a table giving the connectivity index, the experimental lipophilicity, and the calculated lipophilicity. However, the table was not accompanied by a derivation of how the calculation was made. A careful study of the footnotes reveals that for each functional group a separate function is necessary, because the functional group influences the intercept of the straight line representing the equation. Thus, instead of a straightforward calculation for a new functional group, a multiple regression analysis of derivatives with this functional group has to be made to find the intercept. The fact that this is not clearly stated in the article is reason for us to doubt the usefulness of the connectivity index.

Many other kinds of properties might be calculated. One of these would be the distances of the atoms in a molecule. It would be possible to derive parameters needed for quantum mechanical calculations. We are working at the moment on an extension to combine the fragments using pattern recognition techniques.

LITERATURE CITED

(1) M. Osinga and A. A. Verrijn Stuart, "Documentation of Chemical Reactions. II. Analysis of the Wiswesser Line Notation", *J. Chem. Doc.*, **14**, 194–98 (1974).
(2) C. M. Bowman, F. A. Landee, N. W. Lee, and M. H. Reslock, "A Chemically Oriented Information Storage and Retrieval System. II. Computer Generation of the Wiswesser Line Notation of Complex Polycyclic Structures", *J. Chem. Doc.*, **8**, 133–8 (1968).
(3) T. Ebe and A. Zamora, "Wiswesser Line Notation Processing at Chemical Abstracts Service", *J. Chem. Inf. Comput. Sci.*, **16**, 33–5 (1976).
(4) A. Zamora and T. Ebe, "Pathfinder II. A Computer Program That Generates Wiswesser Line Notation for Complex Polycyclic Structures", *J. Chem. Inf. Comput. Sci.*, **16**, 36–39 (1976).
(5) E. G. Smith, "The Wiswesser Line-Formula Chemical Notation", McGraw-Hill, New York, N.Y., 1968.
(6) R. F. Zürcher, "171. Protonenresonanzspektroskopie und Steroidstruktur. I. Das C-19-methylsignal in Funktion der Substituenten", *Helv. Chim. Acta*, **44**, 1380–95 (1961).
(7) R. F. Zürcher, "232. Protonenresonanzspektroskopie and Steroidstruktur. II. Die lage der C-18 and C-19-methylsignale in Abhängigkeit von den Substituenten am Steroidgerust", *Helv. Chim. Acta*, **46**, 2054–88 (1963).
(8) J. E. Page, "Nuclear Magnetic Resonance Spectra of Steroids", *Annu. Rep. NMR Spectrosc.*, **3**, 149–210 (1970).
(9) N. S. Bhacca and D. H. Williams, "Application of NMR Spectroscopy in Organic Chemistry", Holden-Day San Francisco, Calif., 1964.
(10) J. N. Shoolery and M. T. Rogers, "Nuclear Magnetic Resonance Spectra of Steroids", *J. Am. Chem. Soc.*, **80**, 5121–351 (1958).
(11) T. Fujita, J. Twasa, and C. Hansch, "A New Constant Derived from Partition Coefficients", *J. Am. Chem. Soc.*, **86**, 5175–80 (1964).
(12) G. G. Nys and R. F. Rekker, "Statistical Analysis of a Series of Partition Coefficients with Special Reference to the Predictability of Folding of Drug Molecules. The Introduction of Hydrophobic Fragmental Constants (*f* Values)", *Chem. Therap.*, **5**, 521–35 (1973).
(13) A. I. Vogel, W. Greswell, G. Jeffery, and I. Leicester, "Bond Refractions and Bond Parachors", *Chem. Ind. (London)* 358 (1950).
(14) R. F. Rekker and D. J. Reiding, "Over de praktische bruikbaarheid van Vogel's moleculaire bindings-refracties", *Chem. Weekblad*, **58**, 513–19 (1962).
(15) O. Exner, "Additive Physcial Properties. II. Molar Volume as an Additive Property", *Collect. Czech. Chem. Commun.*, **32**, 1–23 (1967).
(16) S. Sugden, "A Relation between Surface Tension, Density and Chemical Composition", *J. Chem. Soc.*, 1177–89 (1924).
(17) O. Exner, "Additive Physical Properties. III. Re-examination of the Parachor", *Collect. Czech. Chem. Commun.*, **32**, 1–23 (1967).
(18) L. B. Kier, L. H. Hall, W. J. Murray, and M. Randić, "Molecular Connectivity. I. Relationship to Nonspecific Local Anesthesia", *J. Pharm. Sci.*, **64**, 1971–74 (1975).
(19) W. J. Murray, L. H. Hall, and L. B. Kier, "Molecular Connectivity. III. Relationship to Partition Coefficients", *J. Pharm. Sci.*, **64**, 1978–81 (1975).

Hash Functions for Rapid Storage and Retrieval of Chemical Structures

W. T. WIPKE,* S. KRISHNAN, and G. I. OUCHI

Board of Studies in Chemistry, University of California, Santa Cruz, California 95064

Received August 8, 1977

A method is described for determining if a given chemical structure or its enantiomer is contained within a file in time essentially independent of file size. The stereochemically extended Morgan algorithm (SEMA) name is used as a key for directly computing the address of the compound. Three separate files of compounds are used to study the effectiveness of four different hash functions. Various subsets of the SEMA name were also used as keys to study effect of information loss on hashing efficiency. A work function is used to compare the amount of work required to access a compound in the file.

INTRODUCTION

"Is this compound commercially available?" and "Is this a new compound?" are frequent questions in chemistry. To answer such questions the chemist generates a name for the compound and "searches" for an identical name in an ordered list of compound names. The names might be IUPAC or WLN[1] and they are normally ordered alphabetically. Even with this ordering the time required to determine if a compound is contained within a file of compounds increases as the size of the file increases. The time required for such a search becomes very important when the number of searches being performed is large.

The same questions arise within our Simulation and Evaluation of Chemical Synthesis (SECS) program.[2,3] When a synthetic precursor is generated, SECS must determine if the compound already exists in the synthesis tree or if it exists in a library of available starting materials or if it is a common precursor existing in other synthesis trees.[4] Since this determination must be made for every precursor and the files being searched may be large, we were interested in an efficient search method.

Hash table methods are well known for searching files with a search time that is independent of the number of records in the file.[5] When the number of records is large, a good hash table method is faster than a linear search method for which the search time is proportional to n or a binary search method whose search time is proportional to $\log_2 n$, where n is the number of records in the file.

The reason for the greater speed of hash table methods is that the hash function, h, operates directly on the "search key", K, to produce an address, $h(K)$, or position in the hash table where the requested entry must be if it exists in the file. Of course, the same hash function must be used for retrieval as was used for creating the file.

Occasionally two different keys will hash to the same address, $h(K) = h(K')$. This situation is called a *collision*. The colliding keys are attached to that entry in the table as a linear list. A linear search of that short list is required when

0095-2338/78/1618-0032$01.00/0

Table I. Components of the SEMA Name

Component name	No. of bytes[a]
HEADER	
NN = length of name in words[b]	1
AN = number of nonhydrogen atoms	1
BN = number of bonds	1
NDB = number of double bonds	1
CHN = number of tetrahedral stereocenters	1
FROM LIST	AN-1
RING CLOSURE LIST	2*(BN-AN+1)
NODE LIST (ATYPE, CHARGE)	AN
BTYPE	BN[c,d]
DOUBLE BOND STEREOCHEM. PARITY	NDB[c]
TETRAHEDRAL STEREOCENTER PARITY	AN[c,e]

[a] 7-bit bytes. [b] For space economy name is densely packed; hence NN depends on word length of computer. [c] 3-bit bytes. [d] BTYPE is forced to begin on a full word boundary for ease of handling. [e] Not included if CHN = 0.

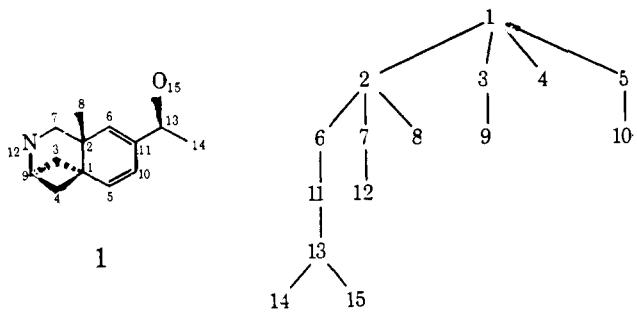

Figure 1. The SEMA spanning tree (right) derived from structure **1**. The FROM LIST cites the parent of each node; e.g., 13 is FROM 11. RING CLOSURES for **1** are 4–9, 9–12, and 10–11.

a key maps to that location. A suitable choice of h will minimize the probability of collisions, but normally cannot eliminate collisions. Ideally, for a hash table of size n, the probability that two randomly chosen keys will map to the same location is $1/n$.

Hash techniques were used by Feldmann[6] for searching molecular formula and CAS registry files although details of the hash function were not presented. Owing to the importance of structure searching in chemical information systems and, in particular, computer-assisted organic synthesis systems, we studied the efficiency of four hash functions using typical chemical structure files, one of which contained stereoisomers, as a function of hash table size and level of information present in the keys.

THE SEMA NAME

The first problem in structure searching is to generate a canonical name for the structure from its structural diagram or equivalent connection table (CT).[2] Using a canonical name eliminates the need for cross-indexing or multiple entries of a structure under different names. The stereochemically extended Morgan algorithm (SEMA), which we described previously, produces a stereochemically unique name from a CT.[7] How the SEMA name is generated is described in detail in an earlier paper.[7] For the purposes of this work it is only necessary to know the form and information content of the different parts of the SEMA name. The name is a variable length string, beginning with a HEADER describing the length of the string, the number of nonhydrogen atoms, number of bonds in the hydrogen-suppressed structure, number of double bonds, and the number of tetrahedral stereocenters (see Table I). Following the HEADER, the FROM LIST describes a spanning tree which includes all atoms in the hydrogen-suppressed structure, and in the case of acyclic structures includes all bonds, but for cyclic structures, one bond per ring is excluded. The excluded bonds appear as RING CLOSURE

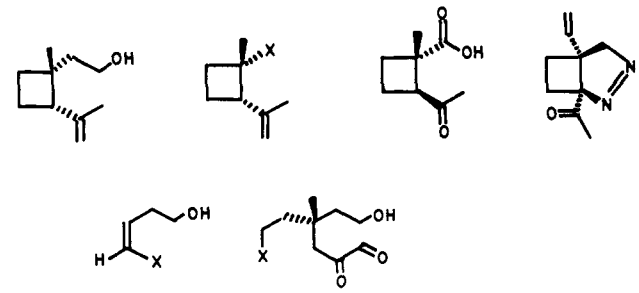

Figure 2. FILE A contained 645 structures derived from a synthetic analysis of grandisol. Some of the structures are illustrated here. The file contained stereoisomers.

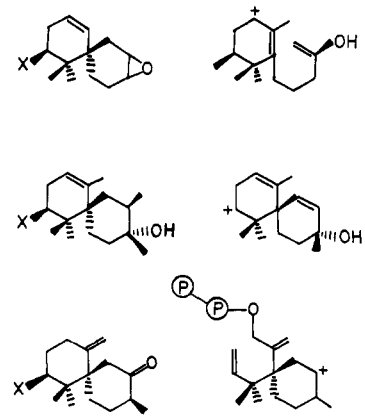

Figure 3. File T contained 953 structures derived from biosynthetic analysis of glandulifera epoxide. These are typical of the structures in the file. The file was pruned of stereoisomers.

bonds (see Figure 1). Next is the NODE LIST giving atom types and charges in the order atoms appear in the FROM LIST, and the BTYPE LIST giving bond types in the order they appear in the FROM LIST, and the RING CLOSURE LIST. Finally come the stereochemical parities for double bonds and tetrahedral stereocenters if there are any. The SEMA name components for structure 1 (Figure 1) are shown below. To form the name, the labels on the left are removed and the components are concatenated in the order given to

HEADER	11,15,17,2,4
FROM	1,1,1,1,2,2,2,3,5,6,7,11,13,13
RING CLOSURE	4-9, 9-12, 10-11
NODE	1,1,1,1,1,1,1,1,1,1,1,3,1,1,4
BTYPE	1,1,1,1,1,1,1,1,2,2,1,1,1,1,1,1,1
DOUBLE BOND PARITY	2,2
TET. STEREOCENTER PARITY	1,5,0,0,0,0,0,0,2,0,0,0,4,0,0

form one long string. Enantiomers have identical SEMA names except for the stereocenter description where they have codes 4 and 5 interchanged.[7]

EXPERIMENTAL SECTION

Chemical Structure Files. Three files of structures were used in this work, files A, B, and T. **File A** contained 645 structures including stereoisomers from a synthetic analysis of grandisol[8] generated by SECS 2.4. Some of the structures from file A are illustrated in Figure 2. **File B**, which contained 491 structures, was prepared from file A by eliminating all but one of each set of stereoisomers. Thus the compounds in file B had stereochemistry, but the file contained no stereoisomers. File B was a subset of file A. **File T** contained 953 structures resulting from a biosynthetic analysis of glandulifera epoxide,[9] but stereoisomers were removed as they were for file B. See Figure 3 for typical structures found in file T.

Hash Tables. Hash tables of 256, 512, and 1024 entries were used in the various experiments. The tables had the

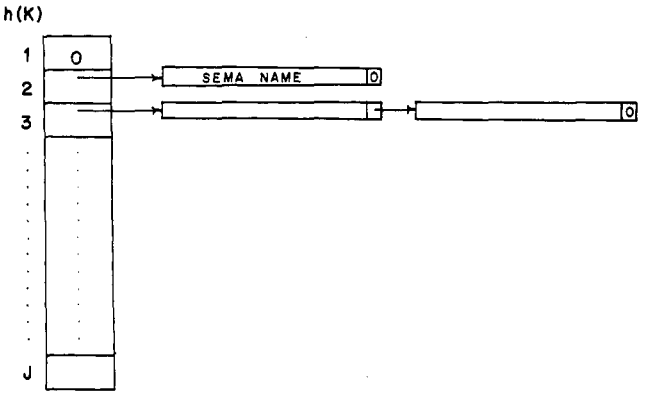

Figure 4. Hash tables contained J entries, where J = 256, 512, or 1024 Each empty entry contained 0; each nonempty entry contained a pointer to the list of SEMA names (variable length) that shared that *home address*.

general structure shown in Figure 4, where h(K) = 1 illustrates an empty entry; h(K) = 2 shows a normal entry, for which only one structure in the file has h(K) = 2 as its *home address*. A collision of two structures is evidenced for h(K) = 3. The table entries point to lists of SEMA names.

Hash Functions. Each of the following functions mapped the variable length SEMA names into 8, 9, or 10 bits depending upon whether the hash table had 256, 512, or 1024 entries, respectively. The four functions are shown schematically in Figure 5 and are detailed below. Implementation is described for a computer having 36 bit words.

Function h_1. Successive words of the SEMA name were exclusive OR'ed (XOR) together forming a 36-bit result. This word was divided into three fields of 12 bits each, and these three fields were combined again by XOR giving a final 12-bit result. The right-hand 10 bits were used as h(K) for a 1024 table. For tables of 512 and 256, the word was divided into 4 fields of 9 bits each, and these were XOR'ed together. For the 256 table, the rightmost 8 bits were used.

Function h_2. Each work in the SEMA name was divided into fields of 6, 10, 10, 10 bits which were XOR'ed together forming a 10-bit result. This was done for each word and the 10-bit results were added together. This was repeated on the result giving a 10-bit value for h(K). For table sizes 512 and 256, the word was divided into four fields of 9 bits and five fields of 4, 8, 8, 8, and 8 bits, respectively.

Function h_3. Each word was taken modulo (J - 1) where J is the table size. The remainders were added together and the sum taken modulo (J - 1). Since J is an even power of 2, J - 1 was used to prevent the *mod* from being a simple bit shift. Thus only J - 1 entries were used. Mod(J) was 3, 6, and 10 times worse than mod(J - 1) for file A (256) at levels STEREO, CONST, and HEADER, respectively.

Function h_4. h_4 is the same as h_2 except that, instead of addition, the XOR operation is used.

File Structure and Updating. Frequently hash-coded files are created in such a way that updating and making additions to the file are difficult if not impossible.[5] Our hash-coded files are easily updated because they are list structured (Figure 4). The files are created by starting with a null file and inserting each compound into the file using direct access methods. An estimate of the ultimate size of the file is needed when the file is originally initialized in order to select the proper hash table size. In the interest of simplicity and generality, we allow collision lists to span several physical disk blocks if necessary.

RESULTS

Evaluation of the Functions. The ideal hash function would distribute the structures evenly into the table to minimize

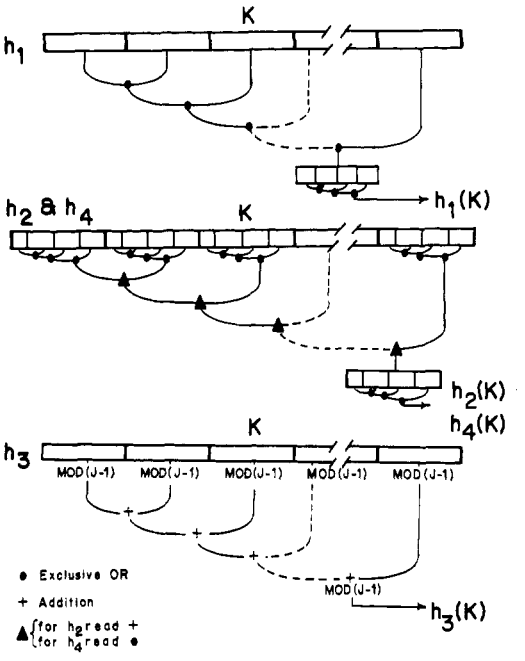

Figure 5. The four hash functions h_1–h_4 map variable length SEMA name keys (K) into a home address h(K) in the hash table (see Figure 4).

collisions. Since we have no a priori knowledge of the contents of the file, the best we can hope for is to distribute the structures randomly into the table. Figure 6a shows graphically how h_1 distributes the 645 structures of file A into a 512 entry table. For comparison, a distribution of 645 random numbers between 1 and 512 is shown in Figure 6b (Fortran random number function). Distributions from h_2 to h_4 are similar to h_1, and all are without any apparent skew. The same is true when files B or T are used.

The four functions appear to generate nearly identical distributions as shown by Figure 7, which plots the frequency (ν) of occurrence of empty table entries, entries having one structure, two structures, etc. We actually plotted the log (ν + 1) to reduce curvature and emphasize low frequency occurrences of entries having many structures. The distributions are essentially superimposable on the random distribution and are not very different from one another.

Work Required for Retrieval. A very important criterion for evaluating the hash functions is "how much work is required to determine if a compound is in the file?" For the moment let us ignore the effort in calculating h(K). Given h(K), the work in finding the structure in the file is simply the work required to search the list of structures having h(K) as their home address. This work is proportional to $1/2(n_i + 1)$, where n_i is the length of the structure list for the ith entry in the hash table. Thus the total work (W) to retrieve every compound using a hash table with J entries is

$$W \approx \sum_{i=1}^{J} n_i(n_i + 1) \qquad (1)$$

and the average work ($\bar{W}$) to find a single structure is then $\bar{W} = W/N$, where there are N structures in the file. Table II tabulates the average work per retrieval for the four hash functions under various conditions. Two other functions are included for comparison, the *random* function and the theoretically *perfect* function. The perfect function assigns one structure to each hash table entry until the table is filled, then a second structure is assigned to each table entry until it is filled, then a third, and so on, so the perfect function produces the absolute minimum number of collisions, with all collision lists being of minimum length, thus leading to the mininum

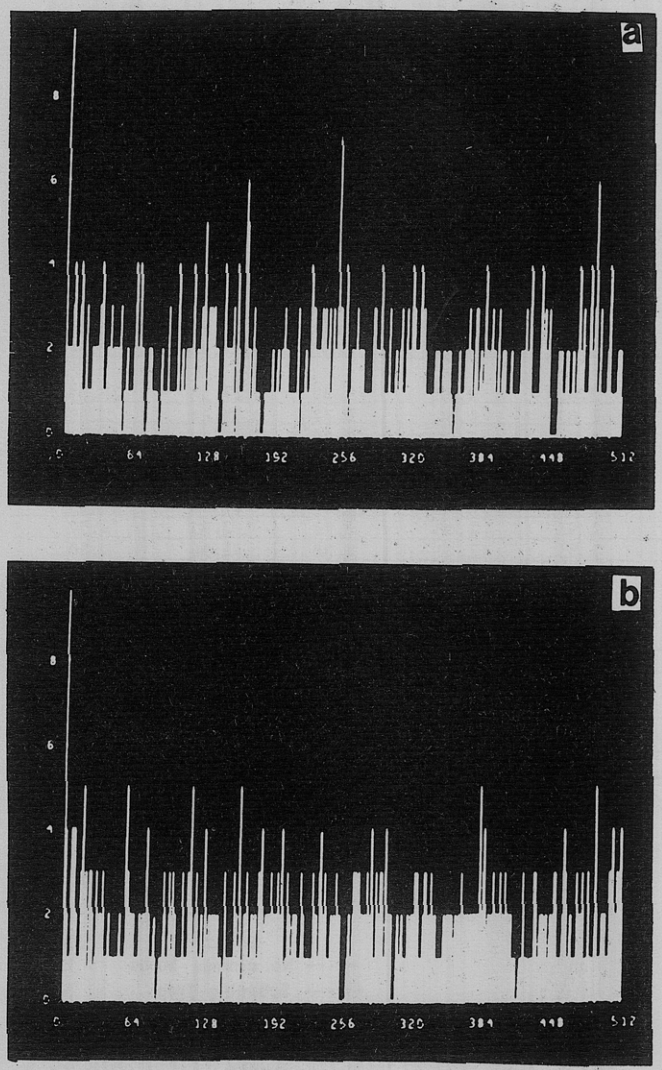

Figure 6. The number of structures per table entry (ordinate) is plotted vs. the hash table index (abscissa) for the distribution generated by (a) hash function h_1 applied to file A containing 645 structures using the STEREO level of key and (b) a random number function generating 645 random members between 1 and 512. In both cases the table size was 512.

$\bar{W}$ possible. The average work for the perfect function is given by

$$\bar{W}_{\text{perfect}} = \frac{q+1}{2N}[q(J-4) + r(q+2)] \qquad (2)$$

where q = INT(N/J) and $r = N \bmod J$, J is the table size, N is the number of structures in the file, and q and r are integers. The derivation of equation 2 follows from equation 1. If the table is large enough to contain the file without collision $\bar{W} = 1.00$, e.g., file A in 1024 table. The random function on that combination gives $\bar{W} = 1.31 \pm 0.01$, which is 76% theoretical efficiency ($1.00 \times 100/1.31$).[10] The STEREO row uses the entire SEMA name as the key. Comparing that row for each hash function with the random function, we conclude again that the four functions are not very different and are comparable with the random function.

The $\bar{W}$ for the random and four h functions using the STEREO key follows the simple relation

$$\bar{W} = (L/2) + 1 \qquad (3)$$

where L is the load factor (N/J) for the hash table. The average work per retrieval rises as the load factor rises, but is always $1/J$th that of retrieval from one large linear list.

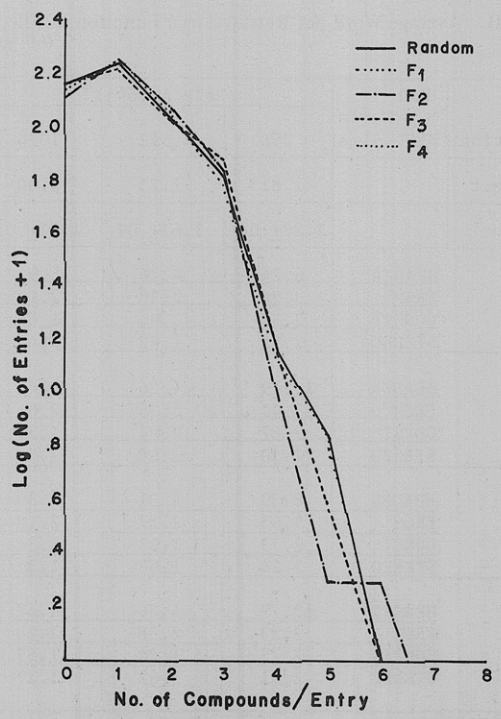

Figure 7. The distributions from h_1–h_4 are found to be very close to the distribution obtained from a random function. These data are for file A with a table size of 512 using the STEREO key.

Calculation of the hash function must be done once per compound when the file is originally created, and later once for each retrieval. Normally the time required to access the hash file on disk and resolve collisions is about 100 times greater than that required to calculate h(K). Rotational latency of a fixed head disk is about 17 ms in contrast to the 0.2 ms required to calculate a complex h(K). However, if the hash file is in memory and no input/output is required, then the work involved in calculating h(K) becomes important. Table III shows the number of machine instructions of various types required to calculate h(K) as a function of U, the number of words in K being used, and F, the ratio of the number of bits in a word to the number of bits in the final h(K). The time on a Digital Equipment PDP-10 for computing a STEREO hash of a typical compound where $U = 9$, $F = 4$ is given in Table III for functions h_1–h_4. Clearly h_1 is the easiest function to calculate and h_2,h_4 the most difficult, but on a disk-based retrieval, calculation of the function represents only 1% of the retrieval time.

Size of Hash Table. Optimally the hash table should be larger than the number of structures in the file, but Table II shows that changing the load factor from 0.5 to 2.0 less than doubles the average work per retrieval. In general, hashing breaks the file into J subproblems where J is the size of the hash table. Choice of a table size depends on which resource is more dear. Table II and eq 3 provide a basis for logically making that choice.

Level of Information in Key. The general principle seems to hold that *the more information in the key, the less work in retrieval.* Inclusion of stereochemistry in the key significantly improved the efficiency of retrieval for file A which contained stereoisomers and did not degrade efficiency for files B and T which did not contain stereoisomers. When the key contained only the HEADER and FROM LIST (FROM entry in Table II), the performance dropped but was still surprisingly good, considering that ring closures and atom and bond types are not included. When the key contained only the HEADER, there was significant degradation in efficiency, particularly for file T. The only information in the key in this case is the length of the name, and the number of atoms, bonds,

Table II. Average Work per Retrieval as a Function of File, Key, Size of Hash Table, and Hash Function

Average Work per Retrieval

Function	Hash Table Size / KEY	File A (645)[a] 256	File A (645)[a] 512	File A (645)[a] 1024	File B (491) 256	File B (491) 512	File B (491) 1024	File T (953) 256	File T (953) 512	File T (953) 1024
Perfect		1.81	1.21	1.00	1.48	1.00	1.00	2.39	1.46	1.00
Random[b]		2.27±.04	1.62±.02	1.31±.01	1.96±.03	1.48±.03	1.23±.02	2.85±.03	1.94±.03	1.46±.02
h_1	HEADER[c]	6.24[g]	4.26	5.92	4.82	3.40	4.49	29.08	27.72	28.28
	FROM[d]	3.24	2.38	2.14	2.56	1.91	1.79	4.03	2.72	2.47
	CONST[e]	2.50	1.94	1.65	1.95	1.48	1.30	3.01	2.01	1.64
	STEREO[f]	2.32	1.62	1.40	2.01	1.47	1.30	2.87	1.98	1.54
h_2	HEADER	10.79	4.26	4.42	8.20	3.40	3.53	29.39	27.54	28.09
	FROM	3.42	2.35	2.01	2.65	1.87	1.62	3.77	2.63	2.33
	CONST	2.67	2.01	1.59	2.08	1.54	1.25	2.88	1.99	1.48
	STEREO	2.30	1.58	1.30	1.90	1.47	1.21	2.84	1.94	1.52
h_3	HEADER	8.39	4.31	4.89	6.41	3.45	3.91	25.19	27.70	28.35
	FROM	3.63	2.37	2.10	2.81	1.91	1.68	3.71	2.83	2.57
	CONST	2.73	1.93	1.54	2.06	1.51	1.21	2.88	1.96	1.50
	STEREO	2.24	1.63	1.31	1.99	1.46	1.25	2.85	2.00	1.46
h_4	HEADER	10.79	4.26	4.42	8.20	3.40	3.53	29.39	27.54	28.09
	FROM	3.53	2.38	2.06	2.75	1.91	1.67	3.95	2.72	2.32
	CONST	2.77	1.94	1.59	2.14	1.48	1.25	2.97	2.01	1.51
	STEREO	2.31	1.62	1.29	2.03	1.47	1.25	2.89	1.98	1.55

[a] Number in parentheses is number of structures in the file. File A contains stereoisomers. [b] Work and standard deviations are derived from averaging 20 random distributions. [c] Only the HEADER part is used as the key (see Table I). [d] HEADER and FROM list used as key. [e] SEMA name up to but not including double bond or tetrahedral stereochemistry. [f] The full SEMA name including stereochemistry used as key. [g] Means average retrieval requires comparisons of 6.24 keys.

Table III. Computing Effort to Evaluate the Hash Functions

Function	Number of instructions by type: Exclusive OR	Logical shift	Integer add	Integer divide	Time, μs[b]
h_1	$U+F-2$[a]	F			45
h_2	$(U+1)(F-1)$	$F(U+1)$	$U-1$		234
h_3			$U-1$	$U+1$	184
h_4	$(U+1)(F-1)+(U-1)$	$F(U+1)$			234

[a] U is the number of words in the SEMA name being used to calculate h(K), F is the ratio of the number of bits in a computer word to the number of bits in the final h(K). [b] Approximate time for $U=9$, $F=4$, on a PDP-10 (where divide requires 16 μs and XOR, LSH, ADD require 3 μs each).

and stereocenters. These levels of information are compared graphically in Figure 8.

Hash coded searching is excellent for rapidly finding an *exact match* with the key. Thus using the complete SEMA name leads one directly to the exact stereoisomer desired if it is in the file. If one wishes to find if a given structure *or its enantiomer* is in the file, there are two possible approaches using hash coding: (1) construct the file using the entire SEMA name as key, then probe for the structure using its SEMA name, and if unsuccessful, change the SEMA name (by interchanging 4's and 5's in the stereodescription)[7] to that of the enantiomer and probe the file again. One cannot efficiently find all stereoisomers this way. (2) Alternatively, construct the file using only the constitutional part of the SEMA name (without stereochemistry) as a key, but store the entire SEMA name in the file. Probe the file using the constitutional level key for the desired structure. All stereoisomers of that constitution will share that home address and will be in the same linear list. One can then browse through, picking out the enantiomer, epimer, etc. Note that in browsing, the entire name must be examined, because unrelated structures may also share this home address. Thus, lowering the level of information in the key decreases efficiency

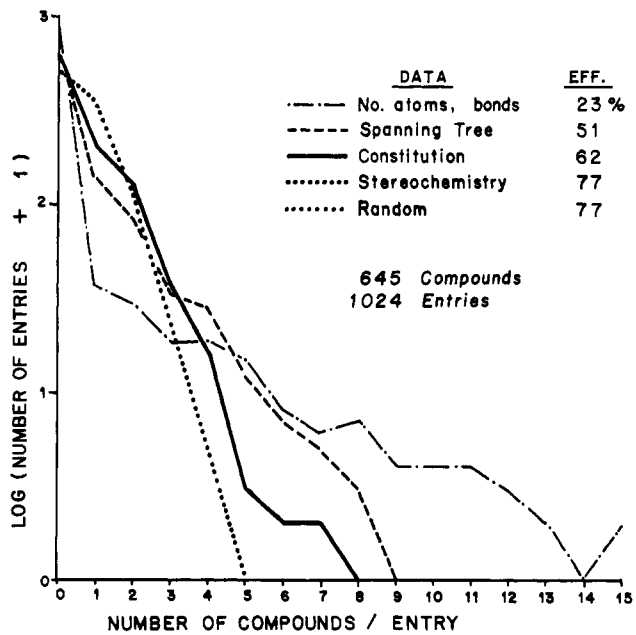

Figure 8. As the amount of structural detail in the key increases, the theoretical efficiency of the hash function approaches that of the random function. The narrower the distribution, the more efficient is the function for retrieval. In this example (h_4 applied to file A with table of 1024 entries) when the key is the full SEMA name (STEREO), no entry contains more than four structures.

of *exact* matches, but increases the capability for and efficiency of *partial* matches.

CONCLUSION

Chemical structures with stereochemistry can be efficiently located in a file by hash-coded search techniques. We found that hash-coded searching was most efficient when the entire SEMA name was used as the key, including the stereochemical descriptions. The four hash functions h_1–h_4 were as efficient

as the random function in distributing structures evenly in the file. This shows that there are many possible ways of hashing the key which give equally good results, as long as all bits in the key are involved. We described the theoretically perfect hash function as a standard against which other hash functions could be compared. We also defined a practical definition of work required for retrieval which facilitates comparison of results on the basis of expected performance. These techniques have been implemented in the SECS program for computer-assisted design of organic synthesis.

ACKNOWLEDGMENT

This work was supported by a National Cancer Institute Contract N01-CP-75816 and by an allocation of the SUMEX-AIM resource at Stanford (RR00785, J. Lederberg, principal investigator).

LITERATURE CITED

(1) For a review of chemical notation systems see M. F. Lynch, J. M. Harrison, W. G. Town, and J. E. Ash, "Computer Handling of Chemical Structure Information", American Elsevier, New York, N.Y., 1971.
(2) W. T. Wipke, "Computer-Assisted Three-Dimensional Synthetic Analysis", in "Computer Representation and Manipulation of Chemical Information", W. T. Wipke, S. R. Heller, R. J. Feldmann, and E. Hyde Ed., Wiley, New York, N.Y., 1974, pp 147–174.
(3) W. T. Wipke, H. Braun, G. Smith, F. Choplin, and W. Sieber, "SECS—Simulation and Evaluation of Chemical Synthesis: Strategy and Planning", in "Computer-Assisted Organic Synthesis", W. T. Wipke and W. J. Howe, Ed., *ACS Symposium Series*, 1977, pp 98–129; W. T. Wipke, G. I. Ouchi, and S. Krishnan, "Simulation and Evaluation of Organic Synthesis—SECS. An Application of Artificial Intelligence Techniques", *Artificial Intelligence*, in press.
(4) W. T. Wipke, G. I. Ouchi, and S. Krishnan, in preparation.
(5) (a) W. D. Maurer and T. G. Lewis, "Hash Table Methods", *Comput. Surveys*, **7**, 5–19 (1975); (b) D. E. Knuth, "Sorting and Searching", *Art Comput. Programming*, **3**, 506–549 (1973); (c) D. E. Knuth, "Algorithms", *Sci. Am.*, **236**, 63–80 (April 1977).
(6) R. J. Feldmann, "Interactive Graphic Chemical Structure Searching", in "Computer Representation and Manipulation of Chemical Information", W. T. Wipke, S. R. Heller, R. J. Feldmann, and E. Hyde, Ed., Wiley, New York, N.Y., 1974, pp 55–81.
(7) W. T. Wipke and T. M. Dyott, *J. Am. Chem. Soc.*, **96**, 4825, 4834 (1974).
(8) J. A. Katzenellenbogen, "Insect Pheromone Syntheses: New Methodology", *Science*, **194**, 139–148 (1976).
(9) M. Suzuki, E. Kurosawa, and T. Irie, "Three New Sesquiterpenoids Containing Bromine, Minor Constituents of *Laurencia glandulifera* Kützing, *Tetrahedron Lett.*, 821–24 (1974).
(10) Note that the theoretical efficiency of the random function varies as a function of L, the table loading factor, and is complicated because of the discrete nature of the perfect function (for example, the first derivative of $W_{perfect}$ as a function of L is discontinuous). Therefore, the $\bar{W}$ is better for comparing real hash functions at different load factors.

A New Linear Representation of Chemical Structures

LUCIANA QUADRELLI*
Montedison—Istituto Ricerche "G. Donegani", Novara, Italy

VITTORIO BAREGGI
Montedison, Milano, Italy

SERGIO SPIGA
Montedison—Centro Ricerche Antiparassitari, Milano, Italy

Received November 8, 1976

A system of recording, analyzing, and displaying chemical structures by a new linear notation system is described.

I. INTRODUCTION

The problem of filing, reading, and analyzing structural formulas by using a computer in a simple and fast way is considered important in any fine chemical research center where relationships between structures of many new chemical substances and their physical-chemical or/and biological properties are studied. This is considered one of the most important problems in the Pesticide Research Center of Montedison's Agriculture Division. This is due both to the need to control the heavy quantity of data that are being collected and to correlate statistically the properties of a substance and its chemical structure. Therefore we designed a new method of chemical structure representation which we call the "linearization method" because the structural formula (two dimensional) takes the form of a linear sequence of characters, as in other methods (Dyson, Wiswesser). This linearization method, is more simple and economic than the older ones.

In the following we shall present the constraints and the criteria used in the linearization method (section II), the technique to represent a chemical structure on a special format (section III), the rules selected in order to write the special format (section IV), a special set of rules to rapidly characterize ring structures (section V), and finally the techniques adopted to determine substructure in a set of structures.

II. LINEARIZATION METHOD

A. Our constraints for the method were: the linearized formula has to maintain all the information shown by the structural formula; the linearized formula has to hold letters normally used in the normal input/output units of computers.

B. The following principles were established. Every formula may be written on a grid with small and insignificant deformations as compared to the classical representation. Each atom or atomic group of the structural formula must be on a junction. Bonds must be disposed on the sides; the formula so arranged can be written line after line in a linear succession of character, with respect, for every line, to the following conventions: (a) all the elements (atom or atomic group, empty junctions, and horizontal bonds) must be listed sequentially, from left to right, using the appropriate coding, row after row; (b) the possible vertical bonds starting from any atom must be expressed in a way that permits one to single out the above starting points; (c) a special character permits one to express in the linearization the end of a row condition of the grid without the necessity to start a new coding line.

0095-2338/78/1618-0037$01.00/0