Developments in SPECOL—a retrieval language for the non-programmer

B. Smith*

* *28 Queens Court, Queens Road, Cheltenham, Gloucestershire. [Mr. Smith is a Civil Servant employed in a Government Research Establishment. The SPECOL project has the support of the Treasury and several other Government Departments]*

SPECOL is a Special Customer Oriented Language for use in querying files by computer. It is a very simple and practical language that anyone can use after very little study. It requires no knowledge of computers or of computer techniques, nor any knowledge of the medium (disk/tape, etc.) on which data may be stored. The language was first described in *The Computer Journal*, Vol. 11, No. 2, August 1968. Since then there have been a number of significant developments in the language and there have been many requests for further information. The present paper sets out, therefore, a little more formally the basic philosophy of the language, and describes some of its new features. In addition a few hints are given on writing certain types of question. The paper concludes with a statement on the current implementation of SPECOL and some comments on its use in a remote access environment.

(Received July 1969)

Basic philosophy

The basic philosophy of SPECOL is that it should be seen as a bridge between natural language and the language of logic and sets. Natural language, although adequate for normal discourse, is too ambiguous for putting questions to a computer; the language of logic and sets, on the other hand, although ideal for the computer and the specialist, is not so acceptable to the ordinary person. In order to bridge the gap between the two languages, SPECOL exploits two features that are common to both languages, *firstly* the almost identical use of the words, AND, OR, and NOT; and *secondly* the way in which data is often classified or addressed in groups. The first feature is important, since it can be shown that by using AND, OR and NOT, it is possible to specify literally any conceivable combination of data. The second feature, the grouping of data into classes, is also important, since it can be shown that by attaching names or descriptors to the classes, it is possible to address vast amounts of data that otherwise might not be accessible at all. Putting information into classes is also an excellent way of organising one's thoughts. Take, for example, the game of Twenty Questions, in which, by asking a few well-chosen questions on classes, it is possible to range over the entire universe, and yet, still light on one required fact thought up by one of the players.

In logic, classes are referred to as sets and sub-sets of data; in SPECOL we think of them in terms of a book, that is consisting of paragraphs and sentences and so on. The following is a full list of the terms and their data processing equivalents.

BOOK	FILE
BOOK TITLE	FILE NAME
CHAPTER	RECORD
CHAPTER HEADING	RECORD HEADER
PARAGRAPH	TRAILER SET
SENTENCE	TRAILER/LINE
WORD	FIELD/DATA ELEMENT
CHARACTER	CHARACTER/DATA POSITION

Fig. 1 shows the record part of the concept in diagram form. The format is extremely flexible. There may, for

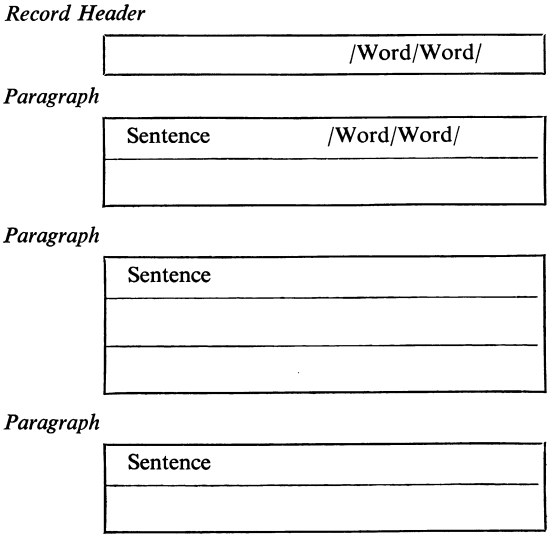

Fig. 1. Typical record format visualised as part of a book

example, be any number of sentences in a paragraph and any number of paragraphs in a record. Headers and sentences, too, may be of varying lengths, although as shown in the diagram these are usually mentally padded up to some maximum length. As another instance of flexibility, not all classes have necessarily to appear in all files: some files, for example, consist entirely of headers and here the paragraph concept would not apply. Likewise, the boundaries of the classes may be thought of as quite flexible and may be changed mentally so to speak without physically altering a file. Certainly, one of the delights of set theory is that if we do not like the universe we are in we can change it; what we think of today as chapters and paragraphs in a file, we may think of tomorrow as paragraphs and sentences. It is this concept of movable boundaries, and of looking at data as if it were part of a book, i.e. in terms of paragraphs and sentences that is so important to SPECOL and allows it to be used on almost any file.

New features

The following are some of the more important features that have been incorporated in the language since the publication of the original paper.

Multiple Questions

The first SPECOL compiler allowed only one question to be asked for each pass of the data. The latest compiler allows several questions to be asked at a time, the precise number depending on the amount of core space available. Each question requires about 6K positions of core. Allowing therefore for a 20K compiler plus a maximum record size of, say, 20K, it should be possible in a core partition of 100K to ask up to 10 questions per run. With a partition of 200K, it would be possible to ask 26 questions. The restriction that multi-SPECOL places on record size is because, unlike single SPECOL which deals completely with one line of data at a time, multi-SPECOL requires the lines again for each subsequent question. Maximum record size can be increased by making more core space available or by using drum or disk back-up. There is of course no limitation on file size. There are good reasons for using both single—and multi-SPECOL and it is envisaged that both versions of the program will be maintained.

Variable length field names

Field names may now be any combination of one to eight characters. They must still, however, begin with a letter.

New mode number

Mode 4 has been added to the list of Mode numbers indicating what part of a record is to be selected for output. Mode 4 indicates that only *sentences* that contain matched data are to be saved. Mode 1 has a similar meaning for headers, and Mode 2 for paragraphs. Mode 3 indicates that a whole record is to be saved.

The connective ANDX

The connective ANDX has been introduced to allow searches to be made across sentences. The rule states that if data is required to occur in the same sentence, use AND; if it may appear in different sentences, use ANDX, eg.

```
GIST  (NUCLEAR) ANDX GIST  (REACTOR)
FNME  (JOHN)      AND  SNME  (SMITH)
SNME  (SM.)       AND  SNME  (@ SON)
```

In the first example, NUCLEAR may appear in one sentence and REACTOR may appear in another. In the second example JOHN and SMITH must appear in one sentence, and in the third example, the data must appear not only in the same sentence but also in the same field, i.e. a name is required that begins with SM and ends in SON. The ANDX feature is similar to the TYPBX feature used for searching across paragraphs.

Counting

There are now three count commands: OVCNT calling for overall counts on selected records; INCNT calling for counts within records and overall; and INCNTP for counts within paragraphs, within records, *and* overall. Given that a file contains census data in Town and Postal district order, the following three short statements would call for records of male workers in Devon and Somerset, aged 30 to 35; and within this class, counts, by district, by town, and overall, of engineers, builders, salaries over £2,000, and the number of people who are single.

```
TYPA    CNTY (DEV. OR SOM.)
TYPB    SEX(M) AND YOBTH(1935 TO 1940)
INCNTP  JOBD(ENGR.) (BUILD.) ¢
        SLRY(> 2000) STAT(S)
```

In the output, the required counts for districts and town would be set out at the side and slightly to the right of, each record. The overall totals would be shown at the end of the run.

In a similar way the contents of fields may be added using the terms OVSUM, INSUM and INSUMP. The following statements would call for a list of salaries and pensions payable to single women typists, together with department and overall totals. Department totals would appear at the end of each record; overall totals plus average costs per department and per person would appear at the end of the run.

```
TYPB     STAT(S) AND SEX(F)
AND      JOBD (TYPIST)
INSUM    SALARY PENSION
PNTA     DEPT
PNTB     NAME/SALARY/PENSION
END
```

Quantity searching

The number of times that a set of conditions is required to occur in a paragraph or a record may be stipulated by the conventional terms *n*, <*n*,> *n*, NOT *n*, NOT <*n* and NOT>*n* written before relevant field expressions, with *n* being equal to any 1-, 2-, or 3-digit number. The operation is called Quantity searching. The following statements call for more than three doctors and not less than six women nurses,

```
    >3      JOBD (DOCTOR)
NOT <6      SEX(F) AND JOBD(NURSE)
```

Since the field name SEX is followed by an AND connective, the effect of the NOT<6 term extends over the whole statement.

Repeat-field searching

Many files have the same type of field, say personal qualifications, repeated adjacently a number of times in the same sentence. It is now possible to ask for a search of each sub-field in this area, using only one field name. SPECOL recognises the repeat-field name and compiles appropriate instructions for searching at the specified intervals. For instance, if a person's language qualifications are represented in a 12 × 4 area by a string of 4-letter mnemonics, e.g.

SPANITALFRENGERMRUSS

the following single expression would call for a qualification in Spanish or German:

QUAL (SPAN OR GERM)

On output the same field name causes the subfields to be automatically spaced:

SPAN ITAL FREN *GERM* RUSS

Interrogation of packed fields and bits

Routines have been written which permit interrogation of packed fields and bits using normal character digits as input in the question. On output the data may be produced either in its original packed form or in readable digits. Similar routines may be written for other forms of packing, e.g. octal, hexadecimal. These facilities however are appropriate only to data that has been packed according to System 360 conventions. To deal with data packed by other machines, it would be necessary to have different routines. This dependency on type of machine emphasises the need to represent data in a file, whenever possible, in the form in which a user envisages it, i.e. in character form. Only in this way may data be easily exchanged between different computers and easily printed on printers of different manufacture.

Hints on writing SPECOL

Specifying an exclusive OR

The OR connective in SPECOL is inclusive, i.e. indicates 'either or', or 'both'. To obtain the exclusive OR, an expression must be followed by the appropriate negative. For example, if we require cases of malformations in children of cleft-palate or hare-lip but not both, we could write:

```
      MALF (CP OR HL)
NOT   MALF (CP) AND MALF (HL)
```

The MALF field in this instance would be defined as a repeat field as described in the preceding section.

Factorising common data

When a request contains alternatives, it is usually good practice to factor out common data and to specify this first. For example, to specify single women or married men at BRISTOL who are under 23 we might write:

```
TYPA   UNIV (BRISTOL) AND YOBTH (> 1946)
AND    STAT(S) AND SEX(F)              OR (
       STAT(M) AND SEX(M)
```

This device avoids having to write out BRISTOL and 1946 more than once.

Specifying figures or letters

Since the character collating sequence of system 360 is letters followed by figures, it is possible to specify 'any figure' by the term, >Z and 'any letter' by the term, <0. For example, to retrieve, say, shoe codes that begin with any figure followed by any letter (such as 1A . . ., 3B . . ., or 4X . . ., etc.), we could write:

SHOES (> Z.) AND SHOES (<.0.)

The dot before the 0 in the second expression indicates that at this point we have already dealt with the first character.

Conditional output

Sometimes even when a search is successful, it may still be required to output certain lines only when they contain specified data. Often, this can be done using a combination of Mode No. (2 or 4) and an OR search. If, for example, we require a list of scientific articles but only want to output a gist line if it contains the word, atomic, we may write:

```
MODE   4
TYPB   SUBJ (SCIENCE)
AND    STAG OR GIST (?ATOMIC)
PNTA   JOURN/DATE
PNTB   SUBJ/STITLE
AND    GIST
```

In Mode 4, sentences containing matched data only will be saved. In order to obtain sentences containing the word ATOMIC it is necessary to specify this in the request. The second TYPB statement (AND STAG, etc.) shows how this can be done without affecting the main search. In other words, if the subject SCIENCE is present, the field STAG, which is the S line identification field, must also be present. The statement will therefore be satisfied whether the word ATOMIC appears or not.

Current implementation and remote access

SPECOL is now available through IBM for most System 360 (OS and DOS) computers and through ICL for most System 4 computers. It is being written for ICL 1900 series computers and negotiations are also taking place with manufacturers about implementing it on other computers.

To date most SPECOL questions have been put to the computer in punched card form, but the language has also been successfully demonstrated in a time-sharing mode with questions being entered from remote typewriters and graphic display units. In remote access working, the procedure adopted depends in the main on the size of the file being interrogated. On small files

(say up to 200,000 lines of data) it is possible to receive fairly rapid replies to a SPECOL question and to display the results at a terminal; for larger files it would seem that the most likely future for remote access SPECOL is in the area known as remote job entry, where questions are entered remotely from a terminal and questions are immediately checked and compiled. If the question contains an error, a message to this effect, plus the offending statement, is sent back to the user and he can correct it there and then and re-submit it. He can also save his question in the computer for re-use or modification later on.

When the jobs have been run the user may display some or all of his results at his terminal or re-direct them to the printer or to some other device or user. This method is seen as a big advance over conventional batch processing, there being considerable time saving all round, perhaps the most noteworthy being the immediate correction of errors, the ability to share files, and the ability to re-direct results.

The early hopes of SPECOL have now been realised and there is little doubt that fairly large scale interrogation of data is now possible by this means. SPECOL itself has been considerably enhanced by the facility of being able to ask several questions with one pass of the file, and by its use in remote access. It is in these two areas, coupled with the ever present requirement for faster turn-round of jobs, in which greatest interest is now expected to be shown, and where probably the most significant developments are likely to take place.

Book review

Methods for Unconstrained Optimization Problems, by J. Kowalik and M. R. Osborne, 1968; 148 pages. (*American Elsevier Publishing Co. Inc.*, 100s.)

The optimisation problem is very simply stated: given a function $f(x)$, find x^* such that $f(x^*) \leqslant f(x)$. This is the unconstrained problem. If there are relations to be satisfied of the form $g(x) \leqslant 0$, or $g(x) = 0$, the problem is constrained.

An important field in which such problems arise naturally is in the statistical estimation of parameters by the method of least squares. The statistical literature is surprisingly sparse except where the normal equations are linear in the parameters. In certain particular problems, English, Scottish and Scandinavian actuaries made quite considerable, though mainly *ad hoc*, progress but by and large the difficulties encountered caused the actuarial profession to abandon optimisation as a step in curve fitting in favour of the smoothing and graduation of data, with sophisticated methods of interpolation which in many ways anticipated much of today's approximation theory. The link between optimisation and best approximation is intimate. Best L_2 approximation leads of course directly to least squares optimisation while best L_1 and L_∞ approximation problems can be reformulated as constrained optimisation problems. Whereas practical methods of obtaining best approximations make great use of techniques of mathematical programming (linear, quadratic, separable and dynamic programming), methods at present in vogue for optimisation surprisingly do not.

This book concentrates on two main lines of attack on the unconstrained problem—direct search and descent methods. In the former, the tactics are to evaluate the function at a suitable set of points and by a set of rules extend the search to further points so that the sequence leads to the optimum point. Methods described include those of Hooke and Jeeves, Rosenbrock and the simplex method of Nelder and Mead. Descent methods depend upon exploiting the geometry of the surface defined by the function in order to obtain directions which descend towards the optimum point. The methods of Davidon, Fletcher and Powell for the case when derivatives can be calculated explicitly are described in detail as is Powell's method which avoids the calculation of derivatives. Several variations are described. It is pleasing to see a unified treatment based on the analysis of the properties of quadratic forms to which functions will approximate near the optimum.

A chapter is devoted to least squares problems. In the discussion of an algorithm for regression analysis there occurs a specific use of an exchange algorithm (commonplace in approximation theory!) to replace variables in the regression at each stage. Golub's method of solving the linear least squares problem using a factorisation by elementary orthogonal matrices is analysed. Newton's method and the secant method of solving non-linear normal equations are given with the more recent methods of Levenberg, Marquardt and Morrison.

Despite the title of the book, about one sixth of the text is devoted to a chapter on constrained optimisation. The main emphasis is placed on converting the problem to one of unconstrained optimisation by including the constraints in the function to be optimised in ways which exact a savage penalty if the constraints are violated while yielding the true value of the original function at the optimum. Here similarities to the techniques of mathematical programming are discernible.

A final, valuable, chapter is devoted to the comparative numerical results of using a variety of methods on eight test problems. The authors express some surprise at the relative success of the simplex method in these comparisons. I wonder why?

I can recommend the book as a useful text on the mathematics of optimisation methods. The book however is overpriced (it is four times more costly per page than some books recently published by reputable British publishers). It is not entirely free of minor misprints.

ANDREW YOUNG (Coleraine, Ulster)