An Approach to Automated Vocabulary Control in Indexes of Organic Compounds, II*

CHARLES H. DAVIS
Graduate School of Library Science, Drexel Institute of Technology, Philadelphia, Pa. 19104

Received April 21, 1969

The feasibility is demonstrated of using an authority list and an appropriate computer program to reduce the scattering of index terms which results from the use of synonymous organic compound names. An authority list was developed and built into a computer program as a dictionary look-up routine. Entries from *Index Chemicus* were read into an IBM 7040, and the "parent" compound names were matched against the terms in the authority list, using a binary search for optimal speed. Names found in the list were converted to preferred index terms before being printed out with the information associated with them. Time and cost figures for the operation show clearly that it is feasible, not only for indexes of organic compounds, but for any index where synonymy is the chief problem.

In the August, 1967, issue of J. CHEM. DOC., the author briefly described three algorithms which might be used to reduce the scattering in an index which results from the use of synonymous organic compound names.[1] One approach, which dealt with the conversion of parent compound names only, looked especially straightforward. This paper describes a demonstration of the feasibility of this approach on a much larger scale.

Chemical nomenclature offers far better prospects for automated vocabulary control than natural language text, because one knows in advance that only nouns are going to be encountered. Thus, there will be no semantic problems of the type found in the classic example, "time flies like an arrow," because there can be no confusion resulting from ambiguous subject-predicate relationships. It is the existence of problems such as this that have prevented a major breakthrough in the area of automatic indexing. However, there have been several important achievements in the development of better tools for linguistic analysis, and they are outlined and discussed in the *Annual Review of Information Science and Technology*.[2,3,4] Probably the best specific achievement which could be noted is the development of compiler languages such as COMIT, SNOBOL, LISP, etc., for greater ease in the manipulation of natural language text by computer.

Despite the advantages offered by systematic chemical nomenclature, few studies have been made concerning it using the new tools of linguistic analysis. Notable exceptions which are germane to this discussion are the early work by Garfield and a recent paper by Vander Stouw *et al.*[5,6]

Perhaps the chief reason that systematic nomenclature has not been investigated more often is that no truly systematic nomenclature standard for indexing has been developed. The best single example, the nomenclature used by *Chemical Abstracts*, is riddled with exceptions to its own rules, because a great number of trivial (unsystematic) names have been allowed to become established in the indexes. The following example of a *CA* rule is typical:

> For a few acids, such as acrylic, crotonic, sorbic, oleic, elaidic, propiolic, and tetrolic acid, *CA* uses common names rather than Geneva names; this applies to both the unsubstituted and the substituted acids. In addition, the trivial names methacrylic acid, angelic acid, tiglic acid, and isocrotonic acid are retained by *CA* for the unsubstituted acids only. Substituted derivatives of these acids are entered under the name of the proper straight-chain acid.[7]

Once this happens, the editorial staff is reluctant to change the names to conform with better nomenclature, because it would mean changing countless cross references and headings. It would also make the indexes more difficult to use for retrospective searches; and if the changes were made at the wrong time, it could also vastly increase the difficulty of preparing the cumulative indexes.

The nomenclature rules used for this study are similar to, but more consistent than, those now used by *Chemical Abstracts*. They are similar in that they follow the order of precedence of functional groups established by *CA*, but individual names are much closer to IUPAC's recommendations. Since these rules are lengthy and were developed for purposes in addition to the purposes of this study, they will not be discussed in more detail here.

CONSTRUCTION OF THE AUTHORITY LIST

In the case of programs for automated vocabulary control such as the ones described in the earlier paper of this title, the authority list which is used is actually built into each program. Because one program, CHEM 2, operated with an authority list which contained only six items, a much larger authority list was constructed in order to show that the program would still run quickly, and also to get a reasonable idea of the percentage of names which would be converted in a test case on "real" data.

The revised program contains the expanded authority list and is in alphabetical order, beginning with ACETALDEHYDE and ending with XYLIDINE. There are 195 entries in the list, and they were selected randomly from the authority list used by *Chemical Abstracts*. The direction of the cross references (and, therefore, the choice of the preferred index terms) was determined by the nomenclature rules mentioned earlier.

* Excerpted and revised from the author's Ph.D. dissertation, which was accepted by the Graduate School of Indiana University on January 31, 1969.

COLLECTION OF DATA

A recent bimonthly issue (Volume 30, Number 14, Issue 255, September 30, 1968) of *Index Chemicus* was chosen as the source for the data. Virtually all organic headings having unique parent-compound names, which totaled 741 entries, were selected for the sample.

The input data are identical with the original entries, except in two respects. First, the parent compound names were changed from plural to singular. Second, the locants, which would normally appear before the name of the parent, were moved from the end of the inverted data to their usual position. These changes merely constitute a change in format, and in no way alter the nature of the experiment. The locants which precede the name were left-justified in a field comprised of the first ten columns of each card. This field is not large enough to handle all conceivable entries, but it was adequate for all 741 of the sample entries. The compound names were begun in column eleven. Since the inverted part of each name was inconsequential to the experiment, the abbreviations and symbols used in *Index Chemicus* were retained.

THE PROGRAM

Although the basic algorithm for the conversion of parent compound names remained the same as that for CHEM 2, two important revisions have been made so that a wider range of compound types could be handled. First, provision was made for locants associated with the parent compound names. Second, the program was expanded so that it could deal with cases where substituent names were implied in the name of a parent. A detailed description of the program follows.

The first rule spaces the printer to the top of a new page and prints out a heading, indicating that the listing which follows consists of changed names in their original order. The READ instruction reads one 80-column card at a time and skips to a rule which by concatenation finds the first ten columns (the field for initial locants), everything following up to the comma of inversion, the comma itself, a space, and everything following (i.e., the inverted part of the name). As before, addressable storage is used for the temporary removal of data other than the parent compound name. The field for initial locants, which may be entirely blank, is stored in one location; the comma of inversion, the blank, and the inverted part of each name are all stored elsewhere. The parent compound name, represented by the string of characters from column eleven to the comma of inversion, is compressed and sent to a list subrule which contains the authority list. Comparisons between the parent name and the names of the authority list are then made using a binary search.

There is also an instruction which is activated only if a data card is read which fails to meet basic format requirements. This feature is especially important in finding missing commas of inversion, because it is this comma which signals the end of the string of characters comprising the parent compound name.

During the binary search through the authority list there can be two eventualities, either a hit or a miss. If no match occurs, then control falls through to a rule immediately following the authority list. This rule shelves (i.e., places in temporary addressable storage) the name of the parent and calls for the string of 10 characters from the field immediately preceding the parent compound name. A test is then inserted to see whether the field contained only blanks. If it did, then control is routed to a rule which places the blanks, the parent, the comma of inversion, and the entire inverted part of the name in proper sequence. Then the program duplicates the entry, writes one copy on magnetic tape, and punches the other on the on-line card punch. If the field preceding the name of the parent is not entirely blank but contains locants, then control is routed to a rule which finds the initial string up to a blank (i.e., the locants, commas, and hyphen), a blank, and all the trailing blanks, if any; this same rule then switches the order, so that the locants are right-justified in the field and in their natural location with respect to the rest of the name. The program then falls through to a rule which calls for the name of the parent and the inverted part of the name and puts them in order. At this point, the writing and punching operations described before are performed.

Whenever there is a match during the binary search through the authority list, control goes to one of several rules, depending on the nature of the conversion which is required. In the case of a straightforward conversion from one parent compound name to another, control goes to a rule initiating a sequence of operations similar to the preceding operations for a parent not on the list; the difference between the two sequences is that one operates on the original parent compound name, whereas the other operates on the converted name from the authority list. One other difference is that the new name is written on the printer instead of magnetic tape; this is done merely to keep changed and unchanged names separate for record keeping during the experiment.

In the event a name is encountered which has a substituent implied in it, then the name is changed to the preferred parent name, followed by a comma of inversion and the name of the implied substituent. TOLUENE, an example of such a name, is converted to BENZENE, 1-METHYL-. The conversion takes place in the built-in authority list, and the new name is immediately stored. Control then goes to a rule which begins another sequence to test whether any initial locants are present. If the initial locant field is blank, control is sent to a rule which shelves the blanks and calls for the original comma of inversion, the blank, and the substituents. Control then falls through to a rule which replaces the original comma and the blank following it with the initial blanks and the new parent with new inverted part, respectively, followed by the original substituents, if any. Following that there is a rule which duplicates the entry and prints one copy while punching the other.

Two things should be noted with respect to this portion of the program. First, TOLUENE is only one example among several which will follow this pattern; ACETANILIDE, which is converted to ETHANAMIDE, N-PHENYL-, is another. Second, the "1-" preceding "METHYL-" in the case of the conversion of TOLUENE is necessary only when other substituents are present; if others are not present, the locant is not wrong, but is merely superfluous. Provision was not made for omitting the locant, because it would have added to the program-

ming without contributing substantially to the quality of the output. Essentially the same sequence of operations is followed in the event that locants are encountered in the initial field; they are treated as before, with the switch which causes them to become right-justified.

An interesting case arises with XYLENE and similar compounds, because locants designating the positions of two implied substituents appear in the field preceding the name of the parent. For example, 2,3-XYLENE becomes BENZENE, 2,3-DIMETHYL-; or META-XYLENE is changed to BENZENE, META-DIMETHYL-. The problem here is one of transferring the locants from the initial field to the inverted part of the name. The program has a sequence of operations which performs this task. After the parent has been converted and stored, a rule calls for the contents of the initial locant field and falls through to a test for an all-blank field. This is necessary because the locants, which ordinarily would be present, may not have been known. If this is the case, then control goes to a rule which shelves the blank characters and calls the original inverted part of the name into workspace. Control then falls through to a rule which calls for the initial field, the new parent name, inserts the comma of inversion and the blank, and then follows with "DIMETHYL-" and the remainder of the original substituents, if there are any. If there are initial locants, they are transferred from the initial locant field and inserted along with the "DIMETHYL-" in the inverted part of the name. In either case, the initial locant field is made blank, and control is routed to a rule which performs the work of printing and punching the output as before.

Another type of conversion which must be allowed for is the case in which a parent name implies locants or stereochemical symbols in the initial locant field. Two examples of this are ACETONE and FUMARIC ACID, which are converted to 2-PROPANONE and TRANS-BUTENEDIOIC ACID, respectively. These entries are handled by reducing the number of blanks in the initial locant field so that the total number of blanks and new characters being introduced equals ten. Thus, in converting FUMARIC ACID, control is sent to a rule which initiates a routine reducing the number of blanks to four and makes way for TRANS-. Similarly, control is routed to this rule after converting TRIMESIC ACID to 1,3,5-BENZENETRICARBOXYLIC ACID, because 1,3,5- and TRANS- have the same number of characters. Two other rules initiate routines identical with this, except in the number of blanks which are deleted. Another similar rule differs in that it handles cases in which the old name implies both an initial locant and a new substituent; for example, ACETOPHENONE becomes 1-ETHANONE, 1-PHENYL-.

After the data cards are read, the READ instruction encounters an end-of-file mark. Control then falls through to a rule which inserts an end-of-file mark on the tape containing the unchanged names; the tape is then rewound. At this point the program spaces the printer and writes out the heading LIST OF UNCHANGED NAMES on the printer. Then a routine begins which reads the entries on the tape and writes them out on the printer. When all the names have been written, the tape is rewound and the program is terminated.

In practice, this program, like CHEM 2, could be run with a fast-sorting program such as IBSORT to merge the changed and unchanged names before printing the index. Alternatively, the cards could be run through a sorter. In any case, the procedures involved are straightforward and already well established.

RESULTS

Of the 741 names in the sample, 71 or 9.6% were changed. The total time required for the running of the program (including processing, execution, and loading) was 9 minutes, 1.4 seconds. Using the IBM 7040 in this manner at a rate of $150.00 per hour, the total cost of running the program is $22.56. Since the sample was taken from a bimonthly index, one could project that the total annual cost would be six times this amount, or $135.36. Alternatively, the cost could be read as 3¢ per entry.

With regard to core storage, 15,980 registers were available after compilation of the program. Of these, 15,628 were unused. Thus, it can be seen that a much larger authority list could have been built into the program. Estimates of the total number of entries which can be introduced using a 32K machine range from 4000 to 5000.[8] In any case, the authority list could be made larger than the 195 in this experiment by at least one order of magnitude.

Examination of the output revealed that all parts of the program as it is written functioned properly. Names which were encountered that were not represented in the authority list were not altered in any way and appeared under the heading LIST OF UNCHANGED NAMES. Names which were encountered that did appear in the authority list were converted exactly as planned.

The bulk of the conversions are of the straightforward type involving the simple replacement of one parent compound name for another. There are several, however, which involve not only the replacement of the parent name, but also the insertion of the name of a substituent which was implied in the old parent name. All of these conversions took place smoothly, but they have one potentially troublesome feature. That is that the implied substituent has always been put first in the inverted part of the name, and this can lead to wrong alphabetization of the substituents according to the given rules. As was pointed out earlier, this is not too serious, since the gross alphabetical order of the entries is determined by the name of the parent. However, it could cause a certain amount of scattering within each heading. It is true that all converted names at a given heading would be treated alike and would appear together, but if an entry at that heading already existed where the substituent name appeared near the end of the inverted part of the name, then scattering within the heading would result. It should be noted that no attempt had been made by *Index Chemicus* to put the substituents in the sample entries into any order. Moreover, to insert the new substituent name in alphabetical order would require features in the program which would greatly slow it down and use considerably more core storage.

The size of the authority list can be increased greatly without greatly affecting the speed of the program because of the binary search. The present authority list of 195

items requires a maximum of eight comparisons, but a maximum of only nine comparisons is required for an authority list which is twice as large. An authority list ten times the size of the one which has been used in this experiment would operate at most with just 11 comparisons. In general, when a binary search is used on a list of N items, the maximum number of comparisons required is given by $\log_2 N + 1$, truncated to the nearest integer.[9]

The program would have run even faster had more names been converted. This is a natural consequence of the fact that all unchanged names require the maximum number of comparisons during the binary search, whereas changed names normally require fewer than the maximum number. Naturally, the number of changes made is determined, in part, by the authority list used.

COMIT is especially well suited for a study of this kind, because it combines excellent string manipulation with the list subrule for performing binary searches. But since it is a compiler-level language, it contains redundant routines, and therefore uses more time than is really necessary. In a commercial situation, it would surely be advisable to rewrite the program in machine language to obtain optimum performance.

Another important factor to be considered is the choice of input and output devices. For this study, the slowest possible devices were chosen, namely, the card reader and the card punch. In a commercial situation, it would not be necessary to use these particular devices, and all the operations could be performed using magnetic tape or disk units for maximum speed. However, since there are certain advantages to having cards as physical records—they can be read by humans and can be manipulated by unit-record equipment—it was thought that time and cost figures should be obtained for the slowest operation which might conceivably be useful.

One situation which is not amenable to processing by this program is the case involving semantic ambiguity. This particular program has been devised to handle problems of synonymy; that is, cases in which several terms designate only one referent. When one term is used to designate more than one referent, then the algorithm is of no help. A specific example of such a case is the term "hexenal," which can denote either an aldehyde containing six carbon atoms and a double bond or a barbiturate, which is quite different chemically. However, since problems of semantic ambiguity in organic chemical nomenclature are far less common than problems in synonymy, the usefulness of the program is not seriously impaired by this shortcoming.

Another difficulty, which has been alluded to earlier, is the inability of the program to consider all the contingencies introduced by the presence of substituent names in the inverted part of an entry. A good example of this is a hypothetical entry at SUCCINIC ACID, METHYLENE-. The program would change this name to BUTANEDIOIC ACID, METHYLENE-, which is an improvement from the standpoint of systematic nomenclature, but is nevertheless inadequate, because the compound *in toto* should be called 2-PROPENE-1,2-DICARBOXYLIC ACID according to the rules. Programs could be devised, perhaps along the lines of programs CHEM 3 and CHEM 4, which were discussed in the earlier paper, that would take substituent names into consideration. However, these programs might be so complex that they would require both an unconscionable amount of time to run and an unrealistic amount of core storage.

CONCLUSION

The need for vocabulary control is self evident. It is a vital part of the process involved in bringing together the user of an index and the information he is seeking. This study has described a method of data manipulation which facilitates this process.

Areas for further research are suggested by this study. First, there is the problem of chemical nomenclature, which needs a rigorous and comprehensive set of rules designed with indexing in mind. Past failures do not mean that this task is impossible. Second, investigations should be continued to determine the feasibility of working with entire chemical compound names, rather than just the names of parent compounds. It is entirely possible that a truly systematic nomenclature would allow for computer-based substructure searching, a technique allowing a user to query the system with respect to moieties present within molecular structures. The paucity of research in this area has been attested to in a review by Tate.[10] Third, attempts should be continued to resolve problems arising from semantic ambiguity, as well as synonymy. Fourth, studies should be made of the effectiveness of the binary search algorithm when used with various direct-access auxiliary storage devices for extremely rapid information retrieval from extraordinarily large files of data. Fifth, a study involving the "see also" type of cross reference between terms at different generic levels might prove fruitful.

It would seem that the system which has been described for handling parent compound names, which is relatively fast and inexpensive, could be used in either of two possible situations: (1) where the output is to be used in the production of an index such as *Index Chemicus*, which is partially automated, and (2) where the output is merely the first stage in the reduction of scattering, the rest of which is to be accomplished by a professional index editorial staff. In the first case, vocabulary control is introduced as part of the automatic procedures involved in index preparation, thus increasing the usefulness of the index to the user without sacrificing the speed with which the index is produced. In the second case, index preparation is facilitated by lessening the burden of the editorial staff, thus allowing the index to be produced more quickly without sacrificing the quality which at present can only be obtained through the efforts of highly trained human beings.

This technique is not limited in its applicability to indexes of organic compounds. It can be used in the preparation of any index, catalog, or other listing where problems of semantic ambiguity are minimal and synonymy abounds.

ACKNOWLEDGMENT

I wish to thank the staff of the Research Computation Center, Indiana University School of Medicine, for the use of their facilities. The center is supported in part by Public Health Research Grant FR 00162-03.

LITERATURE CITED

(1) Davis, Charles, H., "An Approach to Automated Vocabulary Control in Indexes of Organic Compounds," J. CHEM. DOC. 7, 131–34 (1967).

(2) Simmons, Robert F., "Automated Language Processing," *Ann. Rev. Inform. Sci. Technol.* Vol. 1, p. 137–169, Wiley, 1966.

(3) Bobrow, D. G., J. B. Fraser and M. R. Quillian, "Automated Language Processing," *Ann. Rev. Inform. Sci. Technol*, Vol. 2, p. 161–86, Wiley, 1967.

(4) Salton, Gerard, "Automated Language Processing," *Ann. Rev. Inform. Sci. Technol.* Vol. 3, p. 169–99, Encyclopaedia Britannica, 1968.

(5) Garfield, Eugene, "An Algorithm for Translating Chemical Names to Molecular Formulas," J. CHEM. DOC. 2, 177–79 (1962).

(6) Vander Stouw, G. G., I. Naznitsky, and J. E. Rush, "Procedures for Converting Systematic Names of Organic Compounds into Atom-Bond Connection Tables," J. CHEM. DOC. 7, 165–69 (1967).

(7) Chemical Abstracts Service, "The Naming and Indexing of Chemical Compounds," p. 27N, ACS, 1962.

(8) COMIT II: "Computer Programming for Primarily Non-Numerical Tasks," p. 101, Winter 1966. (Unpublished, but available from Victor Yngve *et al.*, currently of the Linguistics Department and the Graduate Library School, University of Chicago.)

(9) Davis, Charles H., "The Binary Search Algorithm," *Am. Doc.* 20, 167 (1969).

(10) Tate, Fred, A., "Handling Chemical Compounds," *Ann. Rev. Inform. Sci. Technol.* Vol. 2, p. 292, Wiley, 1967.

The Double-KWIC Coordinate Index

A New Approach for Preparation of High-Quality Printed Indexes by Automatic Indexing Techniques*

ANTHONY E. PETRARCA and W. MICHAEL LAY
Department of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210

Received June 26, 1969

An indexing scheme is described which lends itself to production of high-quality printed indexes by relatively simple automated techniques. The scheme, an extension of the key-word-in-context (KWIC) indexing concept, leads to the generation of a new type of index called the "Double-KWIC Coordinate Index." This new index has many of the qualities of an articulated subject index; however, it can be produced much more simply and less expensively by automated techniques, because it does not require the depth of syntactic analysis usually needed to produce articulated subject indexes. A prototype double-KWIC coordinate index has been prepared for Volume 7 of the JOURNAL OF CHEMICAL DOCUMENTATION. The computer program for producing this index was written in PL/1. Some of the advantages and disadvantages of the Double-KWIC coordinate index are discussed, together with some areas of immediate useful application.

The need for high-quality printed indexes to facilitate manual retrieval of information has not diminished, despite the strides that have been made in the development of automatic information retrieval systems. Nevertheless, attempts to produce high-quality indexes by automated techniques have only recently begun to merit serious attention. Perhaps the most significant breakthrough in this area occurred when Luhn and others[1,2] successfully applied the key-word-in-context (KWIC) indexing concept as an automated indexing technique. The widespread use of KWIC indexes since that time and the variety of formats in which they have appeared have been reviewed by Fischer[3] and others.[4]

The rapid rise in popularity of KWIC indexes apparently has been due to the high speed and low cost of producing them. However, as noted by Fischer,[3] there has been some dissatisfaction with the quality of KWIC indexes. Most of the attempts to improve quality have dealt with variations in format to improve readability or with enrichment of titles to provide additional index entries which otherwise would not have been derived from words in the titles.

The enrichment of titles improves the quality of KWIC indexes by increasing the breadth of indexing. An equally attractive possibility, which appears to have been little explored, involves extension of the KWIC indexing principle to provide for an increased depth of indexing. If a greater depth of indexing were possible, it would help to overcome one of the major drawbacks of KWIC indexing, viz., a large number of index entries under a given keyword.

One of the difficulties encountered in such a situation is illustrated by the set of KWIC index entries shown in Figure 1, which are taken from a KWIC index derived from titles appearing in Volume 7 of the JOURNAL OF CHEMICAL DOCUMENTATION. Because these index entries are subordered on the basis of words immediately following the word in the index column, the resulting order differs markedly from the usual order one would find in a back-of-the-book index or an articulated subject index. For example, several of the entries indexed under "INFORMATION" indicate that the titles deal with

* Presented in part before the Division of Chemical Literature, 157th Meeting, ACS, Minneapolis, Minn., April 1969.