Information Technology: Research and Development (1982), *1* (113–129)

GOS: A PACKAGE FOR MAKING CATALOGUES

M. F. PORTER

Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ, England

(Received 14 August 1981, revised 17 November 1981)

ABSTRACT

GOS is a program package for creating and manipulating catalogues by computer. It was designed for museum data processing, but could equally be used with bibliographic data. It achieves great power by its ability to manipulate general tree structures of fields, rather than the linear lists of fields of the MARC system. This paper gives a brief history and overview of the package, together with examples of its use.

GOS (Porter, 1980a, b, c) is a program package for use in making catalogues by computer. Within this particular area it manages to attain considerable generality and versatility, and it can in fact be used to build almost any kind of catalogue, for example a dictionary, or a herbarium, or a directory of names and addresses, or a conventional library catalogue. But it was for museum catalogues that the package was originally written, and at present its main use is in cataloguing museum objects.

The writing of GOS began at the Sedgwick Museum, Cambridge. Here up to 1977 a series of research projects was undertaken which involved the computerization of the catalogue of the Sedgwick Museum and of the various activities associated with it, and the general study of museum data recording with the intention of standardizing, at some level, actual recording practice (Porter *et al.*, 1977). In 1978 the newly formed Museum Documentation Association continued this work for the benefit of the museum community in the UK as a whole. A full account of more recent developments in this area both in the UK and other countries may be found in Roberts and Light (1980). GOS was written jointly by Dr Jonathan Cutbill, now at the National Maritime Museum, Greenwich, and the present author. It was written in short bursts over a number of years, so that it is difficult to estimate the total amount of work that went into it, but it took in all not more than about four man years to bring into its present state. It is distributed by the Museum Documentation Association, at Duxford Airfield, Duxford, Cambridgeshire, to whom all enquiries should be addressed. It consists currently of about 14000 lines of source code, divided into over 80 separate modules. It is written throughout in BCPL, and considerable emphasis has been placed on portability.

The initial emphasis on museum catalogues was salutary in designing GOS, because museum data are very varied in nature, and often exceedingly complex. If

GOS is capable of handling museum catalogues to the very exacting standards demanded by museum curators, then it is probably capable of handling any other kind of catalogue as well. Museum data tend to be complex (more complex for example than bibliographic data), and for a number of reasons. In the first place, almost any kind of object can find its way into a museum, whether it is an electron microscope photograph of a fossil coccolith, or an aeroplane from the Second World War. The information that one may wish to record about these objects can therefore be very varied. Again, museum objects have different degrees of importance, and the amount of information that one may wish to record about a single object is very variable. It may be a single line of information, or it may run to several pages. Another important point is that a museum object may be structurally very complex, and one may want the corresponding data description to reflect this. For example the object may be a set of surgical instruments in a box. One may wish to record the various materials from which they are made, the methods of manufacture, the names of the manufacturers, and the inscriptions on the instruments, and to keep clear the precise relation between the items of data about the manufacturing, and the instruments to which the data refer. Finally, there are no settled conventions that the museum curator can readily refer to which will tell him how precisely his data are to be recorded. There is no single guide to which he can turn in the way that a librarian can turn to Anglo-American Cataloguing Rules (Gorman and Winkler, 1980) for example.

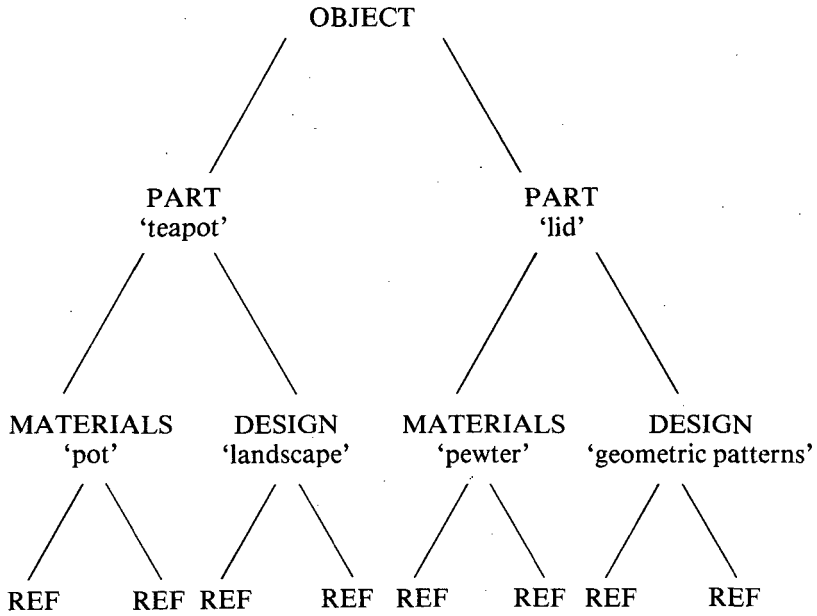
In designing GOS therefore we had to produce a set of programs with a considerable degree of generality. GOS would have to handle very large and very small records, complex records and simple ones; and we could not make any presuppositions about the maximum size or degree of complexity of the records it would be handling, or anything else concerned with the general structure of the data.

The essential point about this generality is revealed in a description of the structure of records which are handled by GOS. Each *record* of information in GOS is made up of a collection of separate *fields*. Records in the MARC Bibliographic System (UK Marc Manual, 1980) have a similar structure. In a MARC record there is a directory of fields used, and a set of fields which give a first level breakdown of the data in the record. Within each field, strings of text may be separated by various markers which effectively supply a second level breakdown of the data in the fields. MARC records can therefore be said, very roughly, to have a two-level structure. In GOS there is provision for record structures which go arbitrarily deep, so that N-level structures may be created, N being as large as we please. At first sight this may seem to be an over generous provision, but in fact it is enormously advantageous. It means that many details of relationships between items of data that are either left vague or established by some unnatural convention in a system which has a strictly limited number of levels can be established by grouping the items together at a higher level.

This point is illustrated by the example (admittedly somewhat artificial) of the picture opposite.

The record here is supposed to describe an *object*. The object is in two *parts*: a teapot with a lid. Each part has recorded the *materials* from which it is made and the *design* which is on it, and each material and design item of data has two bibliographic *references* associated with it. Thus the fact that the lid is made of pewter is supported by two REF headings under 'pewter', and similarly for the other items. The REF headings themselves will have their own data with their own internal structure, but we will not concern ourselves that far. The point is that the relation-

ships between these items of data, which are so easily represented in a hierarchical structure, would inevitably be lost or obscured if the whole was forced into a linear structure, or any structure with too few levels.



The structure in this example is of course a tree structure, and each record handled by GOS in fact consists of a tree structure of fields. It is arranged that the items of actual data, which can be of various types (strings, integers, reals . . .), occur at the ends, or leaf nodes of the trees, and that the higher levels in each tree simply represent groupings of the lower level components.

The example tree structure has a concept name at each node, i.e., OBJECT, PART, MATERIALS, DESIGN, REF. (They will be called *concepts* here, although in GOS itself they are referred to by the less suggestive name of codes.) In the example the concepts repeat. Thus PART, MATERIALS, DESIGN and REF all occur more than once. Furthermore REF repeats in more than one way: because it comes under PART which itself repeats; because it comes both under MATERIALS and under DESIGN; and because it repeats under MATERIALS and under DESIGN. Generally, in GOS any concept, so long as it can appear at some point in the structure once, may repeat, and may repeat indefinitely often.

Another general feature of GOS is that no upper limits are imposed on the size of fields, or on the total number of different fields in a record, or on the total number of concepts that a record may utilize, and so on. Of course there will be upper limits in practice, dictated for example by limitations of hardware. Thus a GOS record must fit into the available computer store in order to be readable, so this gives one upper limit to the size of a record. In fact on the IBM 370/165 at Cambridge, the implementation of GOS is such that a record must be less than 64K bytes in size, which seems large enough for all practical purposes. The important point here however is that the upper limits are dictated by what is a practicable implementation

of GOS, and not as the result of a desire to take short cuts in writing the software.

GOS records are collected together into *files*, and a collection of records in a file will usually have their structure described by some *format*. The use of formats is important in GOS, although we will not go into any details about them here. Essentially formats define record structure, and a format in GOS corresponds to a database definition in COBOL, or a database description in IMS, or a relational definition (or whatever it may be called) in some relational database.

This then is the data which GOS handles, and the question of course remains, what does GOS do with it? It should perhaps be said at once that GOS is not a Data Base Management System (DBMS) or an Information Retrieval (IR) System, in the strict sense in which these terms are nowadays accepted. It is really a system which enables the user to establish catalogues in machine readable form, and to derive from them catalogues which he himself can read, on a variety of different output media, in a variety of different styles, and in particular sorted in a number of different ways. Within GOS a number of individual programs are provided for handling data. These programs are called *processors*, and in fact their total number is quite small. The real power of GOS comes with the ability of these processors to make use of a collection of other utilities, called *systems*, which the processors may enter rather as a main program can call the members of a subroutine library.

The main processors in GOS are as follows: There is a processor BUILD which constructs GOS records from a text form of the records, which will typically have been produced by data preparation staff. There is a processor DISPLAY (more exactly two processors called DISP and PAG) which prints records out. The processor KEY generates keys in records for sorting, and there is a SORT/MERGE capability to process these records. The processor RET retrieves records from a file which satisfy some retrieval criterion. (This works by simple sequential scanning down a file.) The processor COMBINE 'merges' two files of GOS records together to form a new one. This provides a very simple and clean way of updating records. And finally there is a processor EDIT which provides a general purpose editor for manipulating GOS files.

To explain the use of systems, we will introduce a simple example. Suppose that we have a GOS record, and in one of its fields is a piece of text. We wish to put the first character in the field into upper case if it is a letter, and the remaining characters, when they are letters, into lower case:

JAMES	→	James
john	→	John

We will also take it as understood that this is the kind of activity that we are likely to want to do in GOS. The first important point is that we do not turn operations of this kind into inbuilt primitive operations in the package. We would regard them as being at too high a level for that. Instead they are constructed out of simpler primitives. One system in GOS is called the ANEL system, and it enables us to analyse string elements. While ANEL is in use, a cursor points to a character in the string, initially the first:

'jOhN'
↑

The primitive command C forces the character at the cursor into upper case (C

stands for 'capital') and moves the cursor right by one:

after C: 'JOhN'
↑

S forces the character at the cursor into lower case (S stands for 'small') and moves the cursor right by one:

after C S: 'JohN'
↑

So to force the first character to upper case and the remainder to lower case we need to do

C S S S . . .

We cannot of course say how many S commands we will need, since this will depend on the length of the string, but we can force S to be repeated until the cursor falls off the end of the string by writing:

C REPEAT(S)

We can indicate that this belongs to the ANEL system by writing:

+ ANEL(C REPEAT(S))

— and this is in fact how in GOS a command is written to force a string into the form of a capital followed by small letters. Any structure belonging to the ANEL system is called an anel.

Now we may wish to call an anel up from any part of the package. With the example anel given above, we might want to use it as a tidying up process applicable to various fields when they are first entered to the system, with the intention that the people doing the data preparation may ignore distinctions of case in certain fields, and the use of upper and lower case is then sorted out by the appropriate anel. Thus we could want to use the anel in BUILD. Again it might be felt appropriate to keep certain fields in the machine in just one case (upper case only say), and then on certain occasions to print them in the form capital followed by smalls for cosmetic purposes. Thus we might want to use the anel in DISPLAY. Or we might force words into the form capital followed by smalls when generating keys for sorting, this form of keyword being obviously very suitable with certain indexes, and so we might want to use the anel in KEY. And of course we would like to have the anel available in EDIT, since this is a general purpose program from which we want all facilities to be available.

The essential point is that we want the capability provided by the anel to be available to us at all points in the system, and also we want it to be made available in the same way. There are in fact a large number of systems in GOS (over 35), and they perform a wide variety of tasks from doing real and integer arithmetic to data scrambling for security purposes and mapping out the store utilization in the machine. Two systems stand out as particularly important. One is the ANEL system which we have just mentioned, and which provides a powerful string processing

capability. The other is the SCAN system, which effectively supplies the mechanism for moving around a GOS record so as to look at and operate on its different fields.

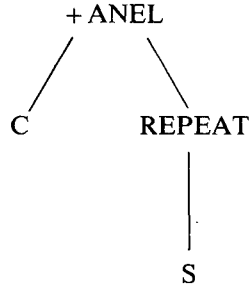
In the example, the anel only employed three components, C, REPEAT and S, with a correspondingly simple effect. But it is possible to write quite complex anels, and their power is greatly enhanced by the ability to switch from the anel system into any other system. Thus it is possible to write anels to analyse taxonomic names, grid references, latitude/longitude measurements, and to convert roman numbers to arabic and arabic numbers to roman.

Each of the GOS processors runs under the control of a processor specification, or spec. The spec effectively gives a list of options, or steering lines, which define the way in which the processor is to operate. Within the spec there may be various calls to other systems, so that for example

. . . +ANEL(C REPEAT(S)) . . .

could be part of a spec which calls up the ANEL system. In GOS the processors can be driven by specs which run to several pages in length, so that it is possible for the operation of a particular processor to be made very elaborate.

Now comes a significant point: as far as GOS is concerned, the specs themselves are just a series of GOS records. For example, the anel given above corresponds to a record with very roughly this structure:



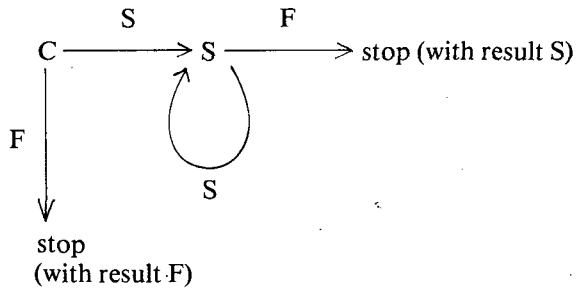
Here, C, REPEAT and S are the concept names; C and S are at leaf nodes in the record, and the REPEAT and +ANEL indicate higher groupings. Effectively what has happened is that the very general record structure offered by GOS has been used not only to accommodate the various data records handled by GOS, but also the various GOS control structures as well. The advantages of doing this are various. It cuts down the amount of code in the package, since the same basic procedures will handle all the different data forms within it. This is an enormous advantage. The smaller the code, the easier it is to debug, maintain, document and transport. In fact, in view of the many facilities it offers, GOS is surprisingly small, and can be implemented on minicomputers fairly comfortably. Another advantage is that the user of the package has much less to learn, since what he knows about GOS records for catalogue data serves equally well when he comes to learn about writing the control structures. This is again important, since the GOS manual is a sizeable object, and the initial learning effort is inevitably quite high. A third advantage is that data from GOS catalogue records can be mixed in with the control structures in a natural way. Thus an instruction to insert a piece of data at a particular point in a GOS record can be represented by making the piece of data to be inserted a part of

the GOS command to do the insert, the whole command being a simple GOS record, or part of a record.

We have said earlier that processor specs can often be quite substantial in size, although they do not of course need to be. Often, writing a processor spec requires the care and skills that go with writing a computer program, and many of the structures for driving GOS exhibit programming language characteristics. So for example the anel structures, which give string processing capabilities, are in many ways analogous to patterns in SNOBOL4 (Griswold *et al.*, 1978). The organization of the flow of program control through the anels however makes them much closer relatives of the expressions in Icon (Griswold *et al.*, 1979), the more recent language designed by Griswold and others. GOS makes use of implicit *signals* S and F (success and failure) to communicate information which determines what to do next. For example, commands S and REPEAT in the anel system yield signals S or F as follows:

> S yields signal S if the cursor is not at the end of the string
> or F if the cursor is at the end of the string
>
> REPEAT (e), where e is some expression, causes e to be repeated until e yields F, and then REPEAT (e) itself yields S.

So the flow of control of the anel expression C REPEAT(S) can be represented as follows:



Note that the expression as a whole gives either result F (when the expression C gives F, which happens with a null string) or result S. This distinction can be utilized at an outer level if necessary. In GOS this use of signals is not confined only to the anel system, but in fact extends over the entire package, and it has proved to be extremely effective.

From its first conception, it was intended that GOS should be a portable piece of software, and considerable effort went into trying to minimize the amount of work that would need to be done in implementing it on a new computer. The choice of language, BCPL (Richards and Whitby Strevens, 1979), was in part motivated by the desire for portability. The language itself is highly portable between machines, a feature which is in fact one of its principal design aims. Apart from this, BCPL will generate very good object code, and is very suitable for non-numerical applications. This made it an appropriate programming language for our purposes. Initially we found considerable resistance to the idea of using BCPL, which came as something of a surprise. Many people suggested it was inadequately supported, and compared

with such old favourites as COBOL and FORTRAN, quite esoteric. We rejected these ideas, and I am sure we were wise to do so. BCPL is gradually gaining respectability, particularly with its ready availability now on mini- and micro-computers, and the portability of GOS has been demonstrated in practice.

```
A.3312. Cephalaspis (Eucephalaspis) agassizi Lamarck. Middle Cornstones, Lower.
  Old Red Sandstone; Kent Church Hill, Hereford. Pres. Rev. W. Symonds.
  Fig'd Stensio, 1932, Cephalaspids Gt. Britain, Brit. Mus. Nat. Hist., p.160, Pl.50,
  fig.1, as Securiaspis kitchini Stensio, Paratype.
```

FIG. 1

A high level of portability in GOS is achieved as follows: the machine dependencies of the package are concentrated in just four of the eighty odd modules. Within these modules certain primitive GOS procedures need to be defined in terms of the local BCPL library procedures, suitably enhanced if necessary. One therefore has to interface the GOS view of the world with the view of the world suggested by the local runtime system for BCPL. A clear definition of these primitive GOS procedures is supplied in the form of program comments, together with hints on how they might be implemented when they are a little tricky. The distribution tape for GOS contains, as well as the code for the package, a description of the steps to be taken to set the package up at the new installation, and a substantial test, which can be replicated to make sure the package is working correctly.

As well as the implementation on the IBM 370/165 at Cambridge, GOS has been implemented on five other machines, namely a Computer Automation LSI-4/30 minicomputer running under the TRIPOS operating system, also at the Computer Laboratory in Cambridge; the IBM 370/168 running under MTS at Newcastle University; the CDC 7600 at the University of Manchester Regional Computer Centre; a Hewlett-Packard 21 MX at the Research Laboratory of the British Museum; and a Cromenco Z-80 based micro at the National Maritime Museum, Greenwich.

It will be seen that there have already been three implementations of GOS on mini- or microcomputers, and in view of the direction that hardware developments are currently taking, the ability to put GOS on machines in this range would seem to

```
*key A.3312 *gcode PIS *store d.c.l
*re*tax +Cephalaspis (+Eucephalaspis) +agassizi Lamarck
*ss*rk Middle Cornstones *comp Lower Old Red Sandstone *loc
Kent Church Hill * Hereford
*oh presented *pers Symonds, W. <Rev.
*re fig'd *a Stensio *d 1932 *t Cephalaspids Gt. Britain
Brit. Mus. (Nat. Hist.) *pp p.160 pl. 50 fig. 1 *tax +Securiaspis
+kitchini Stensio *f paratype
*st fig'd * type
#
```

FIG. 2

be a distinct advantage. The problem of running GOS on a small computer is not caused by the size of the code, so much as by the potential size of the data one may wish to handle. This can be problematical in two ways: either the specs to drive the package may become too large as the user tries to perform increasingly elaborate

operations on the data, or the records of data themselves may overflow the capacity of the machine. Obviously the way around this problem is to keep the applications

```
*gi
  *id
    *key
      *code
        *elem A
        *elem=3312
        *elem=Ø
  *gcode PIS
  *store
    *storel d.c.l
  *st fig'd
  *st type
  *oh
    *f
      *fl presented
    *pers
      *psl Symonds, W.
      *psd Rev.
  *ss
    *geogr
      *loc
        *locl Kent Church Hill
      *loc
        *locl Hereford
    *strat
      *comp
        *compl Old Red Sandstone
        *compd lower
      *rk
        *rkl Cornstones
        *rkd middle
  *re
    *taxon
      *tax
        *taxs \UCephalaspis\N (\UEucephalaspis\N) \Uagassizi\N Lamarck
        *gen Cephalaspis
        *sgen Eucephalaspis
        *spec agassizi
  *re
    *f
      *fl fig'd
    *f
      *fl paratype
    *taxon
      *tax
        *taxs \USecuriaspis kitchini\N Stensio
        *gen Securiaspis
        *spec kitchini
    *ref
      *doc
        *a
          *al Stensio
        *d
          *ryear 1932
        *t Cephalaspids Gt. Britain Brit. Mus. (Nat. Hist.)
      *pp p.16Ø pl. 5Ø fig. 1
#
```

FIG. 3

of GOS simple, and it is not as yet clear just how much complexity these implementations will really allow.

The practical use of GOS is well illustrated by the project to computerize the catalogue of the Sedgwick Museum in Cambridge. Figure 1 shows the record for specimen number A.3312 in the old Sedgwick Museum manual catalogue. A.3312 is in fact a fossil fish, originally identified as *Cephalaspis* (*Eucephalaspis*) *agassizi* Lamarck. It was found at Kent Church Hill, Hereford, and came from a band of rock known as the Middle Cornstones, within the lower part of a larger band, the Old Red Sandstone, and was presented to the museum by the Rev. W. Symonds. Later it was used by Stensio in his 1932 publication *Cephalaspids of Great Britain* as a *paratype* in the definition of the new taxon *Securiaspis kitchini* Stensio. It was also *figured* (i.e., illustrated) in the same work. The figure is in Plate 50, Figure 1. A fossil so used acquires a particular status in the Sedgwick Museum collection, and is given the status values *type* and *fig'd.*

Figure 2 shows how this record may be typed out for input to the BUILD processor of GOS. The various items of data in Figure 1 have been split up, and prefixed by a suitable tag, *key, *rk, *pers and so on. A few extra items of data have been added in this example: the group code PIS (short for Pisces) indicates that the specimen is a fossil fish, and storage location d.c.1 explains where the specimen can be found in the museum. The status words 'fig'd' and 'type' have also been included. A comparison of Figures 1 and 2 shows how closely the data input to BUILD may be made to mirror the form of the data in a manual system. In particular, the input data does not need to reflect the elaborate structure which the GOS record for A.3312 will have when it is established in machine readable form, although a certain sense of this structure needs to be grasped in typing the record, in order that the correct tags may be chosen. A certain amount of thoughtful reorganization during typing will of course be necessary. Thus names must be typed with surname first followed by a comma and forenames or initials, and with titles preceded by the character '<'. This is as an aid to subsequent sorting and displaying. Again significant components of taxonomic names must be indicated by a plus sign.

The structure of the GOS record generated by BUILD from this input is given in Figure 3. Each line represents either a single field of data or a grouping of such fields, and in the former case the data tag (or concept name) is given on the left, followed on the right by the actual data. Indentation to the right indicates group membership, so that the *oh (=ownership history) group on line 13 contains two data groups indicated by *f and *pers, and the *pers group contains two data fields, a main keyword part in *psl, and a detail part in *psd. It will be seen that the textual data of Figure 2 has undergone some readjustment. Thus the *pers data has been split into two fields at the character '<', the components of the taxonomic names indicated by the plus signs have been put into special fields indicated by *gen, *sgen and *spec (i.e., genus, subgenus and species), and the words 'middle' and 'lower' in the rock names have been relegated to special fields. BUILD does this work by interpreting various anels specially designed for the Sedgwick Museum application of GOS.

From collections of records structured in this way catalogue listings and indexes may be generated. The total number of records for the Pisces group in the Sedgwick Museum is just over 4000, and Figure 4 shows the first page of the catalogue for this collection. A.3312 can be seen at the top. The considerable variation in data over the records is at once apparent — note in particular the size of record A.3466. The particular layout here is designed for microfiche output, but Figure 4 was in fact generated for this paper at A4 size by outputting it on a Diablo printer and then putting the result through a reducing photocopier. Practically any kind of layout is

possible, and a facility now exists in the Sedgwick Museum for generating specimen labels. The records are printed in a rectangular array, possibly reduced in size with a reducing photocopier, and then sliced up on a guillotine and fitted into the bottom

SedgMus PIS Main Cat 1981 page 1

A.3312 fig'd; type. (PIS) d.c.1
Presented, Symonds, W. (Rev.).
Lower Old Red Sandstone. Middle Cornstones; Kent Church Hill, Hereford.
Cephalaspis (Eucephalaspis) agassizi Lamarck.
Fig'd, paratype, Stensio, 1932, Cephalaspids Gt. Britain Brit. Mus. (Nat. Hist.), p.160 pl. 50 fig. 1, as Securiaspis kitchini Stensio.

A.3313 fig'd. (PIS) d.c
Presented, Browne, G.F. (Mrs.).
Old Red Sandstone; Reswallie, Forfar, Scotland;
Lower Old Red Sandstone.
Cephalaspis lyelli Agassiz.
Fig'd, Stensio, 1932, Cephalaspids Gt. Britain Brit. Mus. (Nat. Hist.), p.99 pl. xxxvii fig. 3, as Cephalaspis pagei Lankester.

A.3314 (PIS) d.c.1
Cookson Coll.
Old Red Sandstone; Herefordshire.
Cephalaspis lyelli Agassiz.
Recorded, Stensio, 1932, Cephalaspids Gt. Britain Brit. Mus. (Nat. Hist.), p.94, as Cephalaspis whitei Stensio.

A.3327 (PIS) d.c.1
Old Red Sandstone; Herefordshire.
Recorded, Stensio, 1932, Cephalaspids Gt. Britain, Brit. Mus. Nat. Hist., p.94, as Cephalaspis whitei Stensio.

A.3466 type; fig'd. (PIS) xxviii.s.64
(Schists), Devonian; Lantivit Bay, Cornwall;
Darmouth Slates, Siegenian.
Described, McCoy, 1851, Ann. Mag. Nat. Hist., (2) viii p.483, McCoy, 1854, Contrib. Brit. Palaeont., p.234, as Zoophyte Steganodictyum carteri M'Coy.
Fig'd, explained, McCoy, 1955, Brit. Pal. Foss. Geol. Mus. Cambridge, Pl. II A figs.4, 4 a, as Amorphozoa Steganodictyum carteri M'Coy.
Described, Lankester, 1868, Q.J.G.S., xxiv, pp.546-7, (from McCoy's figures) as Fish (allied to the genus Cephalaspis).
Mentioned, Hinde, 1888, Brit. Foss. Sponges, Mon. Pal. Soc., p.182.
Listed, Woods, 1891, Cat. type Foss. Woodw. Mus., p.167, as Pisces Steganodictyum carteri McCoy.
Recorded, Stensio, 1932, Cephalaspids Gt. Britain, Brit. Mus. Nat. Hist., p.179, as Cephalaspis? carteri (McCoy).
Identified, White, E.I., as Arthrodire Drepanaspis (a much distorted fragment of either an arthrodire or Drepanaspis).
Mentioned, Tarlo, 1961, Acta Pal. Polonica, vi, p.370, as (within the genus Drepanaspis).
Holotype, fig'd, Tarlo, 1965, Palaeontologia Polonica, No.15 p.36 pl. viii fig.1, as Drepanaspis carteri (McCoy).

A.6006 fig'd; type. (PIS) xxviii.b.1
Presented, Pollexfen, J.H. (Rev.).
Old Red Sandstone; Orkney;
Old Red Sandstone, Black Devonian Flags; Orkney;

FIG. 4

of the specimen boxes. Figure 9 shows an example of output in this style. Figure 5 shows a page from the taxonomic index to the Pisces, and the entry for A.3312 can be found near the bottom of the page. There will be a second entry for this specimen under the earlier taxonomic name. Figure 6 shows a page from the index for bibliographic references. The entry for the Stensio reference continues from the previous page, and A.3312 appears at the top. (The misplacing below on the left of 'p. 116 pl. iii figs. 7–8' is caused by a data error.) Figure 7 shows a page from the locality index with the entry for 'Hereford, Kent Church Hill'. Similarly indexes for stratigraphy, donors of specimens and museum storage locations may be generated. In GOS almost any kind of index from the basic data can be generated, and Figure 8 shows a page from an index to the headings used in the locality index. This tells us for example that 'Kent Church Hill' occurs both under 'Hereford' and 'Herefordshire' and this index acts as an essential aid in effective use of the locality index.

```
SedgMus PIS Tax ind  1981 page 183
- - - - - - - - - - - - - - - - -

                Fig'd, [syntype], Woodward, A.S., 1889, Proc. Geol. Assoc., XI, p.300 pl. III
                  fig.33, dentition (lower - right).                          J.5822 xvii.t.b fig'd; type.

(PIS) Scaphodus heteromorphus Woodward
        Stonesfield Slate,
           Oxfordshire
                Listed, Woods, 1891, Cat. Type Foss. Woodw. Mus., p.166.      J.5822 xvii.t.b fig'd; type.

(PIS) Sciaenurus bowerbanki Agassiz
        London Clay,
           Essex
              Sparnodus bowerbanki (Agassiz)                                  C.21192 vii.c.3
           Kent
              Sparnodus bowerbanki (Agassiz)                                  C.21193-21195 vii.c.3

(PIS) Scombramphodon crassidens A. S. Woodward
        London Clay,
           Kent                                                               C.21196 vii.c.3

(PIS) Scombrinus macropomus (Agassiz)
        London Clay,
           Kent                                                               C.21198-21199 vii.c.3

(PIS) Scyllium angustum (Agassiz ex Münster ms)
        Senonian,
           Baumberge                                                          F.11527 ix.t.f fig'd.

(PIS) Scyllium angustum (Agassiz)
        Senonian,
           Baumberge
              Scyllium angustum (Agassiz ex Münster ms)
                 Mentioned, Woodward, A.S., 1889, Cat. Foss. Fish Brit. Mus. (Nat. Hist.), part i,
                   p.340.                                                     F.11527 ix.t.f fig'd.

(PIS) Scyllium minutissimum (Winkler)
        Barton Beds,
           Hampshire
                Tooth.                                                        C.14069-14071 viii.t.a
                                                                              C.60189-60193 viii.t.122

(PIS) Securiaspis kitchini Stensio
        Middle Cornstones,
           Hereford
                Fig'd, paratype, Stensio, 1932, Cephalaspids Gt. Britain Brit. Mus. (Nat. Hist.),
                  p.160 pl. 50 fig. 1.                                        A.3312 d.c.1 fig'd; type.

(PIS) Seriola prisca (Agassiz)
        Upper Eocene,
           Italy                                                              C.31438-31439 vii.d.4
                                                                              C.31449

(PIS) Serramus occipitalis Agassiz
        Upper Eocene,
           Italy
              Sparnodus microstomus (Agassiz)                                 C.31425 vii.c.4
```

FIG. 5

SedgMus PIS Doc ind 1981 page 30

[Stensio, 1932, Cephalaspids Gt. Britain Brit. Mus. (Nat. Hist.).]		
(PIS) Securiaspis kitchini Stensio	fig'd, paratype, p.160 pl. 50 fig. 1.	A.3312 d.c.1 fig'd; type.
Tarlo, 1961, Acta Pal. Polonica, vi.		
(PIS)	identified, White, E.I., mentioned, p.370.	A.3466 xxviii.s.64 type; fig'd.
Tarlo, 1961, Acta Palaeontologica Polonica, vi.		
(PIS) Rhinopteraspis cornubica McCoy	identified (as original of McCoy's fig. 3), chosen, lectotype, refig'd, part a p.368 pl. 1.	A.6955 xxviii.s.64 fig'd; type.
Tarlo, 1964, Palaeontologia Polonica, 13 p.116 pl.iii figs.7-8.		
(PIS) Pycnosteus obruchevi Tarlo	identified, Dineley, fig'd, holotype.	H.5170 xxvii.t.1 fig'd; type.
Tarlo, 1965, Palaeontologia Polonica.		
(PIS) Drepanaspis carteri (McCoy)	identified, White, E.I., holotype, fig'd, No.15 p.36 pl. viii fig.1.	A.3466 xxviii.s.64 type; fig'd.
Tarlo, 1965, Palaeontologia Polonica, 15 p.78 text-fig. 20A.		
(PIS) Pycnosteus obruchevi Tarlo	identified, Dineley, fig'd, (number misquoted as H.1,570).	H.5170 xxvii.t.1 fig'd; type.
Tawney, 1882, Proc. Camb. Phil. Soc., iv.		
(PIS) Lamna elegans Agassiz	listed, p.150.	C.19990 viii.t.127
(PIS) Myliobates toleapicus Agassiz	listed, p.150.	C.19991 viii.t.127
(PIS) Otodus obliquus Agassiz	listed, p.150.	C.19989 viii.t.127
Tawney, E.B., label.		
(PIS) Ceratodus n.sp.		J.58317 xix.a.5
(PIS) Ceratodus sp.		J.58351-58352 xx.1142
(PIS) Ceratodus sp.n.		J.58314 xix.a.5
		J.58319 xix.a.5
		J.58335 xx.1142
		J.58339 xx.1142
		J.58341-58342 xx.1142
(PIS) Ceratodus sp.nov.		J.58329 xx.1141
		J.58371 xx.1142
(PIS) Ceratodus var.n.		J.58315 xix.a.5
(PIS) Ceratodus altus Agassiz		J.58313 xx.t.a
		J.58325 xix.a.5
		J.58332 xx.1141
		J.58334 xx.1142
		J.58336 xx.1142
		J.58347-58348 xx.1142
(PIS) Ceratodus sp.n.,var. of altus Agassiz		J.58340 xx.1142
(PIS) Ceratodus curvus Agassiz?		J.58333 xx.1141
(PIS) Ceratodus sp.aff. curvus Agassiz		J.58350 xx.1142
(PIS) Ceratodus daedaleus Agassiz		J.58354 xx.1142

FIG. 6

SedgMus PIS Loc ind 1981 page 50

[Hampshire, Shawford.]
(PIS) Scapanorhynchus subulatus (Agassiz). (Southampton Waterworks new pit), loc: 1086. B.65564-65565 x.596

Hampshire, Stubbington.
Bracklesham Beds,
(PIS) C.69603-69604 vii.t.8
(PIS) Lamna vincenti (Winkler). C.21357-21359 viii.t.h
(PIS) Myliobatis sp. C.70787-70791 vii.t.8
(PIS) Odontaspis crassidens (Agassiz). C.21317 viii.t.h
(PIS) Odontaspis crassidens (Agassiz)?. C.21318-21319 viii.t.h
(PIS) Odontaspis cuspidata (Agassiz)?. C.21321 viii.t.h
(PIS) Odontaspis elegans (Agassiz). C.21336-21340 viii.t.h
C.65622 vii.t.8
(PIS) Odontaspis macrodon (Agassiz). C.66404-66406 vii.t.8
(PIS) Raia sp. C.66344 vii.t.8

Hampshire, Winchester.
Upper Chalk,
(PIS) Ptychodus polygyrus L. Agassiz. B.20764 x.599
Turonian,
(PIS) Ctenothrissa sp. (By-pass cutting), grid ref: 41/496287. B.91635 4.19
(PIS) Scapanorhynchus rhaphiodon (Agassiz). (By-pass cutting), grid ref: 41/496287. B.91618 4.19
(PIS) Scapanorhynchus subulatus (Agassiz). (By-pass cutting), grid ref: 41/496287. B.91636 4.19

Hastings, Hollington.
Wealden,
(PIS) Pisces Hybodus sp. B.53755-53761 xiii.t.37

Hastings, Hollington, Silver Hill.
Wealden,
(PIS) B.53922 xiii.t.38

Hereford, Kent Church Hill.
Middle Cornstones,
(PIS) Securiaspis kitchini Stensio. A.3312 d.c.1 fig'd; type.

Herefordshire.
(PIS) Cephalaspis whitei Stensio. A.3314 d.c.1
A.3327 d.c.1
Devonian,
(PIS) SMPIS Pteraspis sp. H.4703 xxviii.c.1

Herefordshire, Aymestry, Mere Hill Wood.
Ludlow,
(PIS) (Quarry at western tip of Mere Hill Wood - 1 1/2 miles W. of Aymestry). A.59157 xxx.n.25

Herefordshire, Cradley.
Devonian,
(PIS) SMPIS Benneviaspis lankesteri Stensiö. H.4691 d.c.1

FIG. 7

SedgMus PIS Loc keys 1981 page 11
- - - - - - - - - - - - - - - - - -

Kent.			
Kent Church Hill,	Hereford.		
	Herefordshire.		
Keston,	Bromley,	Kent.	
Khaneh Kot,	Fars,	Persia.	
Kildonan,	Eigg.		
Kilkenny,	Ireland.		
Kilkenny Co.,	Ireland.		
Kimeridge,	Dorset.		
King's Quay,	Wootton,	Isle of Wight.	
Kingsthorpe,	Northamptonshire.		
Kington,	Herefordshire.		
Kirkaldy.			
Kirtlington,	Oxfordshire.		
Kressenberg,	Bavaria.		
La Crèche,	Boulogne,	France.	
La Presta,	Val de Travers,	Switzerland.	
Ladybank.			
Ladysmith,	Natal,	South Africa.	
Lagus,	Saucats,	Gironde,	France.
Lanarkshire.			
Lanarkshire,	Scotland.		
Lancashire.			
Lancashire.			
Langenattheim.			
Langholm,	Dumfries,	Scotland.	
Langholm Dumfries.			
Langton Bridge,	Chipping Norton,	Oxfordshire.	
Langton Matravers,	Isle of Purbeck,	Dorset.	
	Swanage,	Dorset.	
Lantic Bay,	Lantwit Bay,	Polperro,	Cornwall.
Lantivet Bay,	Polperro,	Cornwall.	
Lantivit Bay,	Cornwall.		
Lantwit Bay,	Polperro,	Cornwall.	
Laramie,	Wyoming,	U.S.A.	
Laycock,	Wiltshire.		
le Cheminot,	Cresantignes,	Aube,	France.
Le Wast,	Boulonnais,	France.	
Leckhampton,	Gloucestershire.		
Ledbury,	Herefordshire.		
Ledbury Tunnel,	Herefordshire.		
Leeds,	Yorkshire.		
Leicestershire.			
Leigh.			
Leighton Buzzard,	Bedfordshire.		
Leintwardine.			
Leintwardine,	Herefordshire.		
Leith,	Edinburgh,	Scotland.	
	Edinburghshire,	Scotland.	
Leognan,	Gironde,	France.	
Les Pichottes,	Le Wast,	Boulonnais,	France.
Lesmahagow,	Lanarkshire.		
Lethen.			
Lethen,	Nairnshire.		
Lethen Bar.			

FIG. 8

J.69027-69028 xx.t.37
Cambridge, P. Coll., 1981, ref: V341.
Lower Lias; Blockley Station Pit, Blockley Village (1.5 miles N.E.), Worcestershire, grid ref: NGR SP 181373.
Identified, Cox, L.R., as Modiolus scalprum (J. Sowerby).

J.69029 xx.t.18
Cambridge, P. Coll., 1981, ref: V273.
Lower Lias; Blockley Station Pit, Blockley Village (1.5 miles N.E.), Worcestershire, grid ref: NGR SP 181373.
Identified, Cox, L.R., as Ostrea (Liostrea) irregularis (Münster).

J.69030 xx.t.18
Cambridge, P. Coll., 1981, ref: V108.
Lower Lias; Honeybourne Brickpit, Worcestershire, grid ref: NGR SP 11584517, loc: Exposure 1.
Identified, Cox, L.R., as Ostrea (Liostrea) irregularais (Münster).

J.69031 xx.t.35
Cambridge, P. Coll., 1981, ref: V1840.
Lower Lias; Honeybourne (Battle headquarters), Worcestershire.
Hippopodium ponderosum (J. Sowerby).

J.69032 xx.t.37
Cambridge, P. Coll., 1981, ref: V341.
Lower Lias; Aston Magna Brickpit, Worcestershire, grid ref: NGR SP 2036.
Modiolus scalprum (J. Sowerby).

J.69033 xx.t.35
Cambridge, P. Coll., 1981, ref: V1840.
Lower Lias; Aston Magna Brickpit, Worcestershire, grid ref: NGR SP 2036.
Hippopodium ponderosum (J. Sowerby).

J.69034-69035 xvii.b.10
Cambridge, P. Coll., 1981, ref: V253.
Lower Lias; Bretfarton (ditch near), Vale of Evesham, Worcestershire, grid ref: NGR SP 083439.
Identified, Howarth, M.K., as Ichthyosaurus sp. vertebra.

J.69036 xx.t.18
Cambridge, P. Coll., 1981, ref: V435.
Lower Lias; Campden Railway Tunnel, Chipping Campden (near), Gloucestershire, grid ref: NGR SP 165404.
Identified, Cox, L.R., as Ostrea (Liostrea) irregularis (Münster).

J.69037 xvii.b.10
Cambridge, P. Coll., 1981, ref: V253.
Lower Lias; Kemerton (near), Worcestershire, grid ref: NGR SO 9437.
Ichthyosaurus sp. humerus.

J.69038 xx.t.67
Cambridge, P. Coll., 1981, ref: V1847.
Bucklandi Zone, Lower Lias; Redmile, Grantham (near), Lincolnshire.
Identified, Cambridge, as Arnioceras semicostatum (Young and Bird).

FIG. 9

REFERENCES

Gorman, M. and Winkler, P. W. (1980) *Anglo-American Cataloguing Rules.* Second edition. London: Library Association.

Griswold, R. E., Poage, J. F. and Polonski, I. P. (1978) *The SNOBOL4 Programming Language.* Englewood Cliffs, New Jersey: Prentice-Hall Inc.

Griswold, R. E., Hanson, D. R. and Korb, J. T. (1979) The Icon programming language — an overview. *ACM Sigplan Notices 14* (4) 18–31.

Porter, M. F. (1980a) *GOS Reference Manual.* Duxford, Cambridgeshire: Museum Documentation Association.

Porter, M. F. (1980b) *How to use GOS.* Duxford, Cambridgeshire: Museum Documentation Association.

Porter, M. F. (1980c) *Description of the internal structure of GOS.* Duxford, Cambridgeshire: Museum Documentation Association.

Porter, M. F., Light, R. B. and Roberts, D. A. (1977) *A unified approach to the computerization of museum catalogues.* London: British Library (British Library Research and Development Reports. Report no. 5338 HC).

Richards, M. and Whitby Strevens, C. (1979) *BCPL — the Language and its Compiler.* Cambridge University Press.

Roberts, D. A. and Light, R. B. (1980) Progress in documentation: museum documentation. *Journal of Documentation 36* (1), 42–84.

UK Marc Manual (1980) Second edition. London: British Library.