Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2180072 C 2002/02/12

(11)(21) 2 180 072

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(22) Date de dépôt/Filing Date: 1996/06/27

(41) Mise à la disp. pub./Open to Public Insp.: 1996/12/31

(45) Date de délivrance/Issue Date: 2002/02/12

(30) Priorité/Priority: 1995/06/30 (08/497,243) US

(51) Cl.Int.^{6}/Int.Cl.^{6} G06F 17/60

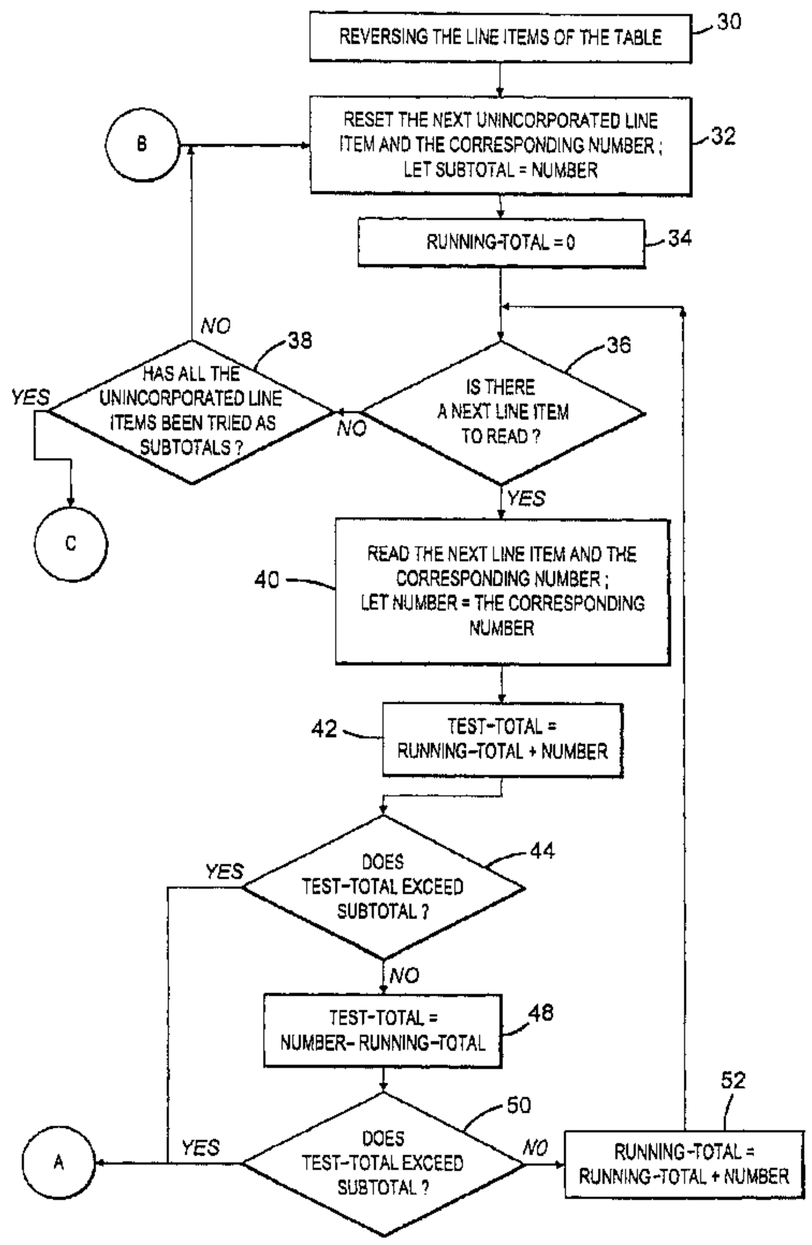
(72) Inventeur/Inventor:
Ferguson, Don Carl, US

(73) Propriétaire/Owner:
Price Waterhouse World Firm Technology Centre, US

(74) Agent: GOWLING LAFLEUR HENDERSON LLP

(54) Titre : METHODE ELECTRONIQUE POUR ETABLIR ET ANALYSER LA STRUCTURE MATHEMATIQUE D'UNE TABLE FINANCIERE

(54) Title: A METHOD FOR ELECTRONICALLY DERIVING AND PARSING THE MATHEMATICAL STRUCTURE OF A FINANCIAL TABLE



(57) Abrégé/Abstract:
A method for deriving the mathematical structure of a financial table and using the mathematical structure to categorize the components of the financial table according to the expected structure of the financial table.

ABSTRACT

A method for deriving the mathematical structure of a financial table and using the mathematical structure to categorize the components of the financial table according to the expected structure of the financial table.

Specification

A METHOD FOR ELECTRONICALLY DERIVING AND PARSING THE MATHEMATICAL STRUCTURE OF A FINANCIAL TABLE

CROSS REFERENCE TO APPENDIX

Appendixes A, B and C, which are part of the present disclosure, consist of three sheets attached herein and are listings of the software aspects of the preferred embodiment of the present invention.

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention generally relates to methods for deriving and parsing mathematical tables, in particular, a method for deriving the mathematical relationships of the line

items composing the mathematical tables in order to categorize the information.

Description of the Prior Art

Financial statements of a number of U.S. public corporations are now available electronically from a number of sources and can be obtained via the internet. In the future, all corporations will be required under the law to file their financial statements electronically. A financial statement is required to contain certain tables of information such as balance sheets, income statements, and cash flow statements, and there may be information explaining the tables and other pertinent information regarding the company.

In the electronic format, a file containing the financial statement is typically uncoded, meaning that there are no codes in the file specifically indicating the type of information represented by each line or column of text. Although the file is typically in plain ASCII text and ASCII text is conducive for reading by a person, it is not conducive for processing by a computer. In order to have the computer extract the desired information from the file, the content of the file must be identified, meaning that the various tables in the file must be recognized and the content within each table must be parsed and be broken down to their constituent parts. Once the data is recognized and broken down, it can be normalized and

manipulated. For example, the normalized data can be placed in a spreadsheet program or a database program and the performance of the company can be illustrated and analyzed by various mathematical, statistical, or financial models. The relationship between various financial statement entries can be compared and hypothetical situations can be generated and tested. Furthermore, industry analysis can be performed as well by gathering and collating data from the financial statements of several companies. Thus, there is great incentive for identifying and parsing the content of a file containing a financial statement.

There are two important considerations in the process of identifying and parsing of a file containing a financial statement. The first consideration is speed; the second consideration is accuracy.

Once the financial statement of a company is released, it will have immediate impact upon the valuation of the stock of the company. It may also, when combined with information relating to other companies, impact the valuation of the industry. Thus, it is time-critical to have the financial statement available in a form that can be manipulated for analysis. Furthermore, if a large number of financial statements must be processed, a method for processing of the statements must have reasonable computational speed. The financial statement must also be accurately recognized and

processed. Inaccurate financial information can have a disastrous impact on the decision making process. It is therefore important that means be available for facilitating timely and accurate analysis of the statements.

A method currently employed by a database company for processing financial statements requires that the information be categorized and manually entered. This is a labor-intensive process that is slow and prone to human error. Hence, there is a need for a fast and accurate method for recognizing and parsing of files containing financial statements.

There are several problems associated with the processing of a file containing a financial statement. A file containing a financial statement would include tables such as balance sheets, income statements, and cash flow statements. These tables and their locations must be identified and the line items that compose these tables must be identified as well. Referring to Fig. 1, a portion of an ASCII file containing a balance sheet is illustrated. Within each table, there may be several years of information set out in column form with column headers. The column headers and boundaries for each column need to be identified in order to identify the content of each column for each line item. Note that although the ASCII files may contain some codes indicated in angle brackets, these codes are not always present and are not sufficient as indicators for a program to properly parse the information in the files.

A problem of particular interest here is that the components within the financial table and the relationships among the components need to be identified in order to properly verify and categorize the information presented by the components of the financial table. One approach to this problem is to parse the mathematical structure of the table. In the prior art, parsing typically starts from the top of the table and proceeds to the bottom of the table. This approach proves to be time-consuming and the results produced are unsatisfactory. If there is a mistaken assumption made at the beginning of the parsing process, the mistaken assumption may not be discovered until further down the table, wasting previous efforts. In addition, the number of permutations of parsing path possibilities for this approach is quite large.

After the components making up the table are verified by the parsing process, the components composing the table must be identified and categorized so that the computer can properly process the data.

SUMMARY OF THE INVENTION

It is therefore an objective of the present invention to provide an automated method for identifying financial statements stored in uncoded electronic format such as an ASCII file.

It is another objective of the present invention to provide an automated method for identifying financial tables such as

balance sheets, income statements, and cash flow statements of a financial statement stored in uncoded format.

It is still another objective of the present invention to provide an automated method for parsing the mathematical structure of a financial table.

It is still another objective of the present invention to provide an automated method for recognizing the components of the tables.

Briefly, a preferred embodiment of the present invention provides a method for deriving the mathematical structure of a financial table and using the mathematical structure to categorize the components of the financial table according to the expected structure of the financial table.

Although the references in this application are made to financial tables, the present invention is applicable to all mathematical tables where there are mathematical relationships between the line items composing the mathematical tables.

The present invention is implemented using the programming language PROLOG. However, it is to be understood that the present invention is not limited the programming language utilized.

An advantage of the present invention is that it provides a method for identifying the constituent parts of financial statements presented in uncoded format such as an ASCII file.

Another advantage of the present invention is that it provides a method for identifying financial tables such as balance sheets, income statements, and cash flow statements of a financial statement stored in uncoded format.

Still another advantage of the present invention is that it provides a method for deriving the mathematical structure of a table.

Still another advantage of the present invention is that it provides a method for recognizing the components of the tables.

IN THE DRAWINGS

Fig. 1 is an example of a portion of an ASCII file containing a financial statement; specifically, a balance sheet is illustrated.

Fig. 2 is a pseudo-code listing of the general operation of the parsing process of the present invention.

Figs. 3a-3c is a flowchart illustrating the steps in parsing the line items of a financial table.

Fig. 4 shows the state of a balance sheet after the line items have been identified but before it is parsed.

Fig. 5 shows the subtotals found after a first pass by the parser of the present invention.

Fig. 6 shows the subtotals found after a second pass by the parser of the present invention.

Fig. 7 shows the subtotals found after a third pass by the parser of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A preferred embodiment of the present invention provides a method for processing the line items of a financial table to ensure that the line items do in fact represent the expected financial table and to identify the components of the table.

A financial table is comprised of line items, subtotals, and grand totals. By recognizing the mathematical structure of the table, the relationship between the line items, subtotals, and grand totals can be readily understood and categorized. The components of the table can be recognized once the mathematical structure of the table is recognized since a component is typically a subtotal.

The present invention provides a bottom-up parser where the series of line items are processed from the bottom of the table to the top of the table. The last line of the table is first taken as a subtotal. In traversing up the table and maintaining a running total of the numbers from the line items preceding the subtotal line, two tests may be performed for each line item being processed. In the first test, the number from the current line item is summed with the running total. If the sum equals the subtotal, the constituent line items are marked as a block having the value of the subtotal. When computing subsequent,

higher-order subtotals in subsequent passes, the blocked line items are treated as a single line item. If the sum does not equal to the subtotal, the second test is performed and determines whether or not the number in the current line subtracting the running total equals the subtotal. If the subtraction equals the subtotal, as discussed above, the constituent line items are marked as a block having the value of the subtotal, and the blocked line items are treated as a single line item for the subsequent passes.

If one or more subtotals are found, it is then necessary to make another bottom-up pass over the data to find higher-order subtotals. When there has been a pass over the data where no subtotals are found, the process stops.

However, if the parser fails to divide the table into its expected components, it may be the situation that a subtotal is not recognized because it is comprised of a sequence of additions and subtractions. The present invention provides a method that can compute "mixed" subtotals by non-deterministically negating selected line items and recomputing subtotals.

Note that the reference to a number from a line item may be a reference to a number vector as well where a given line item has more than one column. The numbers for the different columns for the same line item are processed in the same manner. The number vector allows cross-check between the columns and avoids

the possibility that a sequence of line items will add up by coincidence, since the coincidence would have to co-occur in all the columns of the table.

Once the mathematical structure of the table has been determined, the components of the table can be easily recognized and categorized via a pattern matching process.

DERIVING THE MATHEMATICAL STRUCTURE OF A TABLE

A financial table is composed of line items, subtotals of the line items, and grand totals. The technique employed by the present invention utilizes a bottom-up parser and tests each number to see if it represents a subtotal of the numbers above it. An embodiment of the present invention written in PROLOG is attached herein as Appendixes A, B and C and is expressly incorporated herein as part of this disclosure.

Fig. 2 illustrates the general pseudo-code for the bottom up parser. As described, the line items in the table, referred to as a list, are reversed for bottom-up processing. In a pass over the list, a line item is selected from the list and its number is assumed to be a subtotal. If the sum of the numbers of the line items following the selected line item equals the subtotal, the selected line item is determined to be a subtotal and the lines making up the subtotal are marked. This process repeats until each line item has been tested to determined if it is also a subtotal. If any subtotals were found in the process,

another pass is made over the list. When no more subtotals are found, the list is reversed to return the line items to their original order.

Referring to Fig. 3a, a detail flowchart illustrating the steps of the bottom-up parsing method is depicted. As indicated by block 30, the line items of a table are reversed in order to, in effect, process the data from the bottom to the top. The first line of the table is taken as a subtotal 32 and the program attempts to find line items that may sum up to this subtotal by either addition or subtraction. A variable "running_total" is first initialized to 0 34. The next step is to test whether or not there is a next line item to read. If there are no more line items to read 36, the next unincorporated line item in the table is treated as the subtotal and the process repeats until all the line items have been processed 38. If there are still line items in the table to process 36, the next line item is read and a variable, "number", is set to the number of the line item 40.

For each line item, a first test determines whether the number of the current line item plus the running total 42 equals the subtotal 44. If this is the case, the program flows to connector A to Fig. 3b. If this is not the case, a second test determines whether the number of the current line item subtracting the running total 48 equals the subtotal 50. If this is not the case, running_total is incremented by number 52

and the next line item is processed in the manner described. If this is case, like the first test, referring to Fig. 3b, connector A, the line items forming the subtotal is marked as a block, the line items within the block are returned to their original order, and the block is treated like a line item in subsequent passes 60. Note that the first line item of a block may be a subtotal and is tested in the subsequent passes. If there is another line item to be read 62, the program logic flows back via connector B to Fig. 3a, where the next line item is set as the subtotal, and an attempt is made to find other subtotal blocks.

If all the line items are processed and there were one or more subtotals found 64, another pass is made from the top of the table 66 and the process flows to connector B of Fig. 3a. In this manner, larger blocks of subtotals may be found. If no subtotals were found 64, the process flows via connector D to Fig. 3c.

If the parser was able to determine the mathematical structure of the table and the table is thus dividable into its expected components, the line items are reversed to return to their original order 72 and the process ends. If the parser fails to find all the subtotals within the table and thus the components of the table are not recognizable, a second process is used. Since a subtotal may be comprised of a sequence of additions and subtractions, the present invention provides a

method that compute "mixed" subtotals by non-deterministically negating a subset of the line items and recomputing the subtotals. First, a sequence of unincorporated line items (line items that are not part of a subtotal) is found. A subset of these line items may be determined as unrestricted line items and combination thereof may be negated. A restricted line item is determined by a heuristics process where the process looks to the label of the line item to find words which suggests that the line item can only be subtracted or added. For example, line items containing words such as "expense" or "cost" imply subtraction. For unrestricted line items, they can be either added or subtracted. In the preferred embodiment, a first combination of the unrestricted line items are negated and a pass is made over the list. If new subtotals are found, the list is re-parsed according to the algorithm described above. If no new subtotals are found, other combinations of the unrestricted line items are tried. If there are N unincorporated lines, it is only necessary to negate at most (N-1)/2 lines.

Fig. 4 shows the initial state of a balance sheet after all the line items and their corresponding numbers are identified but before it is processed by the parser. Note that a balance sheet includes an assets section as indicated by lines 3-28, a liabilities section as indicated by lines 31-41 and an equity section as indicated by lines 43-56, and a balance sheet

conforms to the formula: assets = liabilities + equities, which is illustrated by lines 28 and 59. Fig. 5 shows the subtotals found, as illustrated by boxes 80, 82, 84, and 86, after the first pass is made by the bottom-up parser. For box 80, lines 5-9 sum up to line 11. The other boxes are summed in similar manner. Note that for box 84, line 33 is determined to be a section header for the subtotal because its position immediately above the line items is used as an indication that it is a section header for the line items. Appendix C illustrates the heuristics used to find the matching header lines. The first definition for subtotal_heading in Appendix C states that a line without numbers immediately preceding the subtotal is considered the label for the subtotal. The second definition allows a non-numbered line to be the header if the header and the footer labels correspond, corresponding conditions include the following: 1) if the header and footer are identical, 2) if the header is "X," and the footer is "Total X" (e.g. "Revenues" and "Total Revenues"), 3) if the header is "X" and the footer is "X TOTAL" (e.g. "Revenues" and "Revenues Total"). Other heuristics may also be used.

Note that subtotals are treated like a line item. Thus, in the next pass, box 86, lines 43-47, and box 84 are grouped as a subtotal, as illustrated by box 90 of Fig. 6. Referring to Fig. 6, line 31 is included in box 90 as a section header using heuristics described above. Referring back to Fig. 5, box 82,

the present invention recognizes that the first line item of a subtotal, here line 19, may be a subtotal itself. In the subsequent pass, even though the line 19 is part of a subtotal, because it is the first line of a subtotal, it is treated like a line item and tested as a subtotal. Here, lines 15-17 are found to sum up to line 19 and are incorporated in box 88. In the third pass, referring to Fig. 7, all the line items are incorporated in subtotals as illustrated by boxes 92 and 94. A comparison of the subtotals shows that the two subtotals, lines 27 and 50, are equal in conformance with what is expected for a balance sheet.

At this point, the different levels of subtotals can be used to categorize the line items into their expected categories for a balance sheet and imported into a database or a spreadsheet accordingly. Referring to Appendix B, the code illustrates how the subtotals are used to categorize the table into its components. In lines 1-3, a balance sheet is recognized as being composed of an "assets" section and a "liabilities and equities" section. In lines 7-10, the first subtotal is recognized as the assets section. The number vector representing the assets is extracted and returned in the variable ATotal. In lines 11 through 21, the liabilities and equities section is recognized. On line 15, the next subtotal is recognized as the liabilities and equities section. The call

to "same_number_vector" on line 17 validates that the Assets equals to the Liabilities&Equities subtotal.

Finally, a financial table with several columns may have mathematical errors in the columns. If there are three columns and one of the columns contains a mistake, the column containing the error can be identified and the other two columns can be used as reference columns. The type of error can be identified by using the reference columns and the error can be subsequently corrected. Common errors include errors in sign, order of magnitude, difference due to the sign of a line item, or by multiples of 10 (indicating dropping of a digit). Even if there are only two columns, heuristics can be provided to detect which column contains the mistake by analyzing the mathematical structure of the table, and the error can be flagged or corrected.

What I claim is:

APPENDIX A

```
/* compute_totals(Lines, SubTotaledLines) :-
 *  Bottom-up parsing is used to find all nested subtotals in Lines.  The first pass
 *  finds the inner-most subtotals.  If any are found (Flag==1), the process is
 *  repeated, finding increasingly higher-order subtotals.
 */
compute_totals(List, Totals) :-
      compute_subtotals(List, Totals0, Flag),
      ( Flag == 1 ->
          compute_totals(Totals0, Totals)
      ; Totals0 = Totals
      ).

/* compute_subtotals(Lines, SubTotaledLines, Flag) :-
 *  Lines may contain nested subtotals.  If a sequence of lines add (or subtract) up,
 *  then SubTotaledLines will contain this structure, and Flag will be 1.
 *      If no nested subtotals are found, then Flag will be unbound,
andLines=SubTotaledLines.
 *  Note that Lines are in reverse order, so a SubTotal is followed by its parts.
 */
compute_subtotals([], [], _).
compute_subtotals([Line|Lines], [SubTotalTerm|NewLines], 1) :-
      subtotal_term(Line, SubTotalTerm, SubTotalLine, SubTotalLines, Sign),
      compute_subtotal(SubTotalLine, SubTotalLines, Sign, Lines, LinesTail),  !,
      compute_subtotals(LinesTail, NewLines, 1).
compute_subtotals([Line|Lines], [Line|NewLines], Flag) :-
      compute_subtotals(Lines, NewLines, Flag).


/* compute_subtotal(SubTotalLine, SubTotalLines, Sign, List, ListTail) :-
 *  Since the input list is reversed, the header is processed last.
 *  Heuristics are used to recognize the header (note that it can be ambiguous:
 *  the header may be associated with this subtotal, or to a higher-order one).
 *  The lines that comprise a subtotal are reversed to restore the original order.
 */
compute_subtotal(SubTotalLine, SubTotalLines, Sign, List, ListTail) :-
      number_vector(SubTotalLine, SubTotal),
      compute_total([], SubTotal, List, ListTail0, SubList, SubListTail, Sign),
      subtotal_heading(ListTail0, ListTail, SubTotalLine, SubList, SubListTail),
      reverse(SubList, SubTotalLines).

/* compute_total(RunningTotal, Total, List, ListTail, SubList, SubListTail, Sign) :-
 *  Iterates over List, summing up numbers in RunningTotal.  When the RunningTotal
 *  equals the Total, then we've found a subtotal.  The remainder of the list (ListTail)
 *  is returned. The unbound tail of SubList is returned, allowing the caller to tack on
 *  additional lines, such as the subtotal header.
 */
compute_total(RunningTotal, Total, ListTail, ListTail, SubListTail, SubListTail, '+') :-
      equivalent_vector(RunningTotal, Total), !.
compute_total(RunningTotal, Total, [LineA|ListTail], ListTail, [LineA|SubListTail],
SubListTail, '-') :-
      extract_number(LineA, NumbersA),                          % A-B-C=D
      sum_number_vector(Total, RunningTotal, NumbersBCD),
      equivalent_vector(NumbersA, NumbersBCD), !.               % A=B+C+D
compute_total(RunningTotal, Total, [Line|List], ListTail, [Line|SubList], SubListTail,
                                                                Sign) :-
      extract_number(Line, Number),
      sum_number_vector(RunningTotal, Number, RunningTotal1),
      compute_total(RunningTotal1, Total, List, ListTail, SubList, SubListTail, Sign).
```

Appendix B

```
1 balance_sheet(balance_sheet(assets(ATotal, ALines), Liabilities, Equity)) -->
2  balance_sheet_assets(assets(ATotal, ALines)),
3   balance_sheet_liabilities_equity(liabilities_equity(ATotal, Liabilities, Equity)),
4    filler,
5    !.
6
7 balance_sheet_assets( assets(ATotal, ALines) ) -->
 8    random_lines,
 9    [ total(TotalLine, '+', ALines) ],
10    { extract_number(TotalLine, ATotal) }.
11  balance_sheet_liabilities_equity(liabilities_equity(ATotal,  liabilities(LTotal,  LLines),
equity(ETotal, ELines))) -->
12 /* The balance sheet liabilities+equity section consists of a liabilities section followed
by 0 or more
13    noise lines followed by and equity section, more possible noise lines, and a total L+E
line. */
14    random_lines(LabelLines),
15    [ total(LETotalLine, '+', LELines) ],
16     { extract_number(LETotalLine, LETotal),
17       same_number_vector(ATotal, LETotal),
18         ( phrase( (liabilities(LTotal,LLines), equity([], ETotal,ELines), random_lines),
LELines)
19                    ;  phrase(  (equity(LabelLines,  ETotal,ELines),  random_lines,
liabilities(LTotal,LLines), random_lines), LELines)
20         )
21     }.
```

Appendix C

```
subtotal_heading([Label|Rest], _, _, [Label], Rest, 0, _) :-
 /* For the line immediately preceding the total, a simple
validation will suffice */
 Label = line(_,_,Blanks),
 \+ present_numbers(Blanks).
subtotal_heading([Heading|Rest], _, SubFoot, [Heading],
Rest,_, _) :-
 Heading = line(LabelToks,_,Blanks),
 \+ present_numbers(Blanks),
 labels_correspond(LabelToks, SubFoot), !.

labels_correspond(Label, Label) :- !.  /* heading and footing
are identical.*/
labels_correspond(Label, [Total | Label]) :-
total_word(Total), !.
labels_correspond(Label, SubTotal) :- append(Label, [Word],
SubTotal), total_word(Word), !.
```

CLAIMS

1. A method for parsing a financial table being comprised of a sequence of line items each having at least one label or corresponding number, comprising the steps of,

a. reversing the sequence of said line items of said financial table;

b. selecting a line item and designating the number of the selected line items as a subtotal;

c. summing the numbers from each line item following the selected line item;

d. if the sum equals said subtotal, marking all summed line items as a block, said block being treated like a line items in subsequent passes;

e. repeating steps b. - d. for each following line item until all the line items in the sequence have been tested as a subtotal;

f. if one or more sums equal the subtotals, starting from the beginning of said reversed sequence and repeating steps b. - e.; and

g. reversing the sequence of said line items of said financial table.

2. A method as recited in claim 1 wherein said summing step is performed by adding, one number at a time, the numbers of the line items following the selected line item, and comparing the sum to said subtotal.

3. A method as recited in claim 1 wherein said summing step is performed by adding, one number at a time, the numbers of the line items following the selected line item, subtracting the sum from the number of a next line item after the line items following the selected item, and comparing the result to said subtotal.

4. A method as recited in claim 1 and including after step f. the following substeps: if not all the subtotals are found,

i. designating unincorporated line items for said table as restricted or unrestricted,

ii. negating one or more of said unrestricted line items, and

iii. repeating steps b. - f.

5. A method for parsing data presented in table format and being comprised of a sequence of line items each having at least one label or corresponding number, comprising the steps of,

a. reversing the sequence of said line items of a financial table;

b. selecting a line item and designating the number of the selected line item as a subtotal;

c. summing the numbers from each line item following the selected line item;

d. if the sum equals said subtotal, marking all summed line items as a block, said block being treated like a line items in subsequent passes;

e. repeating b. - d. for each following line item until all the line items in the sequence have been tested as a subtotal;

f. if one or more sums equal the subtotals, starting from the beginning of said reversed sequence and repeating steps b. - e.; and

g. reversing the sequence of said line items of said financial table.

6. A method for verifying a financial table composed of a sequence of line items each having a label and an associated value, wherein the values of said line items forming a mathematical structure for said financial table, comprising the steps of:

a. reversing the sequence of said line items of said financial table;

b. selecting a line items and designating the number of the selected line item as a subtotal, and summing the numbers from each line item following the selected line item thereby finding subsequences of consecutive line items having sums equal to the value of a line item immediately preceding each of said respective subsequences of consecutive line items;

c. marking said subsequences of consecutive line items as blocks;

d. treating each said blocks as line items and repeating steps b, c and d, until all blocks have been found;

e. reversing the sequence of said line items of said financial table; and

f. verifying said line items and said blocks of said financial table operate to form the mathematical structure of said financial table.

7. A method as recited in claim 6 wherein said verifying step uses addition operation.

8. A method as recited in claim 6 wherein said verifying step uses substraction operation.

9. A method as recited in claim 6 wherein said verifying step includes the substeps of:

a. designating line items not in said blocks either as a restricted line item or an unrestricted line item; and

b. subtracting one or more unrestricted line items in forming said mathematical structure.

10. A method for verifying a financial table composed of a sequence of line items each having a label and an associated value, wherein the values of said line items forming a mathematical structure for said financial table, comprising the steps of:

a. selecting a line items and designating the number of the selected line item as a subtotal, and summing the numbers from each line item following the selected line item thereby finding subsequences of consecutive line items having sums equal to the value of a line item immediately following each of said respective subsequences of consecutive line items;

b. marking said subsequences of consecutive line items as blocks; and

c. treating each said blocks as line items and repeating steps a., b. and c., until all blocks have been found;

d. verifying said line items and said blocks of said financial table mathematically operate to form the mathematical structure of said financial table.

11. A method as recited in claim 10 wherein said verifying step uses addition operation.

12. A method as recited in claim 10 wherein said verifying step uses subtraction operation.

13. A method as recited in claim 10 wherein said verifying step includes the substeps of:

a. designating line items not in said blocks either as a restricted line item or an unrestricted line item; and

b. subtracting one or more unrestricted line items in forming said mathematical structure.

14. A method for verifying a table composed of a sequence of line items each having a label and an associated value, wherein the values of said line items forming a mathematical structure for said table, comprising the steps of:

a. selecting a line items and designating the number of the selected line item as a subtotal, and summing the numbers from each line item following the selected line item thereby finding subsequences of consecutive line items having sums equal to the value of a line item immediately following each of said respective subsequences of consecutive line items;

b. marking said subsequences of consecutive line items as blocks; and

c. treating each said blocks as line items and repeating steps a., b. and c., until all blocks have been found;

d. verifying said line items and said blocks of said table mathematically operate to form the mathematical structure of said table.

Fig. 1

Consolidated Balance Sheet

<TABLE>
<CAPTION>
In millions, except share data

December 31	1993	1992
<S>	<C>	<C>
Assets		
Current assets		
Cash, including cash equivalents of $160.0 and $95.9	$ 231.8	$ 147.4
Accounts receivable (less allowance for doubtful accounts of $23.1 and $23.5)	329.3	316.3
Inventories	374.9	332.2
Prepaid expenses and other	145.8	107.4
Total current assets	1,081.8	903.3
Property, plant and equipment, at cost		
Land	41.7	39.4
Buildings and improvements	499.3	478.9
Equipment	529.5	530.6
	1,070.5	1,048.9
Less accumulated depreciation	590.0	566.6
net property	480.5	482.3
Intangible assets (less accumulated amortization of $27.9 and $23.6)	145.3	144.8
Other assets	250.4	205.3
Total assets	$1,958.0	$1,735.7
Liabilities and Shareholders' Equity		
Current liabilities		
Debt maturing within one year	$ 70.4	$ 37.3
Accounts payable	378.0	365.2
Accrued compensation	59.5	62.0
Other accrued liabilities	225.7	247.7
Sales and other taxes	95.8	94.0
Income taxes	229.3	196.6
Total current liabilities	1,058.7	1,002.8
Long-term debt	123.7	177.7
Employee benefit plans	295.1	100.7
Deferred income taxes	30.5	23.0
Other liabilities (including minority interest of $43.2 and $36.9)	136.0	121.0
Commitments and contingencies		
Shareholders' equity		
Common stock, par value $.50 - authorized: 200,000,000 shares; issued - 86,528,692 and 86,445,682 shares	43.3	43.2
Additional paid-in capital	652.3	654.3
Retained earnings	150.6	126.5
Translation adjustments	(175.3)	(155.6)
Treasury stock, at cost - 14,430,073 and 14,479,431 shares	(356.9)	(357.9)
Total shareholders' equity	314.0	310.5
Total liabilities and shareholders+ equity	$1,958.0	$1,735.7

<FN>
The accompanying notes are an integral part of this statement.
</TABLE>

```
Reverse the list
    Pass over the list:
        Select a line item from the list and use its value as a subtotal
            * summing the values of the line items following the
            selected line one line at a time;
            * if the sum equals the subtotal, the line items following
            the selected line is marked as a block and treated as a
            line item;
            * include line items with 0 values in the block
            * include header lines if they are present in the block
        Repeat with the next line item as the subtotal until the list is
            empty
    If subtotals were found, pass over the list again
Reverse the list
```

Fig. 2

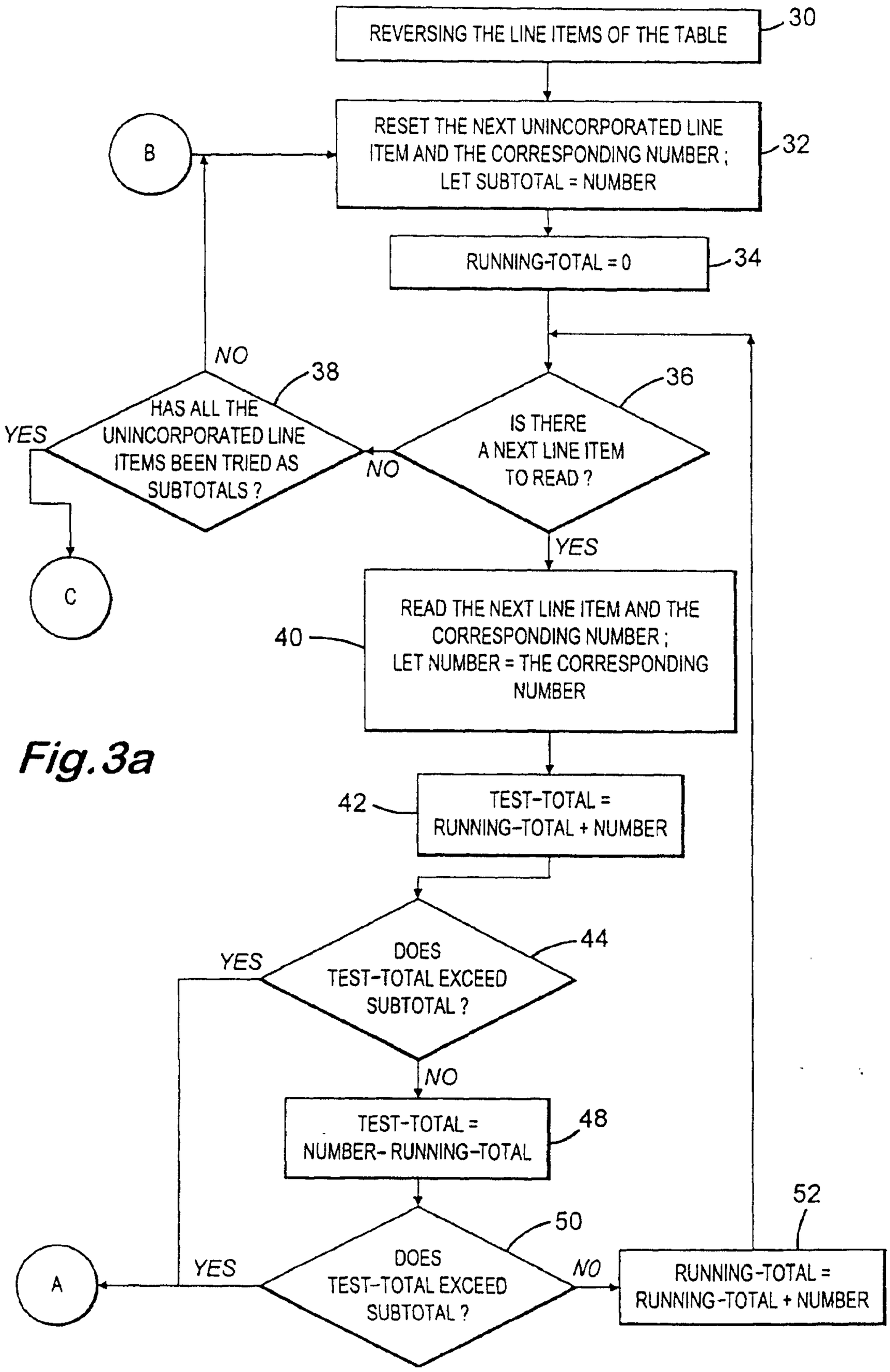

REVERSING THE LINE ITEMS OF THE TABLE
30
B
RESET THE NEXT UNINCORPORATED LINE ITEM AND THE CORRESPONDING NUMBER ; LET SUBTOTAL = NUMBER
32
RUNNING-TOTAL = 0
34
NO
38
HAS ALL THE UNINCORPORATED LINE ITEMS BEEN TRIED AS SUBTOTALS ?
YES
C
36
IS THERE A NEXT LINE ITEM TO READ ?
NO
YES
40
READ THE NEXT LINE ITEM AND THE CORRESPONDING NUMBER ; LET NUMBER = THE CORRESPONDING NUMBER
42
TEST-TOTAL = RUNNING-TOTAL + NUMBER
44
DOES TEST-TOTAL EXCEED SUBTOTAL ?
YES
NO
48
TEST-TOTAL = NUMBER- RUNNING-TOTAL
50
DOES TEST-TOTAL EXCEED SUBTOTAL ?
YES
A
NO
52
RUNNING-TOTAL = RUNNING-TOTAL + NUMBER

Fig.3a

Fig.3b

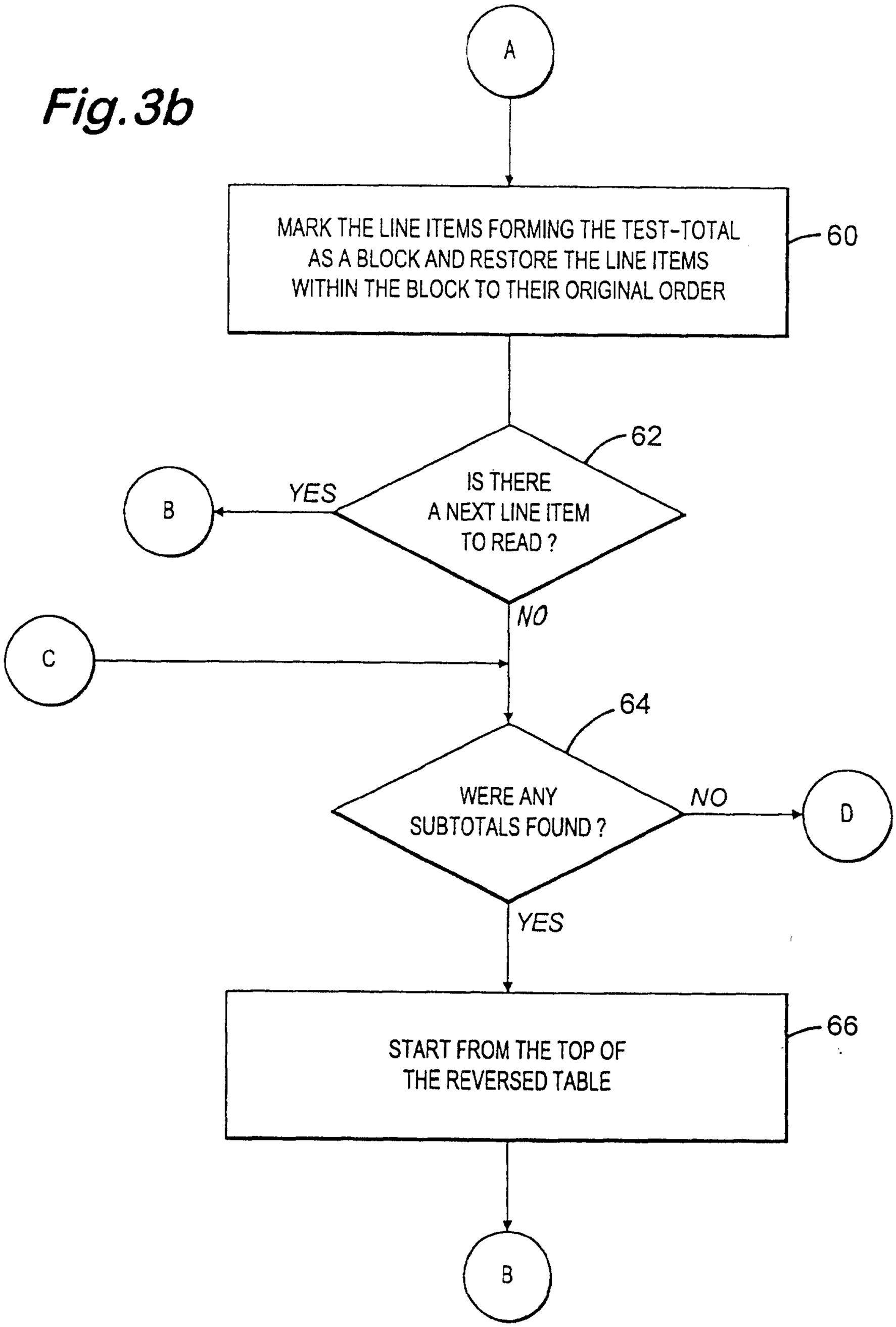
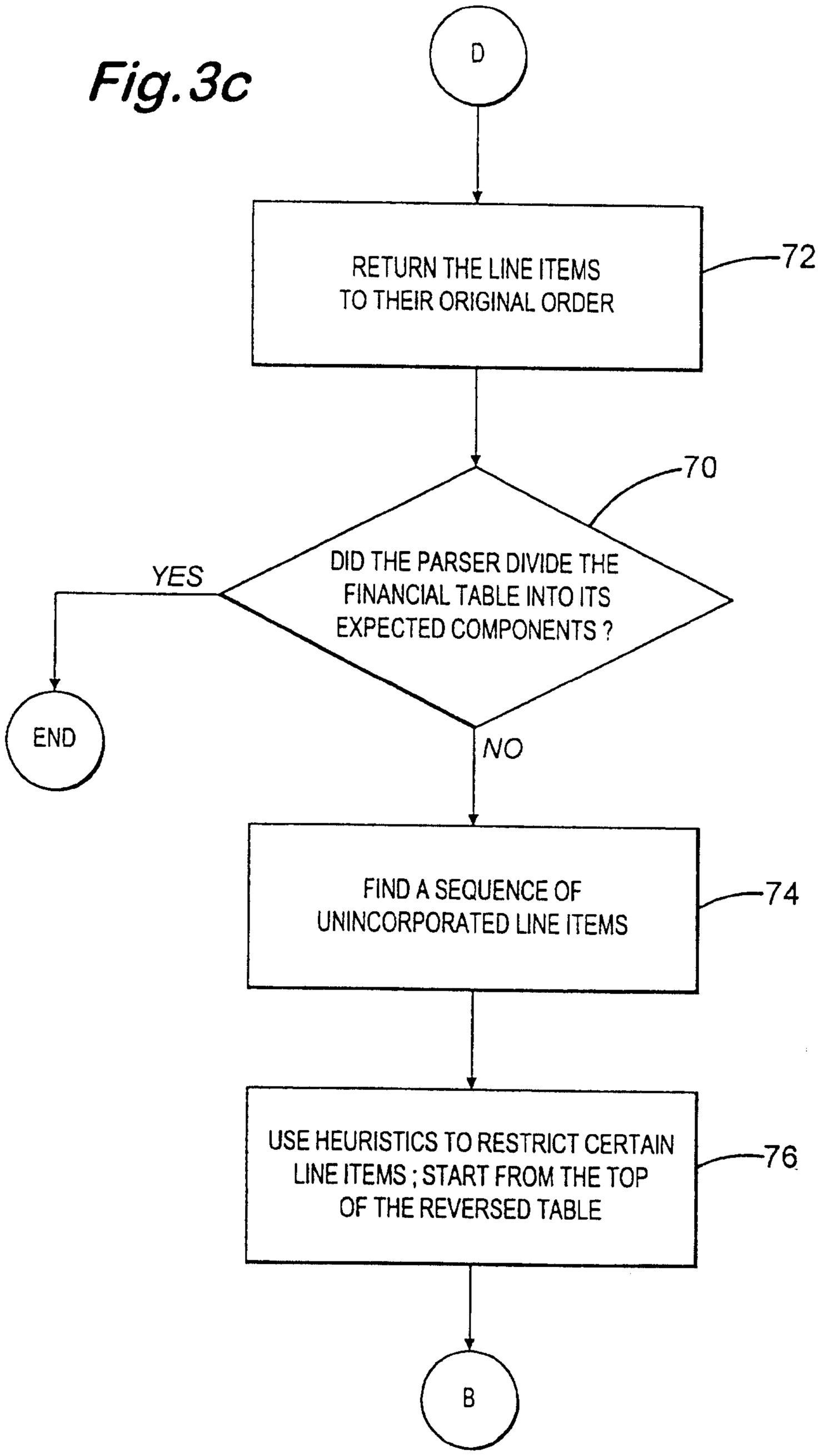

A
MARK THE LINE ITEMS FORMING THE TEST-TOTAL AS A BLOCK AND RESTORE THE LINE ITEMS WITHIN THE BLOCK TO THEIR ORIGINAL ORDER
60
IS THERE A NEXT LINE ITEM TO READ ?
62
YES
B
NO
C
WERE ANY SUBTOTALS FOUND ?
64
NO
D
YES
START FROM THE TOP OF THE REVERSED TABLE
66
B

Fig.3c


D
RETURN THE LINE ITEMS
TO THEIR ORIGINAL ORDER
72
70
DID THE PARSER DIVIDE THE
FINANCIAL TABLE INTO ITS
EXPECTED COMPONENTS ?
YES
END
NO
FIND A SEQUENCE OF
UNINCORPORATED LINE ITEMS
74
USE HEURISTICS TO RESTRICT CERTAIN
LINE ITEMS ; START FROM THE TOP
OF THE REVERSED TABLE
76
B

1	BALANCE SHEET: INITIAL STATE	
2		
3	Assets	[]
4		
5	Current assets	[]
6	Cash, including cash equivalents of $160	+ [231800,147400]
7	Accounts receivable (less allowance for	+ [329300,316300]
8	Inventories	+ [374900,332200]
9	Prepaid expenses and other	+ [145800,107400]
10		
11	Total current assets	[1081800,903300]
12		
13		
14	Property, plant and equipment, at cost	[]
15	Land	[41700,39400]
16	Buildings and improvements	[499300,478900]
17	Equipment	[529500,530600]
18		
19		[1070500,1048900
20		
21	Less accumulated depreciation	- [590000,566600]
22		---------
23	net property	[480500,482300]
24		
25	Intangible assets (less accumulated amor	[145300,144800]
26	Other assets	[250400,205300]
27		
28	Total assets	[1958000,1735700]
29		
30		
31	Liabilities and Shareholders' Equity	[]
32		
33	Current liabilities	[]
34	Debt maturing within one year	+ [70400,37300]
35	Accounts payable	+ [378000,365200]
36	Accrued compensation	+ [59500,62000]
37	Other accrued liabilities	+ [225700,247700]
38	Sales and other taxes	+ [95800,94000]
39	Income taxes	+ [229300,196600]
40		---------
41	Total current liabilities	[1058700,1002800]
42		
43	Long-term debt	[123700,177700]
44	Employee benefit plans	[295100,100700]
45	Deferred income taxes	[30500,23000]
46	Other liabilities (including minority in	[136000,121000]
47	Commitments and contingencies	[]
48		
49	Shareholders' equity	[]
50	Common stock, par value $.50 - authorize	+ [43300,43200]
51	Additional paid-in capital	+ [652300,654300
52	Retained earnings	+ [150600,126500
53	Translation adjustments	+ [-175300,-1556 0]
54	Treasury stock, at cost - 14,430,073 and	+ [-356900,-3579 0]
55		---------
56	Total shareholders' equity	[314000,310500]
57		
58		
59	Total liabilities and shareholders+ equ	[1958000,1735700]
60		

Fig. 4

1	BALANCE SHEET: AFTER FIRST PASS	
2		
3	Assets	[]
4		
5	Current assets	[]
6	Cash, including cash equivalents of $160	+ [231800,147400]
7	Accounts receivable (less allowance for	+ [329300,316300]
8	Inventories	+ [374900,332200]
9	Prepaid expenses and other	+ [145800,107400]
10		
11	Total current assets	[1081800,903300]
12		
13		
14	Property, plant and equipment, at cost	[]
15	Land	[41700,39400]
16	Buildings and improvements	[499300,478900]
17	Equipment	[529500,530600]
18		
19		[1070500,1048900
20		
21	Less accumulated depreciation	- [590000,566600]
22		---
23	net property	[480500,482300]
24		
25	Intangible assets (less accumulated amor	[145300,144800]
26	Other assets	[250400,205300]
27		
28	Total assets	[1958000,1735700]
29		
30		
31	Liabilities and Shareholders' Equity	[]
32		
33	Current liabilities	[]
34	Debt maturing within one year	+ [70400,37300]
35	Accounts payable	+ [378000,365200]
36	Accrued compensation	+ [59500,62000]
37	Other accrued liabilities	+ [225700,247700]
38	Sales and other taxes	+ [95800,94000]
39	Income taxes	+ [229300,196600]
40		---
41	Total current liabilities	[1058700,1002800]
42		
43	Long-term debt	[123700,177700]
44	Employee benefit plans	[295100,100700]
45	Deferred income taxes	[30500,23000]
46	Other liabilities (including minority in	[136000,121000]
47	Commitments and contingencies	[]
48		
49	Shareholders' equity	[]
50	Common stock, par value $.50 - authorize	+ [43300,43200]
51	Additional paid-in capital	+ [652300,654300
52	Retained earnings	+ [150600,126500
53	Translation adjustments	+ [-175300,-1556 0]
54	Treasury stock, at cost - 14,430,073 and	+ [-356900,-3579 0]
55		---
56	Total shareholders' equity	[314000,310500]
57		
58		
59	Total liabilities and shareholders+ equ	[1958000,1735700]
60		

80

82

84

86

Fig. 5

1	BALANCE SHEET: AFTER SECOND PASS	
2		
3	Assets	[]
4		
5	Current assets	[]
6	Cash, including cash equivalents of $160	+ [231800,147400]
7	Accounts receivable (less allowance for	+ [329300,316300]
8	Inventories	+ [374900,332200]
9	Prepaid expenses and other	+ [145800,107400]
10		---
11	Total current assets	[1081800,903300]
12		
13		
14	Property, plant and equipment, at cost	[]
15	Land	+ [41700,39400]
16	Buildings and improvements	+ [499300,478900]
17	Equipment	+ [529500,530600]
18		---
19		[1070500,1048900]
20		
21	Less accumulated depreciation	- [590000,566600]
22		---
23	net property	[480500,482300]
24		
25	Intangible assets (less accumulated amor	[145300,144800]
26	Other assets	[250400,205300]
27		
28	Total assets	[1958000,1735700]
29		
30		
31	Liabilities and Shareholders' Equity	[]
32		
33	Current liabilities	[]
34	Debt maturing within one year	+ [70400,37300]
35	Accounts payable	+ [378000,365200]
36	Accrued compensation	+ [59500,62000]
37	Other accrued liabilities	+ [225700,247700]
38	Sales and other taxes	+ [95800,94000]
39	Income taxes	+ [229300,196600]
40		---
41	Total current liabilities	[1058700,1002800]
42		
43	Long-term debt	[123700,177700]
44	Employee benefit plans	[295100,100700]
45	Deferred income taxes	[30500,23000]
46	Other liabilities (including minority in	[136000,121000]
47	Commitments and contingencies	[]
48		
49	Shareholders' equity	[]
50	Common stock, par value $.50 - authorize	+ [43300,43200]
51	Additional paid-in capital	+ [652300,654300]
52	Retained earnings	+ [150600,126500]
53	Translation adjustments	+ [-175300,-155600]
54	Treasury stock, at cost - 14,430,073 and	+ [-356900,-357900]
55		---
56	Total shareholders' equity	[314000,310500]
57		
58		
59	Total liabilities and shareholders+ equ	[1958000,1735700]
60		

80

88

90

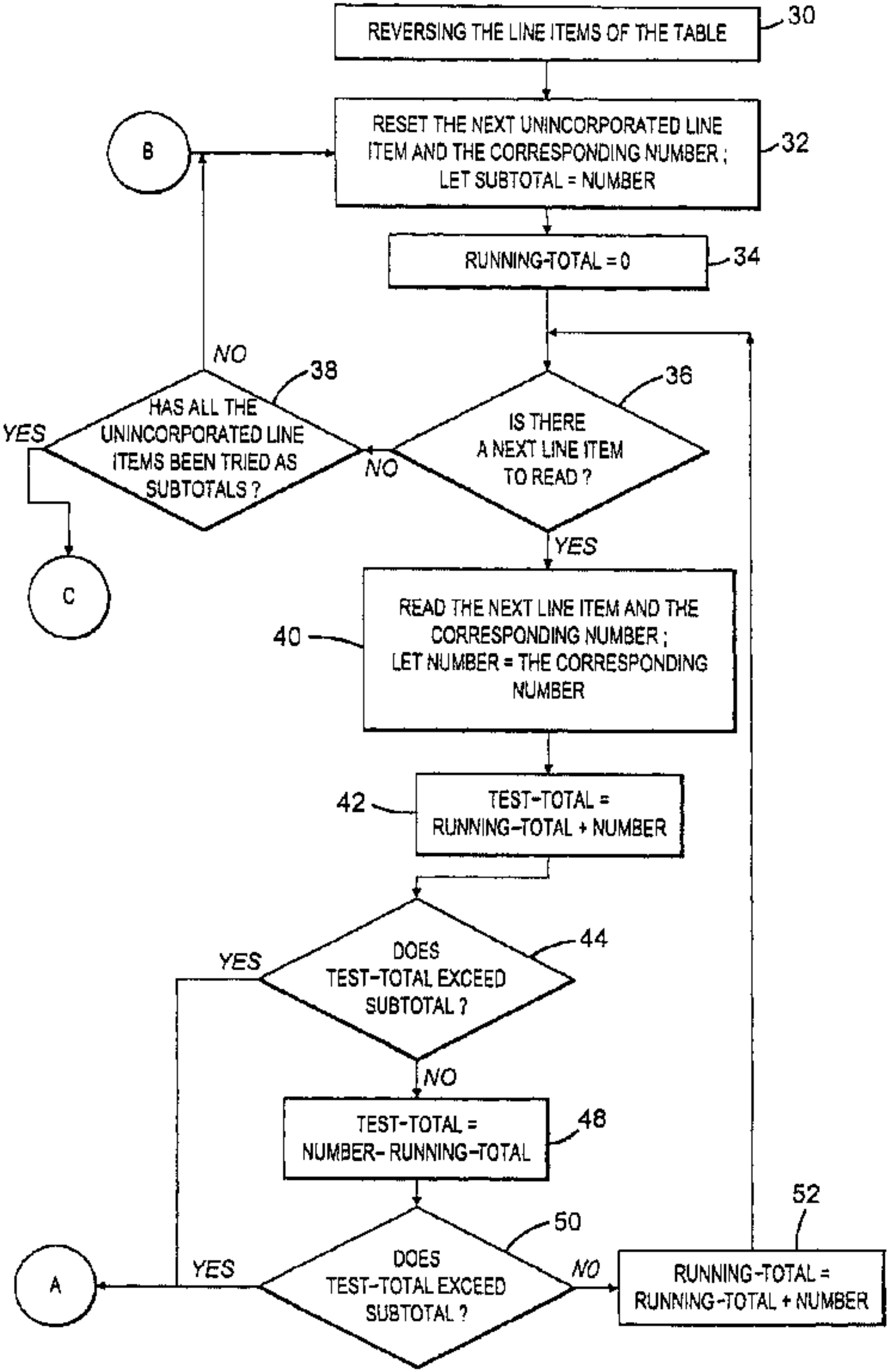
Fig. 6

1	BALANCE SHEET: AFTER THIRD PASS	
2		
3	Assets	
4		
5	Current assets	
6	Cash, including cash equivalents of $160	[]
7	Accounts receivable (less allowance for	[] + [231800,147400] + [329300,316300] + [374900,332200] + [145800,107400] --- [1081800,903300]
8	Inventories	[] + [41700,39400] + [499300,478900] + [529500,530600] --- [1070500,1048900]
9	Prepaid expenses and other	- [590000,566600] --- [480500,482300]
10		
11	Total current assets	[145300,144800] [250400,205300] [1958000,1735700]
12		
13	Property, plant and equipment, at cost	
14	Land	
15	Buildings and improvements	
16	Equipment	
17		
18		
19		
20	Less accumulated depreciation	
21		
22	net property	
23		
24	Intangible assets (less accumulated amor	
25	Other assets	
26		
27	Total assets	
28		
29		
30		
31	Liabilities and Shareholders' Equity	[]
32		
33	Current liabilities	[] + [70400,37300] + [378000,365200] + [59500,62000] + [225700,247700] + [95800,94000] + [229300,196600] --- + [1058700,1002800]
34	Debt maturing within one year	
35	Accounts payable	
36	Accrued compensation	
37	Other accrued liabilities	
38	Sales and other taxes	
39	Income taxes	
40		
41	Total current liabilities	
42		
43	Long-term debt	+ [123700,177700]
44	Employee benefit plans	+ [295100,100700]
45	Deferred income taxes	+ [30500,23000]
46	Other liabilities (including minority in	+ [136000,121000]
47	Commitments and contingencies	+ []
48		
49	Shareholders' equity	[]
50	Common stock, par value $.50 - authorize	+ [43300,43200]
51	Additional paid-in capital	+ [652300,654300]
52	Retained earnings	+ [150600,126500]
53	Translation adjustments	+ [-175300,-155600]
54	Treasury stock, at cost - 14,430,073 and	+ [-356900,-357900] --- + [314000,310500]
55		
56	Total shareholders' equity	--- [1958000,1735700]
57		
58		
59	Total liabilities and shareholders+ equ	
60		

92

94

Fig. 7


REVERSING THE LINE ITEMS OF THE TABLE
30
B
RESET THE NEXT UNINCORPORATED LINE ITEM AND THE CORRESPONDING NUMBER ; LET SUBTOTAL = NUMBER
32
RUNNING-TOTAL = 0
34
NO
38
HAS ALL THE UNINCORPORATED LINE ITEMS BEEN TRIED AS SUBTOTALS ?
YES
C
NO
36
IS THERE A NEXT LINE ITEM TO READ ?
YES
40
READ THE NEXT LINE ITEM AND THE CORRESPONDING NUMBER ; LET NUMBER = THE CORRESPONDING NUMBER
42
TEST-TOTAL = RUNNING-TOTAL + NUMBER
44
DOES TEST-TOTAL EXCEED SUBTOTAL ?
YES
NO
48
TEST-TOTAL = NUMBER- RUNNING-TOTAL
50
DOES TEST-TOTAL EXCEED SUBTOTAL ?
YES
A
NO
52
RUNNING-TOTAL = RUNNING-TOTAL + NUMBER