(48) Dubois, J. E.; Sicouri, G.; Sobel, Y.; Picchiottino, R. "DARC System. Localized Operators and Co-structures of a Reaction Invariant". C. R. Hebd. Seances Acad. Sci., Ser. B 1984, 298, 525-530.
(49) Sicouri, G.; Sobel, Y.; Picchiottino, R.; Dubois, J. E. "DARC System. Localizing Variations onto the Reaction Invariant: the Transformation Structure Concept". C. R. Hebd. Seances Acad. Sci., in press.
(50) Picchiottino, R.; Sicouri, G.; Dubois, J. E. "DARC-SYNOPSYS Expert System. Production Rules in Organic Chemistry and Application to Synthesis Design". In "Computer Science and Data Bank"; Hippe, Z.; Dubois, J. E., Eds.; Polish Academy of Sciences: Warsaw; in press.

Question of Data Format in Organic Chemistry

GUIDO SELLO
Laboratorio di Chimica Organica, Universita' degli Studi, Milano, Italy

Received April 13, 1983

Organic chemistry of necessity has developed mechanized information retrieval. The problem of characterizing the format of the compounds' structures and reactivity is examined. Current and future trends are given with comments and critical observations. Information about both informatic and chemical aspects of data collection is treated from both user and producer viewpoints. Techniques and tricks in structuring and maintaining data are given.

INTRODUCTION

The question of database design became pressing when the opportunity to substitute automatic for manual management occurred. Since this option has been available, many remarkable results have been obtained in nearly every operating field. Database design gave useful and accurate solutions for chemical information. A number of works and publications appeared, sometimes starting from quite different positions, that focused specific questions of chemical data management. A number of specific positions have been reported in the field of formating data relevant to chemical arguments.

In this article, we will examine those positions pertaining to Organic Chemistry. There are two kinds of fundamental information available to the user of organic chemical data: molecular structures and reaction records. A problem became apparent to the people who designed databases for organic chemists. A great amount of work, both theoretical and experimental, appeared; all of which pertained to the precision and reliability of the various methods for listing and codifying structures and reactions. An important amount of success was achieved in the field of codification of structures but not in that of codification of reactions. This happened for two strongly correlated reasons: the lack of a real and efficient methodology for analyzing the theoretical aspects of reactions and the necessity of translating them from usual language to code. We will discuss problems and solutions connected to each matter separately.

FAMILY OF MOLECULAR STRUCTURES

This class was treated in several works. From the first algorithms,[1,2] less refined and less general to the latest and most polished solutions, they came to the best level possible in the field of codification; the strictly informatic aspect is of remarkable quality. In fact, we must remember that, dealing with the question of inserting records in a big database, one has to solve both problems of real logic (connected to codification system) and also the usual problems of storage and working feasibility. These last ones, severely limited by hardware available, must be optimized.

The problem of transforming the representation of molecules (usually bidimensional drawings with some tricks for pointing out the third dimension) into computer-readable form (series of binary numbers) arises. We must translate the instinctive information supplied by the drawing into exact machine representation. The separation of essential parts of a visual picture may appear more or less simple, according to the situation; for example, the description of a range of different colors is evident (e.g. one may call the first COLOR 1, the second COLOR 2, etc.), while agreement on the tonality or shape is less clear. In the case of the description of molecules, it is well-known that a drawn representation is built by points, identifying atoms, and by lines, identifying bonds. Actually, what we learn by looking at the graphic representation of a structure is much more: in fact, we may guess that some points-atoms are tetravalent carbons saturated with hydrogens in the valences not displayed, that some bonds are placed above the molecular plane and others below it, that an oxygen atom marked by the symbol O has no hydrogens connected to it if it has already two bonds, and so on. In some way this must be taught to a machine, which (because of its stupidity) knows nothing about organic matter. In addition, it is necessary to prevent the computer from mistaking two similar structures or codifying the same molecule in different ways. These problems must be solved in the best possible way in order to build a system that is efficient, both for management (ease in updating, deleting, inserting) and for retrieval (strictly connected to the data structures), handy, having good interaction with the outside, and reliable, giving nonambiguous answers.

Methods were sought to codify structures that would give canonical formulations in output and allow the user to rely on the correctness of the results. Among the several algorithms, we must consider Morgan's algorithm, which, appropriately modified and extended, is still the most used.[3-5] They are systems to translate the pictorial molecular view into numerical vectors or matrices, whose various components, $a(i(j))$, are the signs of the presence/absence of certain molecular features.[6] Obviously, the trick for getting canonical results resides in solving the points of choice in an absolute manner.

One of the best solutions in the field of structure representation is based on graph theory and, more exactly, on a treelike graph, which is well suited for describing even very complex chemical formulas. When the problem concerning rings or stereoisomers representation (i.e. tridimensionality) has been solved, it is simple to find an algorithm, apply it to a labeled graph, and obtain an univocal solution (e.g., using a prefixed traversal order; preorder or postorder to get only one output).

More complete and exhaustive works in the field are devoted to solving the different questions unsolved by graph theory, mainly concerning the mentioned problems of rings and stereoisomers. In both cases, one has to adapt a general theory, like graph theory, to the reality of organic molecules. For example, in the case of rings it is useless to state the number and kind of all existing rings, while it is appropriate to consider

0095-2338/84/1624-0249$01.50/0

those useful in the desired application (e.g., rings with six or less constituents that could escape a partial analysis, cis–trans relationship between anular substituents, etc.). The existing solutions are variations to algorithms of graph theory or of canonical Morgan's enumeration of structures.[3,7-12] Among these, each has a quality, either calculation, retrieval, or storage efficiency, with obvious dedicated underlinings.

The latest transformation in connected tables (bidimensional matrix), in crossed lists (vectors or matrices series), and in other different numerical situations reflects the peculiarities of the algorithms used, the programming language, and even the hardware. Thus, we can obtain an enormous series of numbers (overall 0 and 1) divided among several sections (stereochemistry, connection, etc.), more compact representations (i.e., reduced to the minimal possible size), and representation reduced below the threshold of complete description (foreseeing a particular computational system to get the complete informations back).[13,14] At this point, it is interesting to insert these observations into the specific field where the obtained data are used. In fact, we can point to two large areas in which the use of records connected with molecular structures is foreseeable: the area tied to data management for bibliographic use and the area devoted to data handling for synthetic design use. Problems and objectives for these two blocks are different from both the logical and the operative viewpoints. The differences justify a wholly unrelated handling of the two problems.

First, we examine the "bibliographic data" question. Here a different tactic is likely to be appropriate depending on the database dimension one has to build and manage. If the amount of data to be examined is limited, we have mainly to consider retrieval and simplicity/completeness of information. On the other hand, for an enormous amount of records we are faced with a storage problem while it is equally important to assure speed, feasibility, and simplicity in updating. In the first case, the approach is to facilitate data management, while the second one demands a participation by the experts in all the aspects connected to information theory. Thus, one may think that the first case is wholly solved by examining the second one, and we prefer to limit our analysis to it. The final product to be inserted into the computer mass storage is not obtained by simply solving the problem of the exact formulation of an algorithm to translate the object into machine language. If we suppose we applied any of the present processes to codify, we get one or more matrices containing the distinctive molecular numbers back. At this point we have to transform them into real structures the computer can manage. If the codifying system distinguishes among different features and it is exhaustive enough, we will usually obtain a sparse matrix. Maintaining every element in memory would waste a great deal of space. We may decide to use some conversions between matrices and the different systems connected to them. An example is represented by the feasibility of describing a sparse matrix with a five-row matrix (directly derived from three- and four-row matrices); this is also simple to update, while its inferior analogs could appear difficult to modify.[15] Another problem in building big database structures is the difficulty of retrieving, updating, deleting, and inserting records in the existing collection. Thus, we must exactly consider both the selected representation for describing the molecule and the kind and size of collection. In file structures the most usual trend still remains that of using the ordering procedure called "inverted indexes", which is very efficient in retrieval practice though very ponderous in the real indexes management.[16-20] Naturally, we may think of several other solutions such as follows: maintaining direct indexes but ordered with some sorting process, easier to maintain but more complex to use and logically less efficient; maintaining the collection disordered, using for the retrieving, updating, etc. processes as hash functions, key sets, etc.[15] In big databanks a last problem is the necessity of inserting into the code some checking characters that allow verification that the datum has not changed in transmission. We will not discuss this, but we must point to it. Everything mentioned is partly applicable to small data collections, useful and used in a limited center (e.g., institute, laboratory, university course). Naturally, in these cases, some aspects take on more relevance than others. In more limited spheres, the talk of refinement in formating data is less pressing, while the problems of differences among users and of the connected system in interfacing with the outside may become crucial. In the structure field this kind of reasoning is less relevant than in the reaction field, given the ascertained uniformity obtained in codifying and the complete understanding of the underlying problems.

The problem of adapting the structural code for use in synthesis is different, because we must really solve problems of interrelation between autonomous parts and not problems of storage or management in data retrieval. In fact, the reports[3,7,10-13,21,22] that have explicitly faced the solution of the aforesaid approach form a logical beginning, even if they come from different viewpoints and neglect the great amount of refinements of management kind such as ordering, matrix compressing, etc. On the other hand, the problems connected to codes, canonical representation, and efficiency of transposition from view of description, as the envisaged solutions, are evidently of the same level and type. The common feature in all these works is that of transferring, in a preformated scheme, the features of the structures examined. This is realized by building big matrices in which rows (or columns) may represent the general feature, while the corresponding crossing represents the detail. The unique problem is building the connection between the data code section and all the rest of the reasoning devoted to work on synthesis. This requires provision for the insertion in the molecule description of any information (useless in bibliographic data field) that is either essential or simplifying at the time of the real elaboration. As examples of this position, Corey's and Blair's works show very interesting results because they give, in their code experiments, many pieces of information completely redundant structurally but extremely meaningful in working on reactivity. In one report,[22] explicit subdivision of data in atom and bond matrices is used; in another one[13] is used a measure, in BE matrices, of quantity of electrons unshared at the level of simple availability. This allows a stated objective to be realized much more easily and eliminates burdensome problems.

However, some facts remain stated in the structures field. They are that the translating system between usual representation and automatic language can now be considered close to the desired reality, that the various possibilities in database building must consider both the fundamental situation tied to the representation and the contingent situation tied to the using sphere, and that there exists the possibility of change in the interface field to enhance the dialog user/machine.

REACTION FAMILY

Unlike the previous discussion, talk of cataloging the (organic) reactions for a documentary purpose is typical for its logic field, strategic objectives, practical applications, and information bases. It is worth making some historical remarks on this matter. Everyone looking at what has been realized in this field would be greatly disappointed to realize that only a limited number of articles about systems for automatic information (general or particular) exist after at least 30 years of discussion. We can mention six effective attempts carried out in the past (and some of these are not very different): Hendrickson's total code;[21,23] Blair's reactional matrices;[24]

Corey and Wipke's databases systems; Vall's plurindexed system;[25] Ziegler's work[26,27] (from Reactiones Organicae to RIRS); various choices of the most famous series (*Organic Synthesis*, *Theileimer*, *Cahiers de Syntese Organique*, etc.). The first question that appears at this point concerns the reasons that produced this situation. Two arguments hinder the transformation of reactions into code: the possible ambiguity and the poor understanding that still characterizes this field. This makes it difficult to standardize the code. Thus, in practice it is difficult to build computer-devoted algorithms that, owing to the static nature of the machine, are an unavoidable condition for getting an efficient solution. We must also point out that these objections have already been made previously by the authors who studied the problem.

The existing experiences tended toward two principal directions with two different approaches. They are the arithmetic attempt and the one tied to compounds and mechanisms, strictly coming from laboratories.

The total translation from linguistic to numerical expression is seldom attractive for a chemist for two basic reasons: the practice of expressing his concepts by words and the exact comprehension and regimentation of the problem required by numerical representation. Those few attempts, in part successful, of applying numbers to codifying reactions remain very interesting. Blair's reactional (**R**) matrices attempt to substitute a numerical expression for the actual fact of chemical transformations. These are mainly based on the real effects that the application of a determined reaction causes on the substrate. Because the actual changes made by reactions are limited in scope, the matrix representation described by Blair is unexpectedly simple yet comprehensive. This is an incredible result when the apparent substilities of organic reactions are considered. A justification is provided by a careful examination of Blair's choice. In his congruent and autoconsistent matrices, there is no space for all the stereochemical and functional refinements that would, for example, allow one to distinguish the alkylation of a ketone from that of an α,β-unsatured one. Instead, the attention is centered upon broken and formed bonds and produced and neutralized charges, so obtaining a uniformity in the description of the entire field. Then, the starting compounds, to which the transformation is applied, will contain everything that is lacking in the reaction description. This is unacceptable in the field of data management.

Henrickson's work (enormous and articulated) is halfway between numeric solution and the one tied to literal description.[21,23] This author, using a particular structure code, attempts to obtain a homogeneous system in which structures and reactions may be represented by symbols of the same kind, allowing a straight interrelation between the two fields. The worry (fundamental for any work on data) of obtaining an efficient system, univocal and tied to the machine, is not present in the description oriented to synthesis used by Hendrickson to codify structures; rather, he tries to extract from the reality of a molecule information useful for the synthetic chemist. Thus, the optimization reached by other systems is not present, even if his system is useful for synthesis design. If we want to examine critically the realized or possible situations, it is essential to make some observations on Hendrickson's work that do not diminish the logical and factual importance of his determinations. In fact, in this article we discuss the fitting and the efficiency of the code in obtaining a useful system for data management. In this field, the attempt, made by Hendrickson, to free the description of reactions from the prejudice inherent in usual formal structures is very interesting. He obtains a nonpredetermined schema that allows the building of translation algorithms logically related and universally applicable; furthermore, the number of feasible solutions is limited. The weak point of his treatment is excessive generality that does not meet the feature of accuracy that data handling must contain. In fact, using this system, one verifies the impossibility of representing in complete and addressable fashion those details that are of determinant importance in a data structure.

At this point we must consider the experiences that, entirely outside of any mathematical rigor, aim at a wholly descriptive and linguistic code. The majority of existing solutions are of this version, either those devoted to cataloging information or those aimed at building databases for use in elaborative programs. Beyond differences that one may find in the various works, there are common features that limit the discussion. Everytime one wishes to translate a literal description into mathematical language (i.e., to a language that a machine can understand), he is confronted with a series of problems that are seldom solved in an efficient way. This is especially true when the features of the language being translated are very informal and the object argument is poorly standardized. Most of the authors examined chose a solution that provided the formal description of reactions by using only starting and final products. So they pay attention only to the changes a structure undergoes during a transformation, neglecting mechanistic or logical considerations. This approach appears very efficient in the use of databanks in planning synthesis. In the course of automatic analysis, the datum is controlled, elaborated, and readapted many times, and the final result is seldom ambiguous. The question is very different in the case of collecting reaction data for retrieval. In fact, in this field the choice of using simple forms BEFORE–AFTER (i.e., containing only the description of starting and final products) contains many inaccuracies that are unacceptable to any user. Thus, we are obliged to accept a number of corrections that try to narrow and to better specify the real meaning of the reaction that is described. This is a great handicap, for two disparate but consequent reasons: everytime one attempts to further specify a description, he finds difficulties in determining the kind of specification; everytime one has stated a new descriptive addition, a diminuition of effectiveness appears.

For these reasons, most attempts were discontinued, leaving only two works, Ziegler's and Vall's.[27,25] Both of these authors reached similar results, either formally or practically. Disregarding the BEFORE–AFTER solution, they tended to complete it, inserting other features connected to mechanisms (real or supposed) and to environmental peculiarities. Eventually, the exact analysis of the obtained results shows a structure composed by three major sections: (1) description of the changes undergone by a reaction center; (2) description of the features that the reaction provides; (3) description illustrative of the starting and final products.

(1) This first aspect does not show ambiguities in code methods since one has to catalog molecules and this problem has already been solved. On the other hand, the identity of the so-called reaction center may be discussed. If we think, for example, of the reaction $CH_3COCH_2COOEt + RBr$ (B) $\rightarrow CH_3COCHRCOOEt + BH^+ + Br^-$, we may identify the obvious change $-CH_2- \rightarrow -CHR-$, succeeding in making ambiguous the real meaning of the reaction (α-alkylation of β-keto ester, that may be confused with any kind of alkylation obtained in any way). Thus, the attention is directed to that piece of molecule that, in a mechanistic sense, makes the reaction possible. In the previous example $-COCH_2COOEt \rightarrow -COCHRCOOEt$, the meaning of the reaction really becomes clearer, but the building of the necessary algorithms becomes more complex. One is obliged to force the encoding of the reaction center obtaining a compromise (e.g., always limiting the analysis to three atoms or encoding a stated transformation of the center in advance).

(2) The second point in discussion is certainly the most critical but the least clear. The meaning of "feature" is vague; in fact, one has to use long lists of terms describing various transformations, and this solution is clearly far from the desired one. Unfortunately, this section does not show noteworthy optimization, further underlying the lack of comprehension of the problem. In fact, the two mentioned experiences use the expedient we condemn above. In one case, a list of chances is built from which the most congruent is chosen (Vall); in the other, a series of unchecked key words representing the situation is used (RIRS–Ziegler).

(3) There is little to discuss about the third aspect since it is a simple encoding of structures.

All of this shows a situation anything but promising in the field of chemical reactional data management. In fact, the proposed transformations, except those strictly numeric, show a structure of the information completely uncompatible to optimize in the machine form. This means that it is nearly impossible to discuss the translation of information to a machine language form that may be easier to manage (updating, deletion, insertion, retrieval). Some observations, however, have to be made about a more suitable system that allows finding the representations. Naturally, the real string has to be in some place of the memory (principally on mass memory), and it cannot become anything but a series of characters (which is the meaning of string) connected to a system faster in automatic handling. At this level, the aspects of working with the computer come in: occupied space, speed, and usefulness. In this case, we have to verify the necessity of comparing and evaluating carefully the different availability of central and mass memory, with their respective bulk and speed. It would be preferable to maintain in central memory everything useful for retrieving the single record to obtain a good efficiency of the managing system. This solution is practical only for very small data structures and at the risk of encountering considerable problems in updating. In small collections, it may be unimportant to optimize the speed factor at the expense of other features, while in big sets this is of central importance. A partial solution may be realized by building any kind of simple index for ordering and analyzing that is useful for the management. Obviously, this option may present some obstacles in the connection between the index and the records. Changing the whole bank may be very difficult. In this field too, the big data structures tended to use inverted indexes.[16,17,19,20] Some of the possible strategies in this field, each one with its advantages and shortcomings, supporters, and detractors, are as follows: to resort to structures, frequently very complex, of lists and pointers (lists, crossed lists, threaded lists, symmetric lists, etc.); to connect primary indexes to subindexes devoted to updating; to build, in the most complex way, the space unoccupied or recovered (megaarrays, supporting arrays, scatter storage techniques, garbage collection systems, etc.). However, a noteworthy series of works and treatises exists about these aspects, and we are not interested in explicitly speaking about them.

On the other hand, another point is important to this work: making the system wholly "at the disposal" of inexpert people. This need could be partly passed over in the structures case but is more pressing in the reactions case. We must realize that the users of the databank usually have a very vague idea about their request itself, and a complex method of research could discourage them from the automatic approach. On this subject a phrase of J. Valls is very elegant and fitting: "I am convinced chemists badly need a Reaction Documentation and when such a tool is put at their disposal they use it intensively."[25] Unfortunately, this aspect is often underestimated, mainly by the organizations primarily involved in the implementation and popularization of databanks. We have a "two-faceted" problem that explores the possibility of using a databank without attempting to optimize its management. Experience shows that even the best machine organization is entirely useless if the user cannot obtain any information because of the excessive specialization of the inquiring methods. This problem is particularly pressing in the case of reaction databanks, where the error possibility and difficulties for the operator increase.

For example, we may describe two extreme cases: the first when we want to research every record pertaining a stated structure; the second when we search experimental ways to make a transformation (knowing starting and final products). Obviously, the inquiring in the first case is very simple and practically free from language problems when the obstacle of codifying a structure has been overcome. In the second, there exists an a priori difficulty inherent to the kind of question that cannot be exactly specified. This difficulty cannot be eliminated, and it will be magnified by the system protocol. Thus, we return to the two faces of this problem: that of "tearing up" the request from the user and that of interpreting the answers obtained. The difficulty in this aspect is evident, a difficulty common to every question of "kindness" in automation field. Existing solutions choose the creation of "telescopic" inquiring languages, allowing the user to specify his request more exactly. Although not the best solution, this allows the operator to better understand his problem and to make an acceptable request.

THE INFORMATION SECTION[15,28]

From the viewpoint of the software essential to manage a databank, many things are implicit in our discussion to this point: what concerns the translating algorithm between datum and machine and part of the observations relative to storage, maintenance, and management. In this section we are interested in considering again the problem of different storage schemes and their features and the problem of the programming languages. Two main methods of storing data collections exist: arrays (mono- and multidimensional) and lists. The fundamental difference between them is as follows: in the first case, an a priori ordered system is built, while in the second one the problem of connection is solved in another way, leaving unordered the individual record. This causes the differentiation of the whole managing field as well: one must consider different techniques in updating, deleting, and inserting. The problems that appear in the two cases are intrinsic to the system kind and they will mainly concern addressability for arrays and transferability for lists.

The decision to use arrays as the manner of storage also means one must consider perspective the intrinsic structure of this method. In the term array, we include all those sets that, in one, two, or more dimensions, keep some elements; everyone of them is marked by the value of indexes that specify its position, identifying the "rows" of simultaneous belonging. This characteristic of fixed (stated in advance by prearranged values) addressing builds a certain difficulty into the management, difficulty that is increased in size by two collateral considerations: the first connected to the necessity of thinking of the arrays as vectors, because this is the way the machine stores them, and the second one tied to the particular situation of data management when these are not completely standardized a priori. The transformation array–vector is not a problem too difficult to solve. In fact, we can always find an algorithm that computes the linear position of an element, in connection with the features of the language representation. However, the application of method to the different practical cases (e.g., collecting uptriangular, tridiagonal, or sparse matrices) is more complex. For each of these cases, methods and mathematical tools of less or more efficiency exist, but

we can list some common features. Given a precise structure to represent, more than one solution exist for it; the choice must be directed, besides by considerations connected to the structure, by the exact evaluation inherent in the management of the dynamic changes of the starting structure. This means that a representation has to allow the insertion of one or more elements, it has to allow their deletion or changes, and one must be able to obtain this with the smallest possible number of internal shifts. Thus, we must satisfy the optimization of the occupied space and the management of the structure obtained. Partly depending on programming language, another important feature connected to arrays is the necessity of declaring the dimension of the array in advance. The constraint of providing in a starting moment the dimension of all the arrays employed during the use of the program is a very important operative limit. Everybody, that has chosen to use arrays as a storage system, has discovered the advantage of being able to declare dimensions at execution time. This is forbidden by many languages (overall by FORTRAN), and so we had to resort to a whole set of tricks to solve this problem. In this case, the system tends to become more complex from the management viewpoint, obliging the programmer to consider problems of garbage collection, of identification of the part in the whole, and of use of decisional accessories at the level both of control and of efficiency.

The discussion of lists, both in theory and in practice, is definitely different. A list can be thought about as a series of autonomous elements ordered by the use of connections called pointers. This allows one to operate from the list without resorting to addresses derivable from indexes but by using the information given by pointers, thereby avoiding any previous ordering. Thus, any retrieval operation becomes slower, since it is necessary to make a complete tracing to recover an element (and more time is required for items near the bottom of the list); furthermore, the occupied space is greater. But there are some positive aspects; in fact, the dynamics of the system improves. The declarative necessity tied to arrays has already been described; it should be evident that for lists it does not exist. It is clear that some updating operations of an array (if we want to modify dimensions) are more complex than for lists. These are only some of the possible aspects, but what we want to underline is that all the features connected to the developing system (at execution time) become simpler to manage; and the solution is more relevant when we must use symbols that are not numeric. Given these fundamental list features, some peculiarities that distinguish them in management may be described. First of all, we may speak about the possibility of extensive modification of structure. Nearly every problem is solved without physically touching the element but by exclusively working on pointers, so the possibility of having at disposal more pointers allows us to diversify the lists used. We can obtain crossed lists, lists with a complex structure, threaded lists, etc. Obviously, the more one complicates an organization, the more difficult the problems of garbage collection, of storage bulk, of internal transfer, and of expansion become.

It is stated, however, that arrays and lists offer alternative methods for maintaining data in memory. A common aspect of the two storage systems is their applicability to the management of indexes. In many cases, these are the chosen solutions to the screening of the records and so most of the transformations are done on them. On the other hand, it is obvious that the choice of the pointer structure in maintaining the records avoids the reading of masses of useless data but requires that the indexes be worked on a greater number of times. The alternative between these choices needs to be evaluated everytime.

In thinking about the construction of an index, we have two principal solutions: the decision of maintaining the index ordered (alphabetically or in another way) or the one of using external functions. The first case shows ordering problems everytime a new subject is inserted (the more bulky and complex the index, the more difficult the handling). The second one usually requires less work in updating, but it sometimes gives ambiguous results. Providing a series of data with a logical structure means to continuously solve the problem of sorting the data. There are many ways, more or less efficient, to do this. Obviously, our interest concerns the greater facility of ordering an array than a list. The answer is not unequivocal, since the characteristics of the two storage methods suggest values in contrast: in the array case a simpler specification for inserting a new datum, but the necessity of moving a significant number of elements; in the list case the chance of changing only two pointers to obtain the insertion, but the problem of making a complete tracing to find the inserting place.

The alternative of external functions is very attractive, because it avoids ordering the indexes every time, but it also has some drawbacks. By the term "external functions" we mean the hash functions; these are a set of methods that obtain by manipulation of the datum an address to retrieve the datum itself. In any case, the great obstacle to the solution is the necessity of solving the collisions. Techniques to solve the collision problem already exist, but they all give poor results if the number of collisions is high. In any case, for the work on addresses a certain preference is for lists that are simpler to manage without other transformations.

The second strictly informational aspect of the whole question is the programming language problem, as it relates to the correctness, the efficiency, and the feasibility of a chemical databank. In this case, the transformation of existing data into mathematical formulations is rather complex, and it may be strongly influenced also by the language chosen. The analysis of the advantages and disadvantages that the various existing languages can give is an argument too big to be discussed in a few words so we only note the existence of the problem. We must, however, add that the language option may become binding in the characterization of a whole structure and that it is an excellent method to examine a priori and accurately the alternatives that are offered.

HARDWARE AVAILABILITY INFLUENCE

The hardware section of an elaborative system affects the optimization of a databank yield and many of its features. The different use of mass and central memory (characterized by speed and bulk differences) is very burdensome for running a databank structure. In fact, the examination of available possibilities in data collections has to consider their quantity and the necessity of using mass storage for memory. On the other hand, we contemplate a better research operation management maintaining both indexes and pointers in central memory. The actual trend in the hardware field is to build hardware that allows one to minimize or to cancel the difference in operation time between mass and cpu work. This realization is accomplished by connecting microprocessors to the magnetic disks. They can do a limited number of operations on the disk directly, without going through input and output operations. This creates the possibility of examining or working the entire contents of a disk with only one access. We obtain a reduction of the difference between mass and cpu execution time.

CONCLUSIONS

We have attempted to describe features, peculiarities, and problems an organic chemical databank may have, both from

the informational viewpoint (service supplier) and from that of the chemist. Acknowledging the complexity and extent of the arguments, in this section we will avoid drawing real conclusions, preferring to discuss three aspects that clarify some prospect of the field.

First, we present other people's experiences. A new era has begun in the design of alternative structures for chemical databanks. There are two places where the changes have been more evident: the part devoted to the dialogue with the user and the one connected to cataloguing reactions. This was foreseeable considering the observations in this article. The revolutionary change in the dialogue field was the proposal and online introduction of graphics. A French databank and Chemical Abstracts Service chose drawings as a means of exchange informations with operators. From the narrow viewpoint of data collection, the change is not exceptional, but from the viewpoint of usefulness, it is a revolution. The changes in the reactions code field are less conclusive and are concerned more with data cataloguing methods than with a real change of the philosophy of data content. Thus, in spite of an effort to simplify the life of the users, we cannot affirm that the results are in this field near to an acceptable solution.

The second aspect, in which we are interested, concerns the peculiar field of small data collections, which could, for example, be useful in a laboratory or for a research team. Transferred into small dimensions, the problem of maintaining and reusing informations may become simpler since all the difficulties of mass storage disappear, but it may also produce new and different obstacles, because of the diminishing of available opportunities. Possible solutions may have to be defined to interface such systems to large databases. At a small level, it is necessary to limit the collection to those items that are topical in team or in individual research. This is not only for the obvious impossibility of replacing the general databank but also for the different use that a topical record has. Another aspect of the small collections is the informational one; very often it is useless to attempt to optimize the management programs, as data volumes and users' numbers are limited. However, from the viewpoint of the machine–operator dialogue, protection, clearness, and continuous control of the conversation, the system must be optimized since in some cases there may be users that are inexpert on the computer. Thus, it is a good rule to provide to the outside freedom and flexibility of information exchange but to keep inside a strict control of error possibilities.

Finally, on the third point, it is interesting to attempt to find possible developments and trends in the narrow field of organic chemical data format. The present base-line trend is represented by those observations appearing in the most recent works, and it concerns the enhancement of query languages and the necessity of building something entirely different in the reactions code. Concerning the first argument, we may already see interesting new facts that can assure the future of databanks used by nonexperts. This means that in the future the dialogue with automatic structures will be more and more user friendly and the translation problems will be solved more and more in a transparent way to the final user. Thus, we will be able to see a large increase in the use of such systems.

The future for the reaction argument is less clear because, being in a "naive" period, it does not allow the specification of its exact shape. It is impossible to say what is the actual trend of the research and what will be its possible developments. It is clear that much work needs to be done in the coming years if we are to be able to use machines not only to collect but also to elaborate chemical data.

REFERENCES AND NOTES

(1) Petrarca, A. E.; Lynch, M. F.; Rush, J. E. "A Method for Generating Unique Computer Structural Representation of Stereoisomers". *J. Chem. Doc.* **1967**, *7*, 154.
(2) Hoffman, W. S. "An Integrated Chemical Structure Storage and Search System Operating at Du Pont". *J. Chem. Doc.* **1968**, *8*, 3.
(3) Gund, T. M.; Schleyer, P. v. R.; Gund, P. H.; Wipke, W. T. "Computer-Assisted Graph Theoretical Analysis of Complex Mechanistic Problems in Polycyclic Hydrocarbons. The Mechanism of Diamantane Formation from Various Pentacyclotetradecanes". *J. Am. Chem. Soc.* **1975**, *97*, 743.
(4) Moreau, G. "A Topological Code for Molecular Structures. A Modified Morgan Algorithm". *Nouv. J. Chim.* **1980**, *4*, 17.
(5) Shelley, C. A.; Munk, M. E. "Computer Perception of Topological Symmetry". *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 110.
(6) Quadrelli, L.; Bareggi, V.; Spiga, S. "A New Linear Representation of Chemical Structures". *J. Chem. Inf. Comput. Sci.* **1978**, *18*, 37.
(7) Corey, E. J.; Petersson, G. A. "An Algorithm for Machine Perception of Synthetically Significant Rings in Complex Cyclic Organic Structures". *J. Am. Chem. Soc.* **1972**, *94*, 460.
(8) Esack, A. "A. Procedure for Rapid Recognition of the Rings of a Molecule". *J. Chem. Soc., Perkin Trans. 1* **1975**, 1120.
(9) Roos-Kozel, B. L.; Jorgensen, W. L. "Computer-Assisted Mechanistic Evaluation of Organic Reactions. 2. Perception of Rings, Aromaticity, and Tautomers". *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 101.
(10) Esack, A.; Bershon, M. "Computer Manipulation of Central Chirality". *J. Chem. Soc., Perkin Trans. 1* **1975**, 1124.
(11) Wipke, W. T.; Dyott, T. M. "Simulation and Evaluation of Chemical Synthesis. Computer Representation and Manipulation of Stereochemistry. Stereochemically Unique Naming Algorithm". *J. Am. Chem. Soc.* **1974**, *96*, 4825–4834.
(12) Bershon, M. "An Algorithm for Finding the Synthetically Important Rings of a Molecule". *J. Chem. Soc., Perkin Trans. 1* **1973**, 1239.
(13) Blair, J.; Gasteiger, J.; Gillespie, C.; Gillespie, P. D.; Ugi, I. "Representation of Constitutional and Stereochemical Features of Chemical Systems in the Computer-Assisted Design of Syntheses". *Tetrahedron* **1974**, *30*, 1845.
(14) Nakayama, T.; Fujiwara, Y. "BCT Representation of Chemical Structures". *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 23.
(15) Berztiss, A. T. "Data Structures: Theory and Practice"; Academic Press: New York, 1975.
(16) Lefkovitz, D. "The Large Data Base File Structure Dilemma". *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 14.
(17) Fisanick, W.; Mitchell, L. D.; Scott, J. A.; Vander Stouw, G. G. "Substructure Searching of Computer-Readable Chemical Abstracts Service Ninth Collective Index Chemical Nomenclature Files". *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 73.
(18) Dromey, R. G. "Inverted File Structure for Molecular Formula and Homologous Series Searching of Large Data Bases". *Anal. Chem.* **1977**, *49*, 1982.
(19) Rusch, P. F. "Introduction to Chemical Information Storage and Retrieval". *J. Chem. Educ.* **1981**, *58*, 337.
(20) Dessy, R. E.; Starling, M. K. "Information Retrieval and Laboratory Data Management". *Anal. Chem.* **1979**, *51*, 924A.
(21) Hendrickson, J. B. "A Systematic Characterization of Structures and Reactions for Use in Organic Synthesis". *J. Am. Chem. Soc.* **1971**, *93*, 6847.
(22) Corey, E. J.; Wipke, W. T.; Cramer, R. D., III; Howe, W. J. "Techniques for Perception by a Computer of Synthetically Significant Structural Features in Complex Molecules". *J. Am. Chem. Soc.* **1972**, *94*, 431.
(23) Hendrickson, J. B. "Systematic Synthesis Design. IV. Numerical Codification of Construction Reactions". *J. Am. Chem. Soc.* **1975**, *97*, 5784.
(24) Wipke, W. T.; Howe, W. J. "Computer-Assisted Organic Synthesis". *ACS Symp. Ser. No. 61.*
(25) Wipke, W. T.; Heller, S. R.; Feldmann, R. J.; Hyde, E. "Computer Representation and Manipulation of Chemical Information"; R. E. Krieger: Huntington, NY, 1981.
(26) Ziegler, H. J. "A New Information System for Organic Reactions". *J. Chem. Doc.* **1966**, *6*, 81.
(27) Ziegler, H. J. "Roche Integrated Reaction System (RIRS). A New Documentation System for Organic Reactions". *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 141.
(28) Forsythe, A. I.; Keenan, T. A.; Organick, E. I.; Stenberg, W. "Computer Science: A First Course"; Wiley: New York, 1975.