq \rho_2$) and the number of atoms (α) the two rings have in common;[17] the simple designation of a bicyclic system is therefore $\rho_1\rho_2\alpha$. For $\alpha = 0$ the two rings are separated (no atoms in common); the spiro bicyclics have α = 1. Fused rings are either vicinally fused ($\alpha = 2$) or bridged ($\alpha \geq 3$). Fused rings also display a third ring, the envelope ring, $\rho_3 = \rho_1 + \rho_2 - 2(\alpha - 1)$, such that $\rho_3 \geq \rho_2 \geq \rho_1$. There are three possible drawings of any fused bicyclic, with each of the three rings as the outer envelope. The definition given for ranking the relative ring sizes fixes the designation $\rho_1\rho_2\alpha$ as unique. For n atoms the largest bicyclics will have $n = \rho_1 + \rho_2 - \alpha$.

We have generated the bicyclic systems up to 15 atoms, for which the largest bridging will be $\alpha = 6$, i.e., a 4-atom bridge across an 11-membered ring. The enumeration of the bicyclics is characterized in Table VII, and a typical further breakdown by ring sizes is collected for the set of 10-atom skeletons in Tables VIII–XII Even with these nested classifications the numbers remain large, and further breakdown into smaller sets may be obtained by classifying the atom valencies as in Tables IV and V. A sampling of these 10-carbon skeletons may be seen in Figure 3.

The 10-atom results suggest an exploration of the possible monoterpene C_{10} and sesquiterpene C_{15} skeletons, a problem that was first examined by Smith and Carhart[18] using the CONGEN program. They had three methods for generating the

Table XI. 10-Carbon (α = 3) Bridged Bicyclics

	ρ_2					
ρ_1	4	5	6	7	8	9
4	90	104	55	20	5	1
5		40	25	5	1	
6			4	1		

Table XII. 10-Carbon (α = 4) Bridged Bicyclics

	ρ_2		
ρ_1	6	7	8
6	9	4	1
7		1	

Table XIII. Numbers of $K_{3,3}$ Supergraphs

	r								
n	4	5	6	7	8	9	10	11	12
6	1	1	1						
7	1	4	5	4	1				
8	3	11	19	18	8	2			
9	6	31	66	83	54	15	2		
10	15	85	230	369	353	189	49	4	
11	33	235	772	1571	2059	1718	853	211	16

Table XIV. Numbers of Norbornane Supergraphs

	r									
n	2	3	4	5	6	7	8	9	10	11
7	1									
8	3	8	12	11						
9	13	51	142	260	304	223	75			
10	40	253	944	2441	4343	5296	4130	1849	355	14

Table XV. Numbers of Bicyclooctane Supergraphs

	r									
n	2	3	4	5	6	7	8	9	10	11
8	1									
9	2	6	8	12						
10	9	36	114	247	366	337	142			
11	26	185	800	2596	5990	10001	11175	7682	2487	229

possible terpene skeletons. One of these methods was to generate all the 10-carbon acyclic, monocyclic, and bicyclic skeletons and extract those which could be decomposed into isoprene units. We wrote a program to go through our database of skeletons and perform the same extraction. Our numbers for 10-carbon skeletons agree with theirs, as do our numbers for monoterpene skeletons. We were able to use the same method to find the numbers of possible sesquiterpene skeletons with up to two rings: those numbers are 136 for acyclics, 3952 for monocyclics, and 59 969 for bicyclics.

Superstructure searches are possible with the SKEL_GROW program, taking as a "seed" any input skeleton and allowing the program to successively add to it more carbons, one at a time, all possible ways. As an illustration the 6-atom $K_{3,3}$ graph was examined; since this is a sterically nonviable skeleton, all larger graphs with $K_{3,3}$ as a subgraph must also represent chemically impossible structures. All atoms of the parent skeleton have a valence of 3; because of the symmetry there is only one way to add another bond (and ring) to the 6 atoms, and also only one way to add one more bond without violating the valence limit. These three "seeds", shown in Figure 1, were used to generate Table XIII, which tabulates the numbers of superstructures containing $K_{3,3}$.

Since SKEL_GROW only adds new atoms but does not add bonds across existing atoms, any "seed" will produce all superstructures in which no new bonds appear among the seed atoms. This was done for illustration with norbornane ($\rho_1\rho_2\alpha$

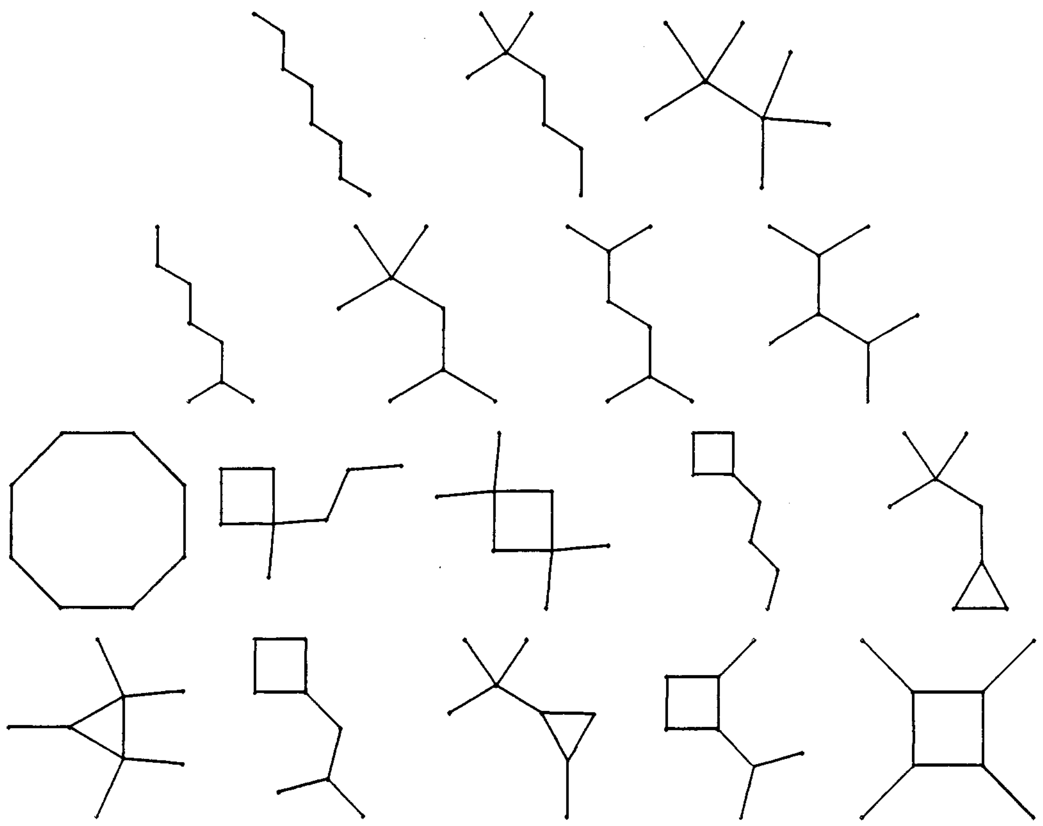

Figure 2. Examples of 8-carbon acyclic and monocyclic skeletons.

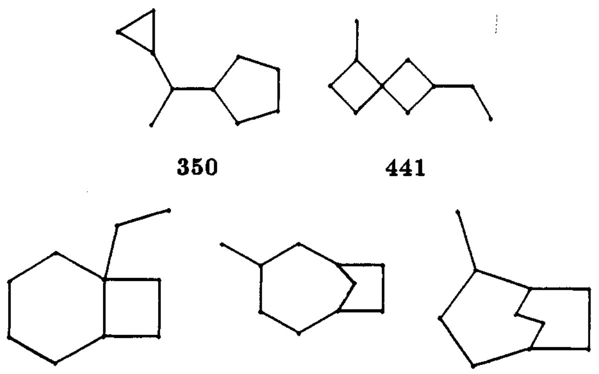

Figure 3. Examples of 10-carbon bicyclic skeletons. (Labels give $\rho_1\rho_2\alpha$ designations.)

= **553**) and bicyclooctane (**664**), and the results are given in Tables XIV and XV, respectively. Because of the large disk access time, however, generating supergraphs in this way is a slow process.

EXPERIMENTAL SECTION

The Program SKEL_GEN. The basic premise of the program SKEL_GEN is that every carbon skeleton has associated with it a unique maximized adjacency matrix. Hence, if a program were to generate all possible adjacency matrices it would necessarily generate all maximized matrices and thus all nonisomorphic skeletons.

The adjacency matrices were generated by generating all combinations of all configurations of zeros and ones for the portion of each row which is to the right of the diagonal (hereafter referred to as *forward bits*). This is done in a bottom-up manner. Thus, if for a skeleton of n atoms the program had just generated a new sequence of forward bits for row k, it would clear all forward bits for rows $k + 1$ to $n - 1$ and check the matrix for validity. This involves a final valence check (row sums must not exceed four) and a check to determine whether the matrix is a maximized matrix. If the matrix is a maximized one, it would be stored in a packed format.

The program would then set the row counter C_r to $n - 1$, call the subroutine NEXT_SEQUENCE, which would generate the next sequence of ones and zeros for row C_r, and check the matrix for validity. When it had done this for all sequences for row C_r it would clear the forward bits for rows $C_r + 1$ to $n - 1$, get the next sequence for row C_r, set C_r to $n - 1$, check for validity, and repeat the process.

Generating all possible adjacency matrices would take a great deal of time, especially when dealing with higher numbers of vertices. However, since the only matrices required are those which are maximized, we can use some of the properties of such matrices to reduce the total number which must be generated.

One property of carbon skeletons is that they are connected. The program checks for this in the following way. Whenever a new sequence for a row is generated, the program checks each row in the matrix to see whether it has bits set to the left of the diagonal (hereafter referred to as *back connections*). In the maximized matrix for a connected graph every row with the exception of the first will have at least one back connection, so if a row is found without any back connections the matrix is not a valid one. In this case, if row k were found to have no back connections, C_r would be set to $k - 1$ and the next sequence for C_r would be found.

The row sum represents the valence of the atom.[9] Hence no row may have more than four bits set. Also, since the first row is the most significant part of the maximal binary string, no row may have more bits set than the first row. Furthermore, there are only two skeletons with a maximum valence of only two, the straight chain (alkane) and the n-atom ring (cycloalkane). The program therefore simply generates these two first and then starts directly on the variations with row one having three bits set, i.e., valence of three.

Cooperative indexing affords restrictions for the reduction of the numbers of matrices generated. Cooperative indexing[9,10] requires that an atom assigned for the first time is given the lowest unused number available; this ensures that the bits in the adjacency matrix appear as far to the left as possible. In every maximal binary string the bits set in row one must be in the first (leftmost) positions. Since the program has only to generate matrices with three or four bits in row one, it has only two possible bit sequences for that row: bits two through four set for a tertiary atom 1 and bits two through five set for a quaternary atom 1. Cooperative indexing also restricts the length of possible bit strings for each row. For example, if row r of a matrix had b back connections and atoms 1 through p had already been assigned in previous rows, no valid bit sequence for row r can set a bit beyond $p + (4 - b)$. In addition, bit $p + 2$ could be set only if bit $p + 1$ were already set, and bit $p + 3$ could be set only if both bits $p + 1$ and $p + 2$ were already set.

There are two properties of the rows for atoms attached to atom 1 which were used to reduce further the number of invalid matrices generated. For the first we found that there are only eight valid bit sequences for row two when atom 1 is tertiary and there are only nine when atom 1 is quaternary. Rather than generate all sequences for row two, the program simply took these valid sequences from a data statement. The second property was that, of the rows for atoms attached to atom 1, the current row could have no more connections to other atoms of the set than did previous rows. If the current row had as many connections to those atoms as did a previous row, then the binary number obtained by considering connections to atoms beyond those connected to atom 1 could be no greater for the current row than that for the particular previous row.

The use of all these properties of the maximized matrices of carbon skeletons greatly reduced the total numbers of matrices generated and therefore allowed the program to be run for higher numbers of atoms. Take for instance the generation of 10-carbon skeletons; the upper-right triangle of the matrix consists of 45 bits. If the program were simply to generate all the combinations of zeros and ones for these bits it would have to generate 2^{45}, or about 3.5×10^{13} matrices. By using the properties mentioned in all but the previous paragraph, the number of generations was reduced to about 13 million. That number was further reduced to about 8.5 million by using only the valid sequences for row two, the first property from the previous paragraph. Finally, by using the second property the number was brought down to less than 2 million.

Even with all the aforementioned ways of reducing the number of matrices generated, the time required became prohibitive when higher numbers of atoms were considered. Since a great number of the graphs with high numbers of rings probably cannot exist as viable molecular skeletons, the program was modified to generate all the skeletons with fewer than a given number of rings. This cut down on the number of matrices generated and therefore allowed the program to run to higher numbers of atoms (Table II).

The Program SKEL_GROW. There were two reasons for the development of a second program for the construction of carbon skeletons. The first was that there was a need for a way to check the results of SKEL_GEN. Concerns of time and error make it impractical to manually generate all the skeletons for more than six carbons. An alternative to that was to develop a program which used different methodology to solve the same problem and use the results from the two programs to correct both programs. The second reason for the development of SKEL_GROW was that the methodology used might reduce the number of skeletons generated far enough that the time saved in skeleton generation would more than offset the time required by isomorphism checks.

The general strategy of SKEL_GROW is relatively simple. The program takes a skeleton of n atoms, adds one more atom $(n + 1)$ to it in all possible nonequivalent ways, and maximizes the resulting skeletons. It then takes those first-level skeletons one at a time and for each one it joins atom $n + 1$ further to a second existing atom in all nonequivalent ways to produce second-level skeletons. The process is repeated on the nonisomorphic second-level skeletons to produce third-level skeletons (with valence of three on atom $n + 1$), then once again to produce fourth-level skeletons. Finally, the program examines all the first- to fourth-level skeletons so produced to find out whether they had been produced before.

The Program DRAW_SKEL. This program accepts a maximal matrix and places point 1 in center screen. The matrix is then searched to find the smallest ring incorporating atom 1 and places such a ring onscreen from point 1, with equal angles. If atom 1 is in two rings, a second ring is added onscreen with calculated angles. Any remaining atoms attached to atom 1 are placed outside the ring(s) at equal angles. If no ring incorporates atom 1, the attached atoms are arrayed equidistant from point 1 (in a triangle if tertiary, or a square if quaternary), and arbitrarily numbered. New atoms continue to be added to the growing skeleton in the same way until it is complete.

Skeletons with an atom in more than two rings revert to an alternative drawing protocol which draws all numbered atoms first in a regular polygon periphery and then creates the bonds between them (as in K_5 in Figure 1). Although the procedure is not meant to create chemically acceptable drawings for difficult, bridged polycyclics, it is very successful for ordinary skeletons, such as those in Figure 2. The program created chemically acceptable drawings for 98% of the skeletons (544/555) in the SYNGEN catalogue (Table II).

ACKNOWLEDGMENT

The receipt of a grant from the National Science Foundation in support of this work is gratefully acknowledged.

REFERENCES AND NOTES

(1) Cayley, A. On the Mathematical Theory of Isomers. *Philos. Mag.* **1874**, *47*, 444–446.
(2) Henze, H. R.; Blair, C. M. The Number of Isomeric Hydrocarbons of the Methane Series. *J. Am. Chem. Soc.* **1931**, *53*, 3077–3085.
(3) Pólya, G.; Read, R. C. *Combinatorial Enumeration of Graphs, Groups, and Chemical Compounds*; Springer-Verlag: New York, 1987; pp 58–74.
(4) Read, R. C. The Enumeration of Acyclic Compounds. In *Chemical Applications of Graph Theory*; Balaban, A. T., Ed.; Academic Press: London, 1976; pp 25–61.
(5) Knop, J. V.; Müller, W. R.; Jeričević, Ž.; Trinajstić, N. Computer Enumeration and Generation of Trees and Rooted Trees. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 91–99.
(6) (a) Masinter, L. M.; Sridharan, N. S.; Lederberg, J.; Smith, D. H. Applications of Artificial Intelligence for Chemical Inference. XII. Exhaustive Generation of Cyclic and Acyclic Isomers. *J. Am. Chem. Soc.* **1974**, *96*, 7702–7714. (b) Lederberg, J.; Sutherland, G. L.; Buchanan, B. G.; Feigenbaum, E. A.; Robertson, A. V.; Duffield, A. M.; Djerassi, C. Applications of Artificial Intelligence for Chemical Inference. I. The Number of Possible Organic Compounds. Acyclic Structures containing C, H, O, and N. *J. Am. Chem. Soc.* **1969**, *91*, 2973–2976.
(7) Read, R. C.; Cameron, R. D.; Colbourn, C. J.; Wormald, N. C. Cataloguing the Graphs on 10 Vertices. *J. Graph Theory* **1985**, *9*, 551–562.
(8) Hendrickson, J. B.; Bernstein, Z.; Miller, T. M.; Parks, C. A.; Toczko, A. G. New Directions in the SYNGEN Program for Synthesis Design. In *Expert System Applications in Chemistry*; Hohne, B. A., Pierce, T. H., Eds.; ACS Symposium Series 408; American Chemical Society: Washington, DC, 1989; Chapter 6.
(9) Hendrickson, J. B.; Toczko, A. G. Unique Numbering and Cataloguing of Molecular Structures. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 171–177.
(10) Kvasnička, V.; Pospíchal, J. Canonical Indexing and Constructive Enumeration of Molecular Graphs. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 99–105.

(11) A similar unique numbering also results from seeking the minimum binary string from the matrix (Randic, M. On Canonical Numbering of Atoms in a Molecule and Graph Isomorphism. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 171-180). Although this approach is of course equally unique as a canonical representation, it is substantially longer and requires more storage space, as discussed in ref 9.
(12) Times given are for the DEC Micro VAX 3500.
(13) Davies, R. E.; Freyd, P. J. $C_{167}H_{336}$ Is The Smallest Alkane With More Realizable Isomers than the Observed Universe Has 'Particles'. *J. Chem. Ed.* **1989**, *66*, 278-281.
(14) Harary, F. *Graph Theory*; Addison-Wesley: Reading, MA, 1969; pp 213-224.
(15) Parks, C. A., Brandeis University, unpublished results.
(16) The degree of each vertex in any K_n graph is $(n - 1)$, and the number of rings is equal to $(n^2 - 3n)/2 + 1$.
(17) Hendrickson, J. B. Fragmentations and Rearrangements in Organic Synthesis. *J. Am. Chem. Soc.* **1986**, *108*, 6748-6756.
(18) Smith, D. H.; Carhart, R. E. Structural Isomerism of Mono- and Sesquiterpenoid Skeletons. *Tetrahedron* **1976**, *32*, 2513-2519.

ELDAR, a Knowledge Base System on Microcomputer for Electrolyte Solutions. The Factual Knowledge of ELDAR

JOSEF BARTHEL* and HERIBERT POPP

Institute for Physical and Theoretical Chemistry of the University of Regensburg, 8400 Regensburg, Federal Republic of Germany

Received July 11, 1990

The knowledge base system ELDAR (**EL**ectrolyte **DA**ta **R**egensburg), consisting of data base, method base, rule base, and communication manager, classifies the knowledge of electrolyte solutions into factual, algorithmic, and rule knowledge. In this paper information is given on the factual knowledge of ELDAR and the mapping of facts in Codd's relational data model with an extension of its "1st Normal Form" to repeating attributes. ELDAR offers equal user interfaces for all factual knowledge services, such as literature, data, thesaurus, module, parameter, basic data, and rule retrieval.

INTRODUCTION

At all times the properties of aqueous electrolyte solutions have been of crucial interest in biology, medicine, geology, oceanography, limnology, and various chemical and electrochemical technologies. During the last decade the high flexibility of nonaqueous electrolyte solutions for tackling technical problems has increasingly focused the attention of applied research on suitable solvents and solvent mixtures for high-energy batteries, wet capacitors, electroplating, phase-transfer catalysis, electroorganic synthesis, solar cells, thin-film procedures, coating, etc.[1,2] An almost unlimited scale of solution properties offered by mixed solvent systems permits the realization of electrolyte solutions with properties planned on the drawing board. A most important obstacle to the search of suitable electrolyte solutions for special applications is the lack of data books or data bases on electrolyte solutions. The electrolyte data are widely spread in the literature, and many problems have already been solved in fields quite different from that of an actual research. However, a time-consuming literature research often does not yield the required information. The needed data must often be interpolated or estimated from similar systems, well-known procedures to every engineer.

ELDAR was developed in 1980 to assist scientists and engineers concerned with electrolyte solutions. A data, method, and rule base and their interaction, controlled by a communication manager make up this knowledge base system, see Figure 1.

ARCHITECTURE OF ELDAR

Past experience has shown that the knowledge produced and used by scientists and engineers cannot be represented by only one of the four elementary types of knowledge representation, i.e., logic programming, production rules, semantic networks, and frames.[4] In accordance with the quite recently developed object-orientated languages,[5] ELDAR manages the knowledge on electrolyte solutions in three categories.[6]

- Factual knowledge is represented in a relational data model and managed by a data base with the basic relations quoted in Figure 1 and explained by their attributes in Table I.
- Algorithmic knowledge is represented in normalized modules and managed by a method base as statistical, mathematical and physicochemical modules.
- Rule knowledge is represented in Horn clauses and managed by a rule base.

Table I. Definition of Relations Representing the Factual Knowledge[a]

LITERATURE	(*primary key*, classification, identification, CA number, language, document type, year, volume, issue, page, authors, title, publication, descriptors, remarks)
CHEMICAL SYSTEM	(*primary key*, *number*, system TAG, components, reaction products, reaction, variable characterization)
DATA	(*primary key*, *number*, *data number*, variable values)
THESAURUS	(*name key*, name, preference name, synonyms, superterms, subterms, associative terms, formula)
MODULE	(*name*, title, category, descriptors, source, date/version, externals, connections, programming language, accuracy, memory, files, standard, theory, remarks)
EFFECT	(*name*, *number*, entry name, program type, function domain, function unit, effect terms, effect, manual, instruction example, test data)
PARAMETER	(*name*, *number*, *parameter name*, data type, domain, usage type, terms, unit)
BASIC DATA	(*primary key*, system TAG, components, data source, degree of freedom, sign changes, error type, mean weight, estimated variance, used module, integer constants, real constants, state variables, state range, variable range, basic data)
RULE	(*primary key*, title, category, descriptors, source, date/version, externals, connections, language, probability, explanation, remarks)

[a] The italic attributes build the primary key.