bilities and performance. The imagination required for forming hunches must be more disciplined or channeled in order to conform to the machine system and structure. This happy combination of circumstances and capacities is achievable only after intensive training and long experience. As this examiner said:

"There is a multitude of areas that you can diversify or vary for combining references, but then it is virtually impossible to code questions, on the one hand dropping feature A and on the other broadening feature B, and run all these searches and then try to composite them so you don't look at the same document time and again or even go through all the documents that fall out."

8. Mechanical search by this system did not appear to encourage browsing. The documents which are not close references often bear little relationship to the concept under examination. Because the system was generated by placing the descriptors and all of the compounds for a patent on a single card, the machine "drops out" the numbers of patents that may use an essential fragment in an entirely different type of compound than that being sought. This process is known as "compositing." As the examiner himself put it:

"Your can't browse through "no drops" or false drops. (The machine either didn't locate any references or else listed irrelevant ones.) Sometimes one of them will give you a starting point but it isn't often. The false drops are much farther from the central idea of the search than are irrelevant patents in the same subclass."

He evidently did not expect "excursions" to be rewarding.

In manual search the examiner can approach the file with several alternate questions at the same time. If he tried to apply the same technique to searching by this mechanical system, it would take repeated machine queries, considerable clerical work, and a good deal of time. A more sophisticated or larger system would permit more varied simultaneous inquiries.

In order to "browse" by machine, either the questions asked must be broadly stated, so as to cover related areas, or many more narrowly based inquiries must be successively addressed to the machine to probe areas as the examiner thinks of them. The latter practice is time-consuming unless the questioner can work "on-line" to the stored information. With a mechanized system such as the one used by Mr. Levow, an examiner would have to *intend* to "browse". Incidental but pertinent information would not be likely to arise as long as the questioner pursued a policy of trying to locate a "pat" reference with the greatest possible dispatch.

The comparison between manual and machine search in the organometallics art, as presented in this paper, has concentrated on the basic difference in mental activities that occur with use of the two systems. Further work is needed to quantify and support the hypotheses presented here. The present analysis is to be considered as developmental only.

Substructure Searching of Chemical Compounds Using Polish-Type Notation

JOHN T. WELCH, Jr.
Goodyear Aerospace Corporation, Akron, Ohio

Received March 19, 1965

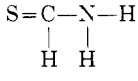
The utility of Polish notation for representing chemical compounds in a compact machine-readable form has been documented in this journal by Eisman (*1*) and Hiz (*2*). The familiar plane structural diagram may be considered a finite, connected, weighted, and, in many cases, undirected linear graph. (Graph-theoretical definitions of these terms need not necessarily concern the reader. Berge (*3*) and other basic texts are available, however.) A Polish string of symbols may then be written to completely represent any tree of the graph. By the term "substructure search," we shall mean the attempt to find a partial subgraph of the graph of the compound which matches node-for-node, *i.e.*, which is isomorphic to, the graph of the substructure. The very real limitations of these structural diagrams in representing chemical compounds will not be considered.

Eisman and Hiz are not concerned with the Polish notation as a medium for substructure search, and both show how the notation can be internally converted to other forms of representation. It is the purpose here to demonstrate the potential of the notation as an internal symbolic language for substructure search, and to indicate the manipulative and bookkeeping abilities which it brings to this task.

The particular variation of the Polish notation to be used will be described very briefly in the next section. Following that, and a general outline of the strategy to

be used in searching, the discussion will take the form of three annotated examples which progressively introduce the machine operations necessary to the search. Concluding remarks include some observations as to the role of this type of substructure search to the chemical retrieval problem.

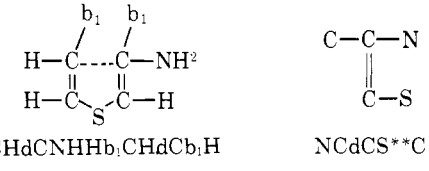
A Polish Notation. A variation of Eisman's notation will be used here, although the description to be given here is complete for the purposes of this discussion. Capital letters will represent vertices of a linear graph, which, in turn, may represent either atoms or conventionally defined fragments, *e.g.*, a benzene ring. Each symbol is followed in a Polish string by symbols which are directly bonded to it and which have not already been represented in the string. These symbols are termed the *arguments* of the original symbol. An additional proviso is that, as the string is written, the arguments of a symbol must precede the remainder of the arguments of a previous symbol. A double or a triple bond will be represented by the letters "d" or "t", placed before the argument. For example, some of the strings representing



would be

SdCHNHH	CdSHNHH
SdCNHHH	NHCHdSH
CHNHHdS	NCdSHHH

Valency information, specifying the number of possible bonds completed in a structure, is used in interpreting a string. Given this information, a string represents only one possible linear graph. The identity of an argument may be of no concern, yet the omission of the argument would cause a misinterpretation of the string. In this case, a "don't care" symbol * is used for the argument, and it is presumed to satisfy one valence bond. For example, in the substructure of Figure 1, if both "don't care" symbols are



SCHdCNHHb$_1$CHdCb$_1$H NCdCS**C

Figure 1. A cyclic compound with acyclic substructure.

omitted, the last "C" vertex would appear to be bonded to the "S" vertex, while omitting one would indicate its connection with the previous "C" vertex. For reasons to be clarified later, the "don't care" symbols are placed last among the arguments of a symbol.

A simple trial will show that the rules applied to writing Polish strings cannot be followed if a cycle (*i.e.*, a ring) occurs in the plane diagram. Thus it is required that the linear graph representing the compound be a tree (*1*). Some device is necessary to remove bonds without omitting from the string the information that they were there. The compound of Figure 1 illustrates how this is done. Instead of a bond along the dashed line shown, a pair of fictitious vertices are added, and these are connected to the participants in the removed bond, by bonds of the same kind. A sufficient number of bonds are removed in this way to produce a tree, and each fictitious pair is given a unique subscript to avoid ambiguity. A Polish string can then be written as before.

A convenient method exists for storing the valence information of a compound diagram in a string expanded and stored for computer use. As the compound of Figure 1, for example, is presented for inclusion in the compound file, an algorithm referring to a valence table can place left parentheses (l.p.) and right parentheses (r.p.) to expand the string into the form

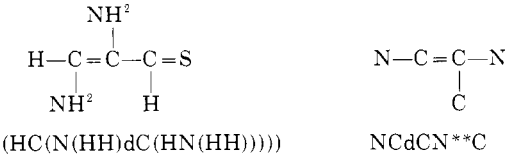
S(C(HdC(N(HH)b$_1$))C(HdC(b$_1$H)))

This string is put into the computer store, and subsequent programs which manipulate strings in search need not refer to valence tables, since that information is now represented in the string. In the expanded string, the portion lying between parentheses corresponding to a symbol will be called the *substring* of the symbol, *e.g.*, the substring of the first C in the above string is HdC(N(HH)b$_1$).

A Strategy for Substructure Searching. A given compound can be represented by many Polish strings quite different in appearance. It is possible to define, however, a number of operations which produce, from a given string, others which have certain desirable characteristics and which represent the same compound. We propose a method of substructure searching in which the initial compound string is manipulated, using these operations, into a form in which the substructure string, with "don't care" positions occupied, appears as an initial, contiguous portion. In general, this procedure will involve backtracking, as initial claimants to a compound string position produced later contradictions. The compactness of the Polish representation, however, allows backtracking to be carried out by generating, as each new substring symbol is examined, all the compound string variations which meet the previous and present conditions, *i.e.*, which match the substring up to the examined symbol. Naturally, an upper limit must be placed on the number of variations generated, the compound being passed as *possibly* containing the substructure when this limit is exceeded. It should be mentioned too that a careful choice of the substructure string often minimizes the number of generated variations.

Since the detailed algorithims of the total search strategy and operations involved would be of little interest here, we shall introduce parts of the strategy in a series of progressively more difficult examples, while describing the operations functionally as they first arise. Algorithms for these operations are not hard to formulate.

An Acyclic Compound with Acyclic Structure. We begin with the compound and substructure of Figure 2, with the compound string in machine form. The first task is to write two strings for the compound, each beginning with a nitrogen symbol. After scanning the string and noting



S(dC(HC(N(HH)dC(HN(HH))))) NCdCN**C

Figure 2. An acyclic compound and substructure.

the position of the nitrogen atom, the string is inverted with respect to that atom. To indicate the type of machine operations involved, inversion with respect to a symbol X could be performed in the following sequence of steps:

(1) Write X(and set $k = 1$.
(2) If X is followed by a left parenthesis (l.p.), copy the substring of X; if not, do nothing.
(3) Read back from X. If first encounter is d(t, q, ...) write d(t, q, ...) next; if not, do nothing.
(4) Read back through first unsatisfied l.p. to next symbol Y, and set $k = k + 1$.

Note: An unsatisfied l.p. is one for which the corresponding r.p. was not in the sequence being read.

(5) Copy Y(. . . , up to X, or dX if d was written in (3) etc.
(6) If X is followed by l.p., copy between r.p. of X and r.p. of Y; if not, copy between X and r.p. of Y.
(7) If Y is the first element of the old string (a) delete the last mark on the new string if it is a l.p., and (b) add k r.p.'s and stop. If not, substitute Y for X and repeat from (3).

In our example, inversion with respect to the two N's leads to two strings

$$N\overset{\vee}{(}HHC(dC(HN(HH))C(HdS)))$$

$$N\overset{\vee}{(}HHC(HdC(N(HH)C(HdS))))$$

The second substructure symbol read is "C," and a search for arguments of "N" which match "C" is begun in the compound string, according to the rules

(1) When a l.p. is encountered, the search is suspended until a count of parentheses shows it to be satisfied by a r.p.
(2) The search ends when the r.p. of "N" is detected by the same count.

Each match is then promoted to a position beside "N" by moving the argument and its substring to the left, if necessary. An algorithm similar to the one described above can do this. In this case, the new strings generated are

$$N(C\overset{\vee}{(}dC(HN(HH))C(HdS))HH)$$

$$N(C\overset{\vee}{(}HdC(N(HH)C(HdS)))HH)$$

When "d" is encountered next, a branch is taken so that the argument symbols are searched for a matching "dC." After this is promoted, and the next match for "N" is also promoted, the compound strings have become

$$N(C(dC(N\overset{\vee}{(}HH)H)C(HdS))HH)$$

$$N(C(dC(N\overset{\vee}{(}HH)C(HdS))H)HH)$$

A caret marker is being used to designate the position from which new searches should begin. It should be noted also that parentheses take the correct positions as new strings are generated, even though their positions in the substructure string are not specified.

Now the first "don't care" symbol is encountered. At this point searches begin in the substring of a previously matched symbol, and since "don't care" symbols are always the last arguments listed for a symbol, there are no more symbols to be matched in this substring. The number n of remaining arguments in the substring must be determined, however, so that the n "don't care" symbols which refer to the remaining arguments of the substring can be considered as matched. This is done in the following way.

(1) A count is begun after the caret marker of the first compound string.
(2) Vertex symbols are counted once, "d" is counted once, "t" is counted twice, etc.
(3) When a l.p. is encountered, nothing is counted until the corresponding r.p. has been read.
(4) When an unsatisfied r.p. is encountered, the count is complete, since the substring has been completely read.
(5) The count is now n, so n "don't care" symbols in the substructure string are designated as being satisfied.
(6) The caret marker is placed over the first unsatisfied r.p. in each variant of the compound string.

In the example at hand, the count is 2 and another "don't care" symbol is read from the substructure string. The compound strings are

$$N(C(dC(N(HH\overset{\vee}{)}H)C(HdS))HH)$$

$$N(C(dC(N(HH\overset{\vee}{)}C(HdS))H)HH)$$

A search is now made for matching "C." It is found in the second string, while in the first, the unsatisfied r.p. terminates the search before a "C" can be found. The final output is one string, namely

$$N(C(dC(N(HH)C\overset{\vee}{(}HdS))H)HH)$$

A Cyclic Compound with Acyclic Substructure. In the first example, complication due to cycles and the removed bonds necessarily corresponding to them was intentionally avoided. Now the compound and substructure of Figure 1 can be considered. As is evident from the structural diagram, the break at b_1 must be moved elsewhere if the given substructure is to be detected.

The machine form of the compound string is

$$S(C(HdC(N(HH)b_1))C(HdC(b_1H)))$$

and inversion with respect to N produces

$$N\overset{\vee}{(}HHC(b_1dC(HS(C(HdC(b_1H))))))$$

while searches for C, dC, and S lead to

$$N(C(dC(S\overset{\vee}{(}C(HdC(b_1H)))H)b_1)HH)$$

According to the rules of the previous example, the combination C(HdC(b_1H)) is taken to satisfy the first "don't care" symbol, and the string is

$$N(C(dC(S(C(HdC(b_1H)))\overset{\vee}{)}H)b_1)HH)$$

The next "don't care" is met by "H" and a match for "C" is sought. Whenever a b_i is encountered in a search, the bond is moved by the following procedure:

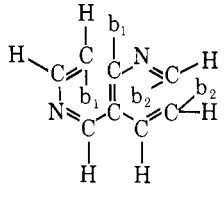
(1) Find the other occurrence of b_i in the string, ignoring b's with other subscripts.
(2) From there read back through next unsatisfied l.p. to the symbol, and copy through the corresponding r. p., here obtaining $C(b_1H)$.
(3) Tack on any preceding d or t, enclose the result in parentheses, and add a preceding b_j, where j is not the subscript of one of the breaks in compound. We have $b_2(dC(b_1H))$.
(4) Invert the result with respect to b_i and strip away b_i and corresponding parentheses; this produces $C(Hdb_2)$.
(5) Substitute the results for the troublesome appearance of b_i.
(6) Substitute b_j for the expression found in (2).

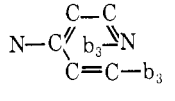
In the new string now produced, namely

$$N(C(dC(S(C(Hdb_2))H\overset{\vee}{)}C(Hdb_2))HH)$$

the final match for "C" is found.

A Cyclic Compound with Cyclic Substructure. A new complication is introduced in searching for a cyclic substructure, since removed bonds will now occur in the substructure string. The compound and substructure of Figure 3





$Nb_3dCCdCN{**}CdCb_3$

$N(dC(HC(C(HdC(Hb_2))dC(N(dC(Hb_2))b_1)))C(HdC(Hb_1)))$

Figure 3. A cyclic compound and substructure.

are considered. Inverting with respect to the second "N" symbol produces, in addition to the string of the figure

$$N(dC(Hb_2)C(b_1dC(C(HdC(Hb_2))C(HdN(C(HdC(Hb_1)))))))$$

The next substructure symbol is a "b," indicating that a break should be made on some single bond of "N." In both strings, there is only one single bond, so the position of the break is determined; otherwise a string would be generated for each case. Promoting the arguments of "N" involved, the strings become

$$N(C(HdC(Hb_1))dC(HC(C(HdC(Hb_2))dC(N(dC(Hb_2))b_1))))$$
$$N(C(b_1dC(C(HdC(Hb_2))C(HdN(C(HdC(Hb_1))))))dC(Hb_2))$$

Since the compound structure with given breaks is a tree, adding a new break without removing an old one would make the structure disconnected. The question arises, then, of which of the old breaks is moved to the new location. That it is not sufficient to generate a new string for each case is obvious from the compound diagram of Figure 3, for if "b_2" is replaced by a break beside the leftmost "N," the upper left portion is no longer connected to the rest of the diagram, which moreover contains an unbroken cycle. The question has a ready answer, however.

Since the compound diagram, with breaks, is a tree, it is well known that rejoining any break will result in exactly one cycle (see (3), for example). If the new break is not made in this cycle, then the resulting diagram will not be a tree. On the other hand, it can be shown that if the new break is made in this cycle, the result remains a tree. The problem then is to determine which breaks define cycles in which the bond to be removed is contained. Any of these breaks can then be moved around the corresponding cycle to the desired position, and a new string is generated for each case. Such breaks can be found simply. If the compound string is inverted with respect to the symbol to be replaced by a break, then such a break will be represented by a "b_i" in two different argument substrings of the symbol. In the current example, "b_1" meets this condition in the first string; "b_2," in the second.

The operation which moves the break is similar to that used to move breaks in the previous example and reduces to the following steps, given that "b_i" is to be moved:

(1) If X is the argument symbol to be replaced by a break, copy X(S), where S is the substring of X. In this case we have

$$C(HdC(Hb_1))$$
$$C(b_1dC(C(HdC(Hb_2))C(HdN(C(HdC(Hb_1))))))$$

(2) Add enclosing parentheses and an initial b_j of unused subscript. Invert with respect to b_i, then strip away b_i and corresponding parentheses, here giving

$$C(HdC(Hb_3))$$
$$C(HdC(HC(C(HdN(C(HdC(Hb_1))))dC(b_1b_3))))$$

(3) Substitute b_j for the expression copied in (1); the result of (2), for b_i in the other argument of the matched symbol.

The new strings for the example are

$$N(b_3dC(HC(C(HdC(Hb_2))dC(N(dC(Hb_2))C(HdC(Hb_1))))))$$
$$N(b_3dC(HC(HdC(HC(HdN(C(HdC(Hb_1))))dC(b_1b_3))))))$$

The remainder of the search proceeds as in the previous examples, except that b_3 cannot now be replaced by a break elsewhere. The second occurrence of b_3 can be treated like any other atom symbol, however, since it represents a legitimate (though fictitious) atom. In the example, the first string leads to a match, while the second is rejected. This concludes the examples.

Concluding Remarks. The use of a file representation of Polish form is suggested for substructure search. Conversion to Polish form from other existing representations should not be a major problem and probably could be done by computer in most cases. One of the major problems of interconversion, that of obtaining a unique or canonical form of the derived representation, can be ignored in converting to Polish form. Detailed searches, such as the one suggested here, should of course be applied only to the output of screening procedures, if the file is large, but has the advantage of practically ensuring correct matching of the final product. The Polish output also has the advantage of showing in what ways the sought substructure occurs in the compounds retrieved.

From this viewpoint, we have attempted (1) to present the Polish notation as a symbolic language for substructure search, (2) to indicate the nature of computer algorithms employed in search, and thus, (3) to show that no substantial difficulty arises in the treatment of the simplest kind of structural diagrams.

LITERATURE CITED

(1) Eisman, S. H., *J. Chem. Doc.*, 4, 187 (1964).
(2) Hiz, H., *ibid.*, 4, 173 (1964).
(3) Berge, C., "The Theory of Graphs and Its Applications," John Wiley and Sons, New York, N. Y., 1962.

Rapid Structure Searches *via* Permuted Chemical Line Notations. III. A Computer-Produced Index*

CHARLES E. GRANITO,[a] JOHN E. SCHULTZ,[b] GERALD W. GIBSON, ALAN GELBERG,[c] R. J. WILLIAMS,[d] and E. A. METCALF
Industrial Liaison Office, Office of the Technical Director, U. S. Army Chemical Research and Development Laboratories, Edgewood Arsenal, Maryland 21010

Received March 5, 1965

INTRODUCTION

The previous papers in this series (*1*, *2*) have discussed the concept of an index of permuted Wiswesser chemical line notations,[e] the significance of a QUICK-SCAN area, and simple methods for preparing this type of index for a small index file of compounds (up to *ca.* 5000). It has been pointed out that the preparation of an index for a large number of compounds would require the use of a computer. This is the subject of this paper.

The project was started in 1963. At that time, a Univac File Computer (Model II) was readily available and, therefore, used for preparing the index. After the program was written and satisfactorily tested on a trial deck of 1000 cards, approximately 55,000 Wiswesser chemical line notations, on file in this office, were indexed. A program which achieves this same result has been written for an IBM 1401 at the T. R. Evans Research Center of the Diamond Alkali Company. Both programs are available for potential users.

The discussion of a general program to accomplish this permutation will be divided into four categories: (1) computer preparation of the index, (2) cost of preparing an index, (3) the index, and (4) uses of the index.

1. Computer Preparation of the Index. The input is a single punched card per compound, containing an accession number, a two-column screen, and a Wiswesser chemical line notation. The program is designed to effect the permutation of the line notation as each card is read onto magnetic tape; *i.e.*, the operations required to select pertinent symbols, generate the scan area, and permute the notation are accomplished and the results stored prior to acceptance of the next line notation.[f] The path followed by a typical line notation card in these operations will be discussed rather than giving the step-by-step details of the less understandable directions and flow charts of the programmer.

For the purpose of this discussion, it is convenient to think of the input information as occupying one row of a compartment in the core memory (Memory I, Figure 1). For easy visualization, the memory row which can hold 120 characters is further divided into four areas: A, B, C, and D (Figure 2). Each area is separated by a blank space to make the final printout more readable, and the following arbitrary assignments are made: (a) area A (8 spaces) for the accession number, (b) area B (2 spaces) for a prefix to serve as a screen in the index, (c) area C (11 spaces) for the QUICK-SCAN symbols to be generated by the computer, and (d) area D (96 spaces) for the line notation.

Areas A, B, and D contain information read directly from the card. At the start, area C is empty; it will be filled by symbols, selected by the computer in its operations on a line notation, for which an entry will be made in the index. The line notation will be found in the last half of area D (spaces 73 through 120); initially, the first half of area D (spaces 25 through 73) is empty. Space 73, the center of area D, corresponds to the index column of the listing.

After all of the information on one card has been fed into the input section of Memory I, the computer is ready to start generating the information for the QUICK-SCAN area and the permutations of the line notation. The notation is transferred to a reserve memory, Memory II (see Figure 1). In this transfer and all subsequent transfers of the notation, the information in areas A, B, and C is asso-

* Presented before the Division of Chemical Literature, Symposium on Work and Time Studies in Technical Information, 149th National Meeting of the American Chemical Society, Detroit, Mich., April 1965.

[a] To whom all inquiries should be addressed.

[b] Westminster College, Fulton, Mo.

[c] Diamond Alkali Co., Painesville, Ohio

[d] Data Processing Division, Management Science and Data Systems Office, Edgewood Arsenal, Md.

[e] Some notations, used as examples in this paper may not be consistent with the revision of Wiswesser line notation rules currently being prepared for publication by Dr. E. G. Smith, Mills College, Calif.

[f] It should be noted that for the Honeywell 200 400 the first step is a card to tape conversion, with the permutation a subsequent action.