Automatic Report Formatting from a Report Specimen

J. R. Lin Sin Cho
Ganymede Division of SIA, 23 Lower Belgrave Street, London SW1, UK

The need to allow unskilled users to produce reports from computer files on their own has long been recognized. Simple Report Generators usually produce a tabular layout with columns of information surmounted by a heading. There is often a need to produce reports with a more complex layout with data and headings in a non-tabular format. The aim of the present Report Formatter is to allow an unskilled user to produce a complex report by demonstrating the appearance of the report rather than detailing the procedures necessary to produce the report.

INTRODUCTION

The need to allow unskilled users to produce reports from computer files on their own has long been recognized. Simple Report Generators usually produce a tabular layout with columns of information surmounted by a heading. There is often a need to produce reports with a more complex layout with data and headings in a non-tabular format. Although the more complex Report Writers have this facility they are difficult for the unskilled user to use. Many Report Writers simply provide a language with similar capabilities as a record-oriented language like COBOL or PL/1, the difference being in the way the language is presented to the user. A Report Program is presented as a group of named paragraphs or sections which are used to identify the actions to be taken when certain conditions are met. Typical examples are AT RECORD DO (Actions) and AT END DO (Actions) which specify the actions to be taken when a record has been read and when the end of file is encountered respectively. The actions, however, still have to be specified in a procedural way.

To the experienced programmer, Report Writers in this class do not offer significant advantages over the use of a high level language. However, the Report Writer may form an integral part of the Data (Base) Management System and the user may be able to write his own program to produce a report because the data may be stored in a format that only the Report Writer and associated modules can read. COBOL Report Writer, on the other hand, provides a facility for specifying the physical appearance of a report with a subsequent reduction in the amount of coding at the procedural level. Nevertheless, considerable skill is still required to write this report specification.

The aim of the Report Formatter discussed in this paper is to allow an unskilled user to produce a complex report by demonstrating the appearance of the report rather than detailing the procedures necessary to produce the report. The term 'Report Formatter' has been used in preference to 'Report Writer' as the selection of records is done outside the Report Formatter. To enhance ease of use of the system, a facility which allows the user to see a sample of the report before the generation of the complete report has also been implemented. This Report Formatter has been implemented in Titus, a Data Management System developed by the author over the past few years and used extensively at SIA Computer Services. The work described in this paper was carried out while the author was a registered PhD student at Birkbeck College, London.

A SHORT EXAMPLE

Consider a small personnel file as shown in Table 1 (Field names are the column headings). Suppose that the personnel manager wishes to produce a report of all employees in every department. The first thing he would do is to draw on a sheet of paper an example of the report to show clearly the contents and physical appearance of the report. He would choose suitable values (he will not use numbers when they represent names of employees for example) and he would indicate that there could be many departments and many employees in a given department. Table 2 shows an example of a Report Specimen as might be produced by the personnel manager. Then he would see a programmer and explain to him what all the example-values on the Report Specimen represent. For example JOHN, JIM, BROWN and BARRY are the names of employees. With most Report Writers, the programmer now has to write a program. However, with Titus, the programmer only has to store the example into a file (called EXPL in our example), and could immediately produce the report as shown in Table 3. Table 4 gives a brief explanation of the run. The declarations of the example-values are self-explanatory but the functions $CNT(EMPNO)$ and $SUM(SALARY)$ deserve some explanation at this point. $SUM(X)$ returns the sum of $X$ where $X$ is the name of a numeric field. Similarly the function $CNT(X)$ returns the count of number of occurrences of the field $X$.

The advantage of the Titus Report Formatter is that a Report Specimen could be used to produce a report, thus eliminating the need to write a program. The report itself is shown in Table 5. The Report Formatter uses the file description, the Report Specimen and the example-value declarations to create an internal Report Program. This internal Report Program (Table 6) is then executed using the data file and the file description. The syntax and semantics of the internal report language is given in the appendix.
AUTOMATIC REPORT FORMATTING FROM A REPORT SPECIMEN

FURTHER EXAMPLES

A variation of the above example which only produces a summary of salary totals is shown in Tables 7(a) and 7(b). Note that 'Break' items (See Appendix) are handled automatically by the system. An example of a report where a section is printed across the page is shown in Tables 8(a) and 8(b). The command USE 65 COLUMNS controls the maximum number of names that should be printed on each line. The example in Tables 9(a) and 9(b) shows Titus Report Formatter being used to write a letter to every employee notifying them of a 12.5% increase in their salary.

Table 1. A personnel file

<table>
<thead>
<tr>
<th>EMPLOYEE NO</th>
<th>EMPLOYEE NAME</th>
<th>JOB TITLE</th>
<th>SALARY</th>
<th>DEPARTMENT</th>
<th>ACCOUNTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>03010</td>
<td>John</td>
<td>Accountant</td>
<td>1000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Jane</td>
<td>Accountant</td>
<td>1000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Mike</td>
<td>Accountant</td>
<td>1000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>John</td>
<td>Clerk</td>
<td>6000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Jane</td>
<td>Clerk</td>
<td>6000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Mike</td>
<td>Clerk</td>
<td>6000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>John</td>
<td>Manager</td>
<td>1000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Jane</td>
<td>Manager</td>
<td>1000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Mike</td>
<td>Manager</td>
<td>1000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>John</td>
<td>Assistant</td>
<td>5000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Jane</td>
<td>Assistant</td>
<td>5000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Mike</td>
<td>Assistant</td>
<td>5000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>John</td>
<td>Secretary</td>
<td>3000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Jane</td>
<td>Secretary</td>
<td>3000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>Mike</td>
<td>Secretary</td>
<td>3000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>03010</td>
<td>John</td>
<td>Secretary</td>
<td>3000</td>
<td></td>
<td>ACCOUNTS</td>
</tr>
</tbody>
</table>

Table 2. An example of a Report Specimen

***** PERSONNEL REPORT *****
DEPARTMENT : ACCOUNTS

EMPLOYEE NO     NAME            JOB TITLE   SALARY
03010          John            Accountant 1000
03010          Jane            Accountant 1000
03010          Mike            Accountant 1000
03010          John            Clerk      6000
03010          Jane            Clerk      6000
03010          Mike            Clerk      6000
03010          John            Manager   1000
03010          Jane            Manager   1000
03010          Mike            Manager   1000
03010          John            Assistant 5000
03010          Jane            Assistant 5000
03010          Mike            Assistant 5000
03010          John            Secretary 3000
03010          Jane            Secretary 3000
03010          Mike            Secretary 3000

TOTAL SALARY = 29000

Table 3. A report produced from a file with Titus

BEGIN TITUS.

TITUS DATA MANAGEMENT SYSTEM LEVEL 1.0
ENTER NAME OF DATABASE DIRECTOR
7 EMPLOYEES
ENTER NEXT COMMAND
7 GENERATE REPORT USING EXPL
DEFINE ALL VARIABLES USED ON EXPL
7 EMPLOYEE TYPE
7 EMPLOYEE NAME
7 JOB TITLE
7 SALARY
7 DEPARTMENT
7 ACCOUNTS
7 NUMBER OF EMPLOYEES
7 TOTAL SALARY
7 END OF REPORT

END REPORT.

Table 4. An explanation of the run

It is assumed that the format of the input file PERSONL has already been defined and sorted on department name before generating the report, otherwise the command

SOFT PERSONL ON (DEPARTMENT)

has to precede the GENERATE command. A good introduction to the Data Management facilities of Titus is given in Ref. 1.

(1) Calling the Data Management System.

(2) User names the database directory which contains a list of all the data files and description files.

(3) Using the GENERATE command to produce the report. PERSONL is the name of the input data file and EXPL is the file containing the Report Specimen. The Example-value declarations may be stored in a file and passed as parameter to this command.

(4) Declaring the Example-values on the Report Specimen.

(5) Entering a carriage return to end declarations.

(6) Entering a carriage return to end page layout instructions. The user may enter HELP at this stage to obtain a list of commands available.

(7) This is a useful facility to see the actual report using a few records of the file only.

Table 5. A report produced from a Report Specimen

* APE B1 *

***** PERSONNEL REPORT *****

EMPLOYEE NO     NAME            JOB TITLE   SALARY
03010          John            Accountant 1000
03010          Jane            Accountant 1000
03010          Mike            Accountant 1000
03010          John            Clerk      6000
03010          Jane            Clerk      6000
03010          Mike            Clerk      6000
03010          John            Manager   1000
03010          Jane            Manager   1000
03010          Mike            Manager   1000
03010          John            Assistant 5000
03010          Jane            Assistant 5000
03010          Mike            Assistant 5000
03010          John            Secretary 3000
03010          Jane            Secretary 3000
03010          Mike            Secretary 3000

TOTAL SALARY = 29000

Table 6. The Report Program

```
01 SECTION 01 $EMT $EM2
02 P1(1) VALUE "***** PERSONNEL REPORT *****"
02 SECTION 02 (DEPARTMENT) $EMT $BM2 $BM1 $BM0
02 P1(1) VALUE "DEPARTMENT"
02 P1(1) VALUE "ACCOUNTS"
02 P1(1) VALUE "TOY"
02 P1(1) VALUE "ELECTRICAL"
03 SECTION 3 BREAK
04 SECTION 4 BREAK
05 P1(1) VALUE "TOTAL SALARY"
05 P1(1) VALUE "ACCOUNTS"
05 P1(1) VALUE "TOY"
05 P1(1) VALUE "ELECTRICAL"
06 P1(1) VALUE "TOTAL SALARY"
07 P1(1) VALUE "TOTAL NO OF EMPLOYEES"
07 P1(1) VALUE "ACCOUNTS"
07 P1(1) VALUE "TOY"
07 P1(1) VALUE "ELECTRICAL"
```

© Heyden & Son Ltd, 1982

THE COMPUTER JOURNAL, VOL. 25, NO. 2, 1982 243

Downloaded from https://academic.oup.com/comjnl/article-abstract/25/2/242/380510 by guest on 03 January 2019
Table 7. (a) Report specimen. (b) Actual run

<table>
<thead>
<tr>
<th>(a)</th>
<th>DEPARTMENT</th>
<th>NO OF EMPLOYEES</th>
<th>TOTAL SALARY</th>
<th>AVERAGE SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TOY</td>
<td>1</td>
<td>4000</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>SHOE</td>
<td>12</td>
<td>20000</td>
<td>1666.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>TOTAL</strong></td>
<td><strong>TOTAL</strong></td>
</tr>
</tbody>
</table>

(b) `begin, titus.
TITUS DATA MANAGEMENT SYSTEM LEVEL 1.C
ENTER NAME OF DATABASE DIRECTORY
? EMPLOYEE
ENTER NEXT COMMAND
? GENERATE PERSONAL USING EXPLA
DEFINE ALL VARIABLES USE ON EXPLA
? TOT,SHOE = DEPARTMENT
? J,H,T = DOC.,NAME
? 1000,2000,3000 = SALARY
? 4000,5000,6000 = SALARY
? CM
ENTER PAGE LAYOUT INSTRUCTIONS
? CR
SO YOU WISH TO SEE A SAMPLE FIRST (REPLY YES/NO)?
? N

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>NO OF EMPLOYEES</th>
<th>TOTAL SALARY</th>
<th>AVERAGE SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTS</td>
<td>3</td>
<td>24000</td>
<td>8000</td>
</tr>
<tr>
<td>ELECTRICAL</td>
<td>2</td>
<td>13000</td>
<td>6500</td>
</tr>
<tr>
<td>TOT</td>
<td>1</td>
<td>3000</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>TOTAL</strong></td>
<td><strong>TOTAL</strong></td>
</tr>
</tbody>
</table>

Table 8. (a) Report Specimen. (b) Actual run
Numbers of employees are dummy values

(a) `begin, titus.
NUMBER OF EMPLOYEES = 3

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>NUMBER OF EMPLOYEES = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOY</td>
<td>JOHN</td>
</tr>
<tr>
<td></td>
<td>BARRY</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEPARTMENT SHOE</td>
</tr>
<tr>
<td></td>
<td>JIM</td>
</tr>
<tr>
<td></td>
<td>JAMES</td>
</tr>
</tbody>
</table>

(b) `begin, titus.
TITUS DATA MANAGEMENT SYSTEM LEVEL 1.C
ENTER NAME OF DATABASE DIRECTORY
? EMPLOYEE
ENTER NEXT COMMAND
? GENERATE PERSONAL USING EXPLA
DEFINE ALL VARIABLES USE ON EXPLA
? J,H,T = DOC.,NAME
? 1000,2000,3000 = SALARY
? 4000,5000,6000 = SALARY
? CM
ENTER PAGE LAYOUT INSTRUCTIONS
? CR
SO YOU WISH TO SEE A SAMPLE FIRST (REPLY YES/NO)?
? N

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>NUMBER OF EMPLOYEES = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTS</td>
<td>PHILIPS</td>
</tr>
<tr>
<td></td>
<td>SMITH</td>
</tr>
<tr>
<td></td>
<td>JONES</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEPARTMENT ELECTRICAL</td>
</tr>
<tr>
<td></td>
<td>WISE</td>
</tr>
<tr>
<td></td>
<td>WILLIAMS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEPARTMENT TOT</td>
</tr>
<tr>
<td></td>
<td>BROWN</td>
</tr>
<tr>
<td></td>
<td>RANDALL</td>
</tr>
<tr>
<td></td>
<td>SIMPSON</td>
</tr>
</tbody>
</table>

Table 9. (a) Report Specimen (b) Actual run

(a) Dear MR BROWN
Thank you for your services over the last year. This letter is to officially inform you that your basic salary will be increased from $7000 to $9000 p.a. effectively as from 01 Jan 80. This represents an increase of $2000 or 28.6 per cent over last year. This letter amends your contract of employment accordingly.

Yours sincerely,
A. Person
Managing Director

(b) Dear MRS JONES
Thank you for your services over the last year. This letter is to officially inform you that your basic salary will be increased from $7700 to $9900 p.a. effectively as from 01 Jan 80. This represents an increase of $2200 or 29.1 per cent over last year.

This letter amends your contract of employment accordingly.

Yours sincerely,
A. Person
Managing Director

Dear MR PHILIPS
Thank you for your services over the last year. This letter is to officially inform you that your basic salary will be increased from $10000 to $12500 p.a. effectively as from 01 Jan 80. This represents an increase of $1250 or 25% per cent over last year.

This letter amends your contract of employment accordingly.

Yours sincerely,
A. Person
Managing Director

Dear MRS SMITH
Thank you for your services over the last year. This letter is to officially inform you that your basic salary will be increased from $9600 to $975000 p.a. effectively as from 01 Jan 80. This represents an increase of $150 or 25% per cent over last year.

This letter amends your contract of employment accordingly.

Yours sincerely,
A. Person
Managing Director

by a Section Footing. For example, an organization may consist of many departments and each department may have many teams and each team may have many members. The formal definition of a Report Specimen written in a BNF-like syntax follows:

```plaintext
<REPORT SPEC> ::= <SECTION>
<SECTION> ::= (HEADING) <SECTION> (FOOTING)(HEADING) <FOOTING>
<HEADING> ::= <Text and/or Expressions not using functions>
<FOOTING> ::= <Text and/or Expressions using functions>
```

DEFINITION OF A REPORT SPECIMEN

The Report Specimen must be in a form whose components or sections can be clearly identified. Informally a report may consist of a Report Heading, a Section and followed by a Report Footing. A Section may in turn consist of a Section Heading, a sub-Section and followed...
where
\[
\text{::=} \quad \text{means is defined as}
\]
\[
\langle A \rangle \quad \text{means that an occurrence of } A \text{ is mandatory}
\]
\[
(A) \quad \text{means that an occurrence of } A \text{ is optional}
\]
\[
\text{indicates an alternative definition}
\]
Text is any character string and functions are $\text{MIN}$, $\text{MAX}$, $\text{AVG}$, $\text{SUM}$, $\text{CNT}$ counters which return the values of minimum, maximum, average, sum and count respectively of a specified field.
Terminal symbols are in lower case characters.

From the above definition it follows that a Section needs to be specified twice (and only twice) to indicate repetition of that Section as shown in Table 2. This forms the basis of the method used. Note that a report which simply consists of the line:

```
SALARY TOTALS = 20000
```

where 20000 is an expression involving a counter variable is also covered by the definition. Both the Heading and the Section are null in this case.

An example where the Section is null but the Heading and the Footing are non-empty is as follows:

```
TOTAL SALARY FOR DEPARTMENT
TOY = 10000
TOTAL SALARY FOR DEPARTMENT
COSMETICS = 15000
```

the Heading consists of ‘TOTAL SALARY FOR DEPARTMENT’ and the name of the department. The Section is null and the Footing consists of the salary totals for that department assuming that 10000 and 15000 are declared as $\text{SUM}$(SALARY). In the more usual case, the Heading, the Section and the Footing are all non-empty.

**EXAMPLE-VALUE DECLARATIONS**

Example-values on the Report Specimen must all be declared to enable the Report Formatter to decompose the Report Specimen into its constituent parts and produce the Internal Report Program. The current version of the Report Formatter imposes the restriction that all example-values declared on the Report Specimen must be unique. The syntax of these declarations is given below in a BNF-like grammar. Use of left-recursion has been avoided.

\[
\langle \text{EX-VAL-DECL} \rangle \quad ::= \text{EX-VAL-LIST}
\]
\[
\langle \text{EX-VAL-LIST} \rangle \quad ::= \text{EX-VAL} \quad (, \text{EX-VAL}) \ldots
\]
\[
\langle \text{EXPRESSION} \rangle \quad ::= \langle \text{TERM} \rangle - \langle \text{TERM} \rangle
\]
\[
\langle \text{TERM} \rangle \quad ::= \langle \text{FACTOR} \rangle \quad \langle (+ | -) \text{TERM} \rangle \ldots
\]
\[
\langle \text{FACTOR} \rangle \quad ::= \langle \text{EXRM} \rangle \quad \langle \text{EXRM} \rangle \ldots
\]
\[
\langle \text{EXRM} \rangle \quad ::= (\langle \text{EXPRESSION} \rangle | \langle \text{FIELD} \rangle | \langle \text{NUMERIC} \rangle | \langle \text{FUNCTION} \rangle
\]

**FUNCTION**

\[
\langle \text{FUNCTION} \rangle \quad ::= \langle \text{SUM} \rangle \quad \langle \text{MIN} \rangle \quad \langle \text{MAX} \rangle \quad \langle \text{AVG} \rangle \quad \langle \text{CNT} \rangle
\]

where (A) . . . means that A may occur zero or more times.

and ‘(‘ represents the actual character (rather than the metasymbol ‘(‘. Similarly for ‘)’.

**CHOOSING EXAMPLE-VALUES**

The format of the printable items is derived from the Example-values on the Report Specimen. Choice of Example-values is therefore important in the case of numeric and date values. For numeric items it is necessary to indicate the number of decimal parts required and also whether leading zeroes are required in the printing of the integral part. For example the declaration 0100. 2000 = PARTNO indicates that the field PARTNO is to be printed as an integer of four digits with leading zeroes. The declaration 10-00, 120-00 = UNICTOST*QTY indicates that the value of UNICTOSY*QTY should be printed as a real number with two decimal places. Leading zeroes are not required (the length of the integral part differs).

A wider range of possibilities exist when choosing a date Example-value. The Example-value must be chosen such that the components of the date value i.e. day, month and year field can be uniquely determined. For example: 01 JAN 60 = BIRTHDATE clearly indicates that 01 represents the day and 60 represents the year. Furthermore, JAN represents the month and should be printed as a monthname in abbreviated form. Text may also be associated with a date value, for example MONTH IS JAN 80 = BIRTHDATE where ‘MONTH IS’ is assumed to be text. The declaration 26–01–21, 02–12–56 = BIRTHDATE is an interesting case because the Example-values individually do not uniquely determine the date components but used together it can be shown that the fields represent day, month and year respectively.

The algorithm for determining the position and format of the date components from a set of Example-values work by a process of elimination. In the above example, the values in the first position (26 and 02) indicate a year or day field. The values in the third position can only be a year field as one of the values is greater than 31. It may then be concluded that the values in the first position represent the day field and therefore the second field is the month.

**PAGE LAYOUT SPECIFICATIONS**

Simple commands have been devised when it comes to controlling the page layout. These are entered immediately after the Example-values have been declared (Table 3). Default values are assumed in all cases.

(a) **USE \( n \) LINES PER PAGE**

This command sets the maximum number of lines that can be printed on a page to \( n \).
(b) USE m COLUMNS PER LINE
Specifies the maximum column position on a line of printed report. This command only applies to a Section which is to be printed across the page. Consider the following line of a Report Specimen:

**LANGUAGES SPOKEN:** FRENCH ITALIAN
where the Example-values FRENCH and ITALIAN are derived from the same field. The command may be used here to control the maximum number of languages that should be printed on a line.
The actual report might look like:

**LANGUAGES SPOKEN:** FRENCH ITALIAN GERMAN ENGLISH

(c) PRINT x item PER PAGE. E.g. PRINT 1 DEPARTMENT PER PAGE
This command controls the number of Sections that could be printed on a page. For example the user may wish to print the records of employees of every department on a fresh page.

(d) PAGE NUMBER AT col (Example)
AT col and (Example) are both optional. col indicates the column number where the page number is to be printed. The page number is always printed on line one. Additionally the user may specify the format in which the page number is to be printed by means of an example.

Example: PAGE NUMBER (SHEET: 01)
Page number will be printed as SHEET 01, SHEET 02, ...

(e) DATE AT col (Example)
This command prints the current date at the indicated column position with the specified format. As with command (d), AT col and (Example) are both optional. The date is always printed on line two.

Example: DATE AT 40 (76 JANUARY)

**SYNTAX ANALYSIS OF A REPORT SPECIMEN**

The Syntax Analysis of the Report Specimen uses a top-down approach. First the Sections at the top level are located (more on that later) assuming the definition:

\[\text{REPORT SPECIMEN} ::= \text{(HEADING) (SECTION) (FOOTING)}\]

The Heading and the Footing are then immediately known. The first Section is then discarded and the second Section is then treated as a new Report Specimen and the same process is carried out until the innermost Section is reached when the definition:

\[\text{REPORT SPECIMEN} ::= \text{(HEADING) (FOOTING)}\] holds true

We now come to the problem of locating the Sections of the Specimen Report. Example-values of a given declaration which occur exactly as a pair are first obtained. If there is more than one such pair then the pair of Example-values which is encountered on the Report Specimen first is chosen. Suppose that the Example-values are located on line \(L_f\) and \(L_i\) where line \(L_f\) is before \(L_i\). The case where \(L_f = L_i\) will be considered later. The width of the Section is given by \(W_0 (L_i - L_f)\). The lines \(L_f\) to \(L_f + W_0 - 1\) are then compared with the lines \(L_i\) to \(L_i + W_0 - 1\). Before matching takes place, all Example-values must be substituted by an internal code. Example-values of a given declaration are assigned the same code. If the lines match then the locations of the Sections have been identified. Code is then generated for the Heading, the Footing is stacked for code generation at a later stage and the first Section together with all Example-values which are located on that Section are discarded. The second Section is then treated as an inner Report Specimen and the same process is carried out. After the complete Report Specimen have been analysed, code is then generated for all Footings which have been stacked. Note that the Heading or part of the Heading could also be situated on the line \(L_f\) as in the following case:

| DEPARTMENT | SHOE | SMITH  | 8000-00 |
| DEPARTMENT | SHOE | BROWN  | 7000-00 |

where 'DEPARTMENT SHOE' is the Heading. This Heading should be extracted and replaced by spaces before the two lines are compared.

If lines \(L_f\) to \(L_f + W_0 - 1\) do not match lines \(L_i\) to \(L_i + W_0 - 1\) then it may be necessary to start the matching before the lines \(L_f\) and \(L_i\) as in the case below:

| DEPARTMENT | TOY  | 8000  |
| DEPARTMENT | SMITH | 8000  |
| DEPARTMENT | BROWN | 7000  |
| DEPARTMENT | COSMETICS | 6000  |
| DEPARTMENT | JONES | 9000  |

The Example-values which occur exactly as a pair are TOY and COSMETICS and these are situated on lines two and seven respectively and the width of the section is five. However the lines two to six do not match the lines seven to eleven. The lines would match exactly if the matching is done one line earlier. It then follows that the Sections actually start at the lines one and six respectively.

In the case where \(L_f = L_i\), the items are printed across the page. Assuming that the column positions of the pair of Example-values are \(C_f\) and \(C_i\) respectively where \(C_f\) is before \(C_i\), then the width of the Section is \(W_h\) column positions \((C_i - C_f)\). The Sections are matched using columns \(C_f\) to \(C_f + W_h - 1\) and \(C_i\) to \(C_i + W_h - 1\). As before, it may be necessary to begin the matching before the positions where the first Example-value in each Section start as illustrated in the following line of a Report Specimen.

**NAME:** SMITH
**LANGUAGE:** FRENCH, GERMAN

The Heading is then 'NAME: SMITH'
The Sections are 'LANGUAGE: FRENCH' and 'LANGUAGE: GERMAN'
The Footing is null.
CONCLUSION

The need to design systems that are simple, effective and easy to use is becoming greater as the number of users with no programming experience continues to grow. The author believes that the concepts introduced in the Titus Report Formatter represents a step in that direction. In order to study the effectiveness of Titus Report Formatter a program has been written to monitor usage of the system. Early results are encouraging although more data need to be collected for a thorough investigation.

Acknowledgements

The author is indebted to Professor J. J. Florentin of Birbeck College for his comments and criticisms and to A. Grig of SIA Computer Services for his encouragement and support throughout the Titus Project. The author would also like to thank S. Beresford for his assistance in devising some of the examples used.

BIBLIOGRAPHY

Titus Tutor, SIA Computer Services (1979).
ANSI 74 Cobol Report Writer.

APPENDIX

1. Code generation

Titus Report Formatter generates an Internal Report Program from the Report Specimen and Example-value declarations, and this Internal Report Program is then used to produce the actual report. For ease of reading and understanding a symbolic form of this internal program is presented here. Each language statement starts with a level number where the outermost Section is assigned a level number of 01, the next inner Section a level number of 02 and so on.

2. Section

This clause identifies the start of a new Section. For example 01 SECTION S1 where 01 is the level number and S1 is the name of the Section. SECTIONS effectively behave as program loops which are executed a number of times until certain conditions are met, for example the value of an item (called a BREAK item) changes.

3. SKIP n

This statement specifies the number of blank lines to be printed. Example: 01 SKIP 2

4. P (l, c) or P (l, cR)

This clause identifies the position where the next item is to be printed. l is the number of blank lines to skip and c is the column position where the next item is to be printed. If cR is used then c is a relative column position, i.e. c is added to the current column number counter to form the absolute column number. Relative column number is used when a Section is to be printed across the page.

5. VALUE (Text)

This clause specifies the text string to be printed. Example: 01 P (0,4) VALUE (PERSONNEL REPORT) The above statement causes the text string ‘PERSONNEL REPORT’ to be printed on the current line at the absolute column position four.

6. [Expr]

Expr represents the expression to be printed. Example: 02 P (1,10) [SURNAME] This statement prints the value of the field SURNAME on the next line at the column position ten.

7. BREAK (Item)

All Sections except the level 01 Section may contain a BREAK clause. Whenever the value of an item specified on the BREAK clause changes, the Section where the break occurs is re-entered. For example, if a list of employees in every department is being produced and the value of department changes, the appropriate Section must be re-entered to produce heading (if any) for the new department.

8. Counters

It is often useful, after printing a list of numeric values, to print the sum of these values at the end of the Section. It is thus necessary for the Report Formatter to create and maintain these counters at the appropriate levels. Counters are declared on the SECTION clause. Example: 02 SECTION S2 BREAK (COST) $SUM1 $CNT2 The above statement creates a SUM and a CNT counter. The number immediately after the function name identifies a particular counter. These counters will be initialized the first time the SECTION is obeyed and whenever the item specified on the BREAK clause changes value.

9. UPDATE

Execution of this statement causes the appropriate counters to be updated. Example: 04 SECTION UPDATE $SUM2