Automatic Interpretation of the Texts of Chemical Patent Abstracts. 2. Processing and Results

G. G. Chowdhury and M. F. Lynch*

Department of Information Studies, University of Sheffield, Sheffield S10 2TN, England

Received March 17, 1992

Part 1 of this series described the lexical isolation and categorization of the text tokens of statements describing generic structures in the texts of Documentation Abstracts from Derwent Publications Ltd.;[1] this paper describes the syntactic and semantic processing of the tokens with a view to producing the corresponding GENSAL expressions. The syntactic analysis proceeds as an expectation-driven process; the result of the analysis is then validated by semantic information associated with each token. The prototype system can satisfactorily process 86% of the 545 descriptions studied. Routines for processing variable expressions, multiplier expressions, nested parameter expressions, nested substitutions, compound token declarations, and conditional expressions are described. Messages are automatically produced calling for manual intervention in the 14% of statements which are beyond the scope of the prototype system. The prototype could be implemented either for retrospective conversion of databases of generic chemical structures from printed sources or could be adapted to serve as an intelligent editor during preparation of patent abstracts.

1. INTRODUCTION

The first part of this paper described the background and the lexical isolation and categorization phases in methods for semiautomatic conversion of *assignment statements* from the text of abstracts in Documentation Abstracts into the corresponding GENSAL expressions.[1] In this paper, the actual processing stages are detailed together with an evaluation of the degree of success attained to date. Processing involves both syntactic and semantic analysis. Syntactic analysis is a phase in NLP where valid sentences of a language are recognized and their underlying structures are determined, whereas semantic analysis involves the use of semantic information to represent the meaning of natural language texts. The lexically categorized tokens, in this study, are first processed syntactically, and the results are then semantically validated in order to arrive at a correct resolution. Once a given expression is interpreted, it is passed to the appropriate subroutine(s) to produce the GENSAL expressions corresponding to the input assignment statement.

The process of syntactic analysis follows an expectation-driven approach; at each stage, the system expects the occurrence of the tokens in a typical pattern and attempts to match each expression with one of a number of predefined conditions. These predefined conditions are written in the form of production rules involving the condition in the IF part and the corresponding measure in the THEN part. These predefined conditions or alternative paths reflect common patterns of expressions in assignment statements. The options available to the system and the bases for selecting paths are briefly described. Issues to be considered for further improvement of the prototype for implementation are also mentioned.

2. EXPRESSION PATTERNS

A study of assignment statements (Section 4 of Part 1) revealed certain regular patterns of expressions in the abstracts. Figure 1 shows the different paths available at the first level of processing, and Figure 2 shows common patterns of multiplier and variable expressions. Assignment statements usually begin either with a variable or a multiplier expression.

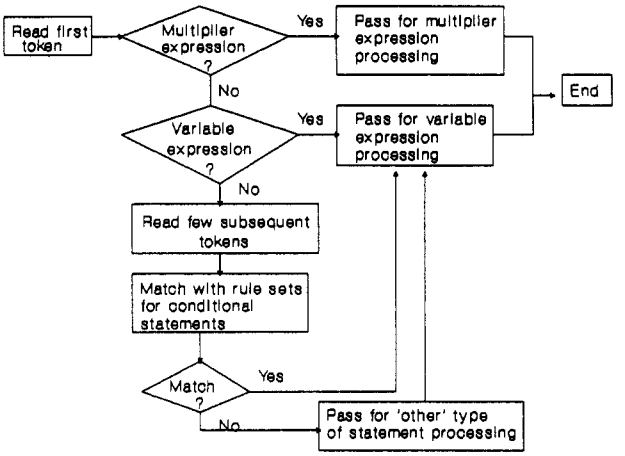

Figure 1. First level of processing.

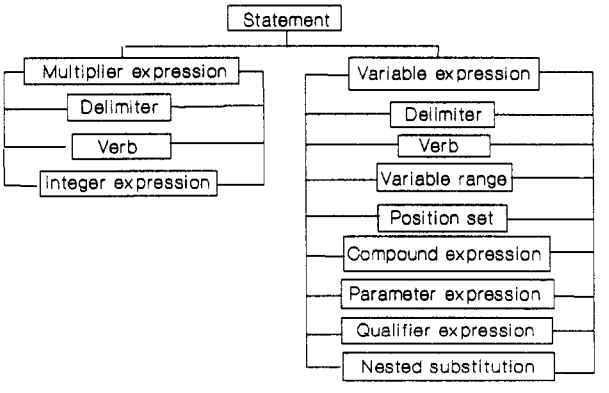

Figure 2. Common expressions patterns.

However, conditional expressions can also occur. These basic patterns form the first level of expectation matching (Figure 1). When the first token is processed, a check determines whether it is a variable or a multiplier expression; otherwise, the system matches the expression with five types of conditional statements. If none of these conditions is satisfied, the system attempts to categorize it as part or a continuation of a previous variable or conditional expression.

0095-2338/92/1632-0468$03.00/0

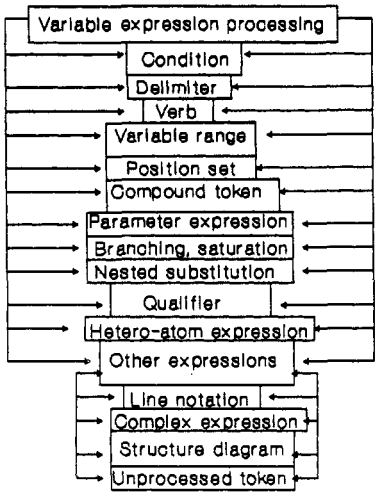

Figure 3. Routines for variable expression processing.

Multiplier expressions are generally simple and usually take the form of an integer or integer range for the multiplier group. Variable expressions, on the other hand, are quite complex, and different types of expressions may be embedded in a single assignment statement—Figure 3 shows the various routines called, which are also detailed below. Thereafter, the semantic information associated with each token is examined in order to substantiate the match, and the expressions are then passed for conversion to the corresponding GENSAL expressions.

3. MULTIPLIER EXPRESSIONS

Multiplier expressions usually consist of one or more integers or of a range of integers. Four subroutines comprising 17 rules were formulated to determine the nature of the multiplier declaration and generate the corresponding GENSAL expressions. However, multiplier expressions can also form part of a variable declaration statement and are then handled within the routines for variable expressions.

4. VARIABLE EXPRESSIONS

Twelve subroutines, shown in Figure 3, deal with different expression types. They involve a total of 180 rules, which may be applied iteratively. They become most complex in cases of nesting of parameter expressions, substitutions, etc. as well as in cases of compound token expressions, conditional expressions, etc.

Line notations and partial structure diagrams occurring in assignment statements also call for special treatment. The database creation module, MicroGensip, developed for International Documentation in Chemistry by Barnard Chemical Information Ltd., treats line notations, e.g., $-CH_2CH_2-$, as specific nomenclatural terms which retrieve an equivalent connection table from the system dictionary. Handling partial structure diagrams in MicroGensip calls for the use of the graphics mode where the user inputs the structure diagram. The present prototype system automatically identifies the occurrence of partial structure diagrams and produces a message for the user indicating this, but automatic processing within the graphics module has not yet been undertaken.

5. SEMANTIC INFORMATION

The syntactic analysis is validated at each stage through semantic information associated with each token, which is attached to each token in the dictionary lookup phase

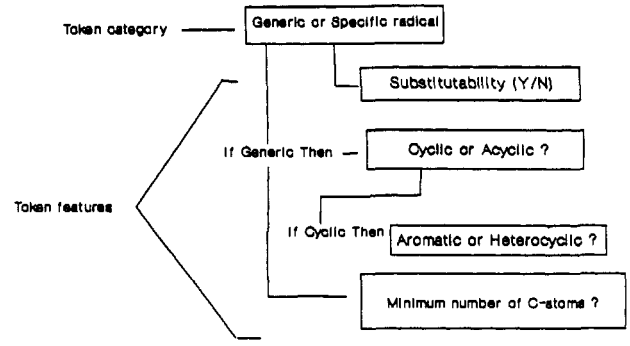

Figure 4. Semantic features of radical names.

(discussed in Section 8 of Part 1). The semantic features of each token and their importance in processing are reflected in the data structure of the dictionary. The semantic features included in token definitions in the dictionary are related to the parameters initially developed to qualify generic radical expressions in GENSAL. The features are necessary to identify the extent of qualification of series of radical names, particularly in nested expressions, such as

'R2 is 1–5C alkyl, opt. halo-substd. 2–5C alkenyl or 2–5C alkynyl'

'Q is phenyl or phenylene, substd. by 0–3 nitro, halo, CF3, or R5'

'R3 is halo or 1–4C alkyl, opt. substd. by up to 3 halo'

Again, a common practice in Documentation Abstracts is the declaration of a range of carbon atoms followed by a number of radical names separated by delimiters like comma or 'or'. Which radicals may have the declared parameter (e.g., the declared range of carbon atoms) needs to be identified. In some cases the parameter applies only to the following radical, while in other cases it may apply to some or all of the following radicals. A knowledge of whether each of the following tokens is a generic or specific radical is necessary, because a parameter expression denoting the range of carbon atoms is applicable only to a generic radical. However, even if the radical is generic, the parameter expression giving a range of carbon atoms will not always be applicable; 'halogen' is an example. Confusion may also arise in compound expressions like '2–5C alkylthio' and '2–5C haloalkyl'. In the former case, the expression denotes 'thio substituted by alkyl group having 2–5 carbon atoms', or in GENSAL 'thio SB alkyl (2–5)'. The same analysis will produce an incorrect result in the latter case—'alkyl SB halo (2–5)'; the correct expression is 'alkyl (2–5) SB halo'. Knowledge of the applicability of a parameter to a particular token can resolve the ambiguity. Analogous circumstances arise in other contexts; branching applies to acyclic radicals, aromaticity to cyclic radicals having some minimum number of atoms, etc.

The data structure of the dictionary was chosen so that some simple characteristic features were available to disambiguate assignment statements. They are not comprehensive and could readily be extended.

The characteristic features used here were identified on the basis of 'literary warrant', i.e., they appear in the texts of Documentation Abstracts. In cases of nested substitution, with expressions such as '... opt. substd. ...' or '... substd. ...', one has to know which of the preceding or following tokens can be substituted. This requires an indication of substitutability. Further, a radical may be cyclic or acyclic. If cyclic, it may be alicyclic, aromatic, or heterocyclic. If acyclic, it may have unsaturation (double or triple) or branching (ternary or quaternary) features. Successful identification of the extent of nesting is further facilitated by the carbon atom range. The

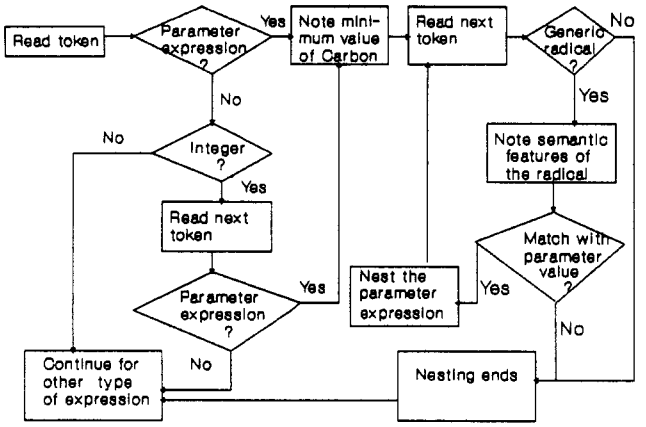

Figure 5. Parameter expression processing.

lower limit of a range of carbon atoms, for example '2' in '2–5C' can exclude nesting of cyclic radicals, which have a minimum of three carbon atoms.

6. SEMANTIC TOKEN FEATURES

Figure 4 represents the different semantic features associated with the generic and specific radical names. The features comprise the following:

- substitutability (applicable both to generic and specific radicals)
- cyclic or acyclic nature; if the radical is cyclic, then (a) whether it is aromatic or (b) whether it is heterocyclic
- limits on ranges of applicable parameters, e.g., if cyclic, the permissible carbon atom range

These semantic features are used at one stage or another in the processing routines discussed below.

7. NESTED PARAMETER EXPRESSIONS

Processing becomes especially complex when a qualification applies to more than one of the generic radicals which follow or precede it. For the former case, the decision relates to the cutoff point where the process of nesting should end. A simple answer to this could be: *The given parameter expression will apply to all the following generic radicals, and the process of nesting will end as soon as a specific radical is met.*

This is one of the necessary conditions, but it is not sufficient to disambiguate all the situations. A given parameter expression may not apply to all generic radicals, and in fact there are certain limits to this. For example, in the statement *R1 is 1–5C alkyl, alkenyl, alkynyl, or aryl*, the parameter expression '1–5C' applies to the generic radicals 'alkyl', 'alkenyl', 'alkynyl', etc. but not to 'aryl', so that checks of this kind must still be made.

Figure 5 shows a graphical representation of the decision rules that govern the processing system for handling parameter expressions. Sometimes parameter expressions are represented in a way which is different from the normal style of *integer followed by hyphen followed by integer followed by upper case C*, as in '2–5C', e.g., 'R1 is 2–5C alkyl or 3 or 4C alkenyl or 3 or 4C alkynyl'. In this example, the first parameter expression applies only to 'alkyl', whereas different parameter expressions are declared for the other two generic radicals. The lower range of carbon atoms in these parameter expressions occur in the form of a single integer (here '3') rather than in the form '3–4C'. The decision rule that determines the parameter expression here can be stated as follows: If ambiguity occurs, the system is generous, i.e., it tends to err on the side of inclusion of a wider variety of circumstances rather than excluding some which are dubious. *If a given token is an integer then read the next token, and if it is a parameter expression (in this case it will be one integer followed by 'C' as in '4C') then the first token (that is the integer) denotes the minimum value in the range of carbon atom possible for the following generic radical.*

8. NESTED SUBSTITUTION

Variable expressions often refer to radicals which are further substituted or optionally substituted by one or more radicals. Some common examples follow:

'R1 and R2 are H or opt. substd. alkyl or aryl'

'R is 1–8C alkyl, 3–6C cycloalkyl, or Ph (opt. substd. by halogen, methoxy, NO2, or carbalkoxy)'

'R2 is alkyl opt. mono- or disubstd. by methyl, ethyl, ...'

'R1 is substd. alkyl, alkoxy, alkylthio, (di)alkylamino, or aryl'

'R1, R2, and R3 are each H, opt. substd. 1–8C alkyl, 2–5C alkenyl, phenyl (opt. substd. by halogen, NO2, CH3, OCH3, and/or COOH), or opt. substd. heteroaryl'

Different types of phrases denoting substitution occur here, viz., **'substd.', 'opt. substd.', 'opt. substd. by', 'opt. mono- or disubstd. by', 'opt. substd. by 0–3 ...', 'opt. substd. by up to 3 ...', 'opt. halo-substd.'**, etc. GENSAL prescribes a specific format for each of these. Substitution in an assignment statement may refer to more than one following radical name. In some cases, however, substitution refers to radical names which precede it, for example

'R1 is 1–8C alkyl, 2–8C alkenyl, or 2–8C alkynyl (all opt. substd. by cyano, hydroxy, 1–6C alkoxy, carboxy, 2–9C alkoxycarbonyl, carbamoyl (opt. monosubstd. or independently disubstd. by 1–8C alkyl, 2–8C alkenyl, or 2–8C alkynyl), 2–9C alkoxycarbamoyl, 2–9C alkylsulfonamido carbonyl, CO-Het, or one or more halogen atoms), or 3–6C cycloalkyl (opt. substd. by one or more 1–4C alkyl groups)'

The phrase **'all opt. substd. by'** in the example indicates that all the preceding tokens are optionally substituted by a number of radicals, beginning with **cyano**, and ending with **halogen atoms**, the substituting token **carbamoyl** again being substituted by some radicals. Such substitution thus requires consideration regarding the extent of nesting. A number of complex subroutines handle commonly occurring substitution expressions. Altogether 49 rules were formulated which identify the occurrence of a particular construction that denotes substitution, determine the extent of nesting, and then produce the appropriate GENSAL expressions. The number of decisions that the system has to take results in a complex network, a simplified version of which is reproduced in Figure 6.

The system first attempts to find the occurrence of expressions denoting substitution, like **'substd.', 'opt. substd.', 'opt. substd. by', 'opt. mono- or disubstd. by', 'opt. halo-substd. ...'**, etc. (Figure 6), and then matches it against a number of conditions to determine the occurrence of significant keywords. For example, if the token denoting substitution is preceded by the keyword 'all' then the parser goes backwards to find the beginning of nesting, and then proceeds forward to determine the end of nesting. This routine may be repeated a number of times in a given assignment statement; nested substitution may even occur inside another nested substitution, as shown in the example above. In some cases, the expression

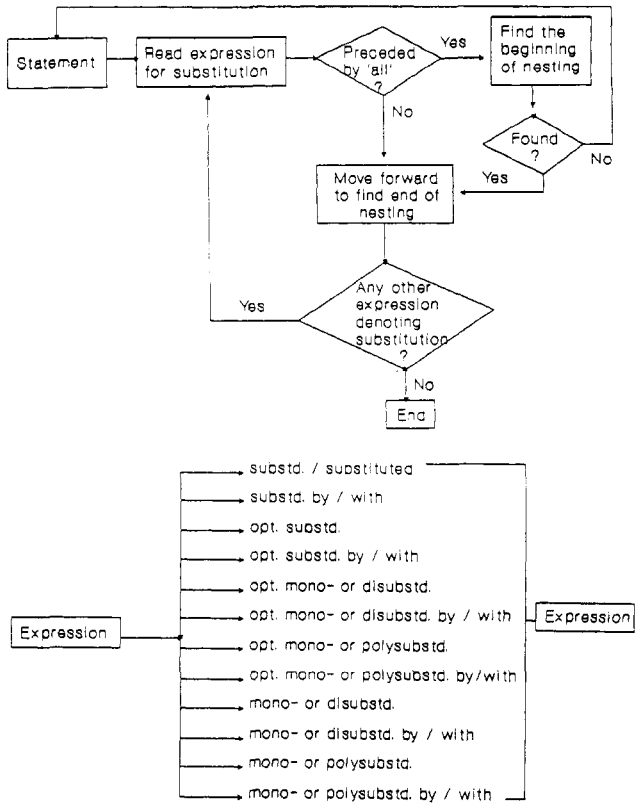

Figure 6. Nested substitution processing.

denoting substitution is followed by a qualifier or a parameter expression, or in some cases by an integer expression denoting the number of substituents. The occurrence of such expressions invokes the relevant subroutines to determine the extent of nesting required for the current parameter or qualifier expression or the number of substituents. The examples given above demonstrate that 'verb', 'qualifier', or 'parameter expression', 'conditional expression', and, indeed, any type of expression may occur within nested constructions, and thus when nesting begins the system expects the occurrence of any of these and passes the statements to almost all the subroutines written for the processing of variable expressions. It is thus an iterative process which involves the identification of nesting, finding the beginning and end of nesting, and then processing the intermediate tokens by passing them to the appropriate subroutines.

9. COMPOUND EXPRESSION PROCESSING

Compound expressions involving two or more radical names, with or without preceding parameter or qualifier expressions, are common in Documentation Abstracts. They also occur compounded with parameter expressions denoting the range of carbon atoms applicable to the given generic radical name. In GENSAL, compound expressions need to be written in their simplified form. For example, an expression like 'haloalkyl' is represented as 'alkyl substituted by halogen'. The analysis of such compound expressions becomes complex when the expressions involve or are preceded by parameter or qualifier expressions. For example, the expression **2–9C alkoxycarbonyl** is rendered as **carbonyl SB alkoxy ⟨2–9⟩**, and the expression **1–4C haloalkyl** is interpreted as **alkyl ⟨1–4⟩ SB halogen**. Thus, the system has to match each compound expression against a number of rules to determine the exact type of expression; the result is then validated with the associated semantic information to determine the applicability of the parameter or qualifier expression to a given radical; the whole expression is then passed for creation of the GENSAL expression. Altogether 24 rules are used for validation of compound expressions.

More than one variable expression can be combined in another kind of compound expression such as '**NR7R8**' and '**=CR5R6**', etc. The Microgensip interpreter requires entry to graphic mode to handle these; in such cases users are asked to interpret the expressions manually. The present prototype system can successfully identify such expressions and produce messages for the user at two stages. A message is produced for the user at the lexical isolation stage pointing out the occurrence of such compound expressions and asking the user to alter them. If the user ignores the message at that stage, another message is produced at the processing stage.

10. CONDITIONAL EXPRESSION PROCESSING

Variable or multiplier expressions often occur in the form of logical conditions. The occurrence of such conditional expressions needs to be represented in the form of 'IF ... THEN ...' logic in GENSAL. The determination of the extent to which a given part of an expression will go under the 'head' of the condition, that is under the IF part, and which will go to the 'result' part, that is under the THEN part, is often a difficult task for an expert analyst. The present prototype involves 22 rules which can handle simple conditional expressions. Common examples of conditions include the following:

R2 is 7C alkyl when one of R1 and R3 is H and the other is methyl

provided that when X is halogen, Y is methyl or haloalkyl; when X is methyl, Y is methyl; and when X is haloalkyl, Y is haloalkyl;

Such conditional expressions are represented in GENSAL by one or more 'IF ... THEN' statements, e.g.

IF R1 = H AND R3 = methyl THEN R2 = alkyl⟨7⟩

IF X = halogen THEN Y = methyl/alkyl SB halogen ELSE IF X = methyl THEN Y = methyl ELSE IF X = alkyl SB halo THEN Y = alkyl SB halo.

Conditional expressions may occur as part of a statement beginning with a variable expression, as in the first example above, or as a statement beginning with conditional expressions, as in the second example. In the latter case the conditional expression is determined at the first level of matching and is then passed to the rules for processing variable expressions (Figure 1). To handle the former type of conditional expressions, a subroutine was written as part of the set of rules for processing variable expressions (Figure 3). These rules were formulated first to determine the occurrence of a conditional expression and then to identify the condition (the IF part) and the result part (the THEN part) of the expression. Some complex routines were written for this purpose, especially with regard to expressions involving more than one 'IF ... THEN ... ELSE ...' logic.

11. RESULTS

The processing phase performs the syntactic analysis and validates the results by means of the associated semantic information, whenever necessary, and finally produces the GENSAL expressions. Each assignment statement is passed to a series of subroutines for interpretation of the expression and for reformulation in GENSAL. The GENSAL expressions for each of the 545 assignment statements occurring in the sample of 73 Documentation Abstracts were compared with the corresponding input. Instances where the whole

Chart I. Sample Input Documentation Abstracts and Their GENSAL Output

Input

```
R is 1-8C alkyl, 3-6C cycloalkyl, 1-4C alkenyl, 1-4C
  haloalkyl, benzyl, alkoxymethyl, or Ph (opt. substd. by
  halogen, methoxy, NO2 or carbalkoxy);
R1 is H, halogen, CF3, alkoxy, carbalkoxy or CN;
R2 and R3 are each H, halogen, 1-4C alkyl, CF3, alkoxy,
   carbalkoxy or CN;
R4 and R5 are each 1-4C alkyl;
X is O or S.
```

Output

```
R63  = alkyl<1-8>  /cycloalkyl<3-6>  /alkenyl<1-4>  /alkyl
       <1-4>  SB halo /benzyl / methyl SB alkoxy/ Ph (OSB
       (halogen / methoxy / NO2/
       carbalkoxy )) ;
R1  = H / halogen / CF3 / alkoxy / carbalkoxy / CN ;
R2,3 = H / halogen / alkyl<1-4>/ CF3 /alkoxy / carbalkoxy /
       CN ;
R4,5 = alkyl<1-4>;
R52  = O / S .
```

Input

```
X is halogen or CF3;
Y is halogen or alkyl;
R is methyl-substd. cyclopropyl or a formula;
Z is H or alkyl;
n is 0 or 1.
X is chlorine or bromine;
R1 is lower alkyl, 3 or 4C alkenyl or 3 or 4C alkynyl;
R2 is lower alkyl;
when X is Cl, R1 is 3 or 4C alkenyl or 3 or 4C alkynyl.
```

Output

```
R52  = halogen / CF3 ;
R51  = halogen / alkyl ;
R63  = cyclopropyl SB methyl/ SD[1] ;
R50  = H / alkyl ;
n = <0/1>;
R52  = chlorine / bromine ;
R1  = alkyl<1-6>  / alkenyl <3-4> /alkynyl <3-4>;
R2  = alkyl<1-6> ;
```

1 A message is produced at the processing stage calling for manual interpretation of the substructure diagram

output formed valid GENSAL expressions and where the major part was correctly processed, with only a small part calling for manual intervention, were regarded as successful. However, in most cases the prototype highlighted those parts of the statements which it could not analyze and informed the user of the need for manual interpretation. Such cases are considered successful, because in these cases the software identifies the failures successfully. The system produces messages in the cases of some compound token expressions, line notations, structure diagrams, and textual expressions (Figure 3). Further extensions of the prototype might enable such expressions to be processed.

Overall, 70% of the statements were processed correctly with no user intervention, while in 16% of the cases, minor corrections were necessary. Thus 86% of the assignment statements belonged to the category where either the whole statement was processed correctly or minor manual corrections were required. As mentioned in Sections 4 and 9, the prototype system cannot process line notations, partial structure diagrams, and some typical compound expressions, although in each case it comes up with a message asking for user intervention. A few input Documentation Abstracts and their corresponding GENSAL expressions (output) are presented in Chart I and II. The rest of the statements, 14%, called for major manual interpretation, mostly owing to the inadequacy of the prototype to handle some complex expression types, examples of these being shown below:

'... alkyl, alkoxy, and alkanoyl have 1–20C; alkenyl and alkynyl 2–22C; aryl 6–10C; and aralkyl 1–8C in the alkyl part'

'A is an opt. substd. 3- or 4-atom bridge, contg. 1 or 2 O, S, or N atoms, which completes a 5- or 6-membered nonaromatic heterocycle; two O atoms must be sepd.

Chart II. Sample Input Documentation Abstracts and Their GENSAL Output

```
IF R52  = Cl THEN R 1  = alkenyl <3-4> /alkynyl <3-4>.
```

Input

```
R1 is halogen or CF3;
R2 is H, halogen, CF3 or NO2;
R3 is H or halogen;
R4 is H, halogen or NO2;
R5 is H or 1-5C alkyl;
R6 is alkyl, alkenyl, alkynyl or alkoxyalkyl;
X is O or S.
```

Output

```
R1  = halogen / CF3 ;
R2  = H / halogen / CF3 / NO2 ;
R3  = H / halogen ;
R4  = H / halogen / NO2 ;
R5  = H / alkyl<1-5>;
R6  = alkyl / alkenyl / alkynyl / alkyl SB alkoxy ;
R52  = O / S .
```

Input

```
R1 and R2 are H, F, Cl, Br or Me;
R3 is H, cyano, ethynyl or 1-propynyl;
R4 is a gp. (a) or (b);
X and Y are halogen, methyl or haloalkyl;
provided  that when X is halogen, Y is methyl or haloalkyl;
  when X is methyl, Y is methyl and when X is haloalkyl, Y
is haloalkyl;
W is methylene or O;
n is 0 or 1;
Z is H, halogen or methoxy;
m is integer 1-4;
(Z)m may be methylenedioxy.
```

Output

```
R1,2 = H / F / Cl / Br / Me ;
R3  = H / cyano / ethynyl / [1]propynyl ;
R4  = (a) / (b)[1] ;
R52,51 = halogen / methyl / alkyl SB halo ;
IF R52  = halogen THEN R 51  = methyl / alkyl SB halo ;
IF R52  = methyl THEN R 51  = methyl ;
IF R52  = alkyl SB halo THEN R 51  = alkyl SB halo ;
R53  = methylene / O ;
n = <0/1>;
R50  = H / halogen / methoxy ;
m = <1-4>;
(Z)m[1]  = methylene dioxy .
```

1 A message is produced at the processing stage calling for manual interpretation of the expressions

by at least one carbon, and oxygen and sulfur atoms can be bonded to each other only if the sulfur is present as SO2 or SO gps'

'R3 is an opt. unsatd. 1–4C alkylene chain opt. substd. by 1–3 alkyl or alkylene residues with a total of 1–8 C-atoms, by a 6–10C aryl residue, or by a benzyl residue

'W is a free or functionally converted hydroxymethylene or free or functionally converted MeCOH in which the OH gp. is alpha or beta'

'R5 is thiophene, pyrrole, oxazole, thiadiazole (1,3,4; 1,2,4; 1,2,3; or 1,2,5), oxadiazole (1,3,4; 1,2,4; 1,2,3; or 1,2,5), triazole (1,2,4 or 1,2,3) residue all opt. substd. by one or two of 1–4C alkyl, haloalkyl, or R5 is a furan residue substd. by a halogen atom or by 1 or 2 methyl'

'A is an opt. unsatd. 1–10C aliphatic residue opt. interrupted by O and opt. substd. by up to 6 halogen atoms, a satd. 3–12C or mono- or polyunsatd. 5–12C cycloaliphatic residue opt. substd. by up to 4 halogen atoms and/or by one or more 1–4C alkyl or mono-, di-, or trihaloalkyl residues or a (1–4C alkoxy)-carbonyl residue; a bicyclic satd. or mono- or diunsatd. 7–12C aliphatic residue which is opt. substd. by up to 6 halogen atoms or one or more 1–4C alkyl residues or in which a CH2 bridge can be replaced by O; or a residue of formula (Ia)'

In all of these cases the prototype responded with a message identifying the unprocessed part and its location in the abstract, and indicating the need for manual interpretation. Further

software development would be necessary to include these complex declarations. This could be achieved in two ways. A preprocessor could be developed to identify irregular expressions (i.e., expressions which are uncommon in assignment statements) and to allow the user to convert them into regular expressions interactively. The other approach is to develop rules for handling each of these cases, and then to incorporate the resulting complex subroutines into the processing system to process these expressions.

12. PROSPECTS

The basic objective of this study was to develop a semiautomatic method for the conversion of assignment statements into their corresponding GENSAL expressions, and thus to facilitate automatic creation of database of generic chemical structures in patents. The results obtained so far are encouraging and indicate that a higher performance could be achieved through further software development.

This prototype could be implemented for retrospective database creation of generic structures in patents, substantially reducing the need for human involvement. It could also be used as an intelligent editor, whose role would be to produce GENSAL expressions automatically, in parallel with the process of creation of patent databases, with necessary indications of those expressions which fall outside the scope of the editor.

ACKNOWLEDGMENT

G.G.C. gratefully acknowledges funding of this work by the Commonwealth Scholarship Commission, U.K. We thank Derwent Publications Ltd for providing materials for this work.

REFERENCES AND NOTES

(1) Chowdhury, G.; Lynch, M. F. Automatic interpretation of the texts of chemical patent abstracts. 1. Lexical analysis and categorization. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, preceding paper in this issue.