JUN 1 1965

SC-DC-65-1426
CONF-650527-2

MASTER

SPRING MEETING

1965

RIO GRANDE CHAPTER

ACM

A Logic for Multiple-Generation Tracing of Structures

By

W. F. Jemison

and

H. B. Souther

Thursday, Friday, May 13 & 14, 1965

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

PATENT CLEARANCE OBTAINED. RELEASE TO
THE PUBLIC IS APPROVED. PROCEDURES
ARE ON FILE IN THE RECEIVING SECTION.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

The following pages are an exact
representation of what is in the original
document folder.

ABSTRACT

A Logic for Multiple-Generation Tracing of Structures

W. F. Jemison and H. B. Souther

A logic has been developed and implemented for efficient tracing of multiple-generation structures where a common ancestor is known. An efficient tape-tracing logic is proposed for determining common ancestors. This logic was evolved in conjunction with the solution of a large-scale product-configuration data problem. The problem setting is discussed, the detailed logic is presented, trace times are discussed, and some proposals are made for minimizing tape processing time.

OUTLINE

- I. Introduction
- II. Internal Tracing
- III. External Tracing
- IV. Summary

I. INTRODUCTION

Much has been said and written, Ref. 2, 3, 4, concerning structure tracing and the parts listing problem. (Fig. 1) For the most part this has been directed at random access processing and on relatively small files. At Sandia our problem is exactly the opposite. (Fig. 1) We have a very large file and only a limited random access capability. We therefore have directed our attention to structure-tracing and parts-listing of large files on tape and would like to discuss with you some of the steps we have taken and propose to take in order to minimize tape passage time.

(Fig. 2) We have selected two portions of our overall program logic to discuss in detail. The first of these is a logic for tracing structure where a common ancestor (or end item) is known. That is, for each drawing we not only know all of its constituents but also we know all of its end item usages. This knowledge could have been gained by original input or by having been previously traced by a logic which does not depend on knowing a common ancestor. (One such logic is described in the second portion of this talk.)

II. INTERNAL TRACING

(Fig. 3) Tracing Logic #1 - Common ancestor (end item) known

Here we show the major steps in generating an indented drawing list, assuming we have a file containing both the structural information and the Indicative (or descriptive) information. Step No. 3 is the portion we have singled out for a closer look, but let us quickly run through all six steps.

(Fig. 4) The first step is to separate out purely structural information for tracing.

(Fig. 5) The next step is to sort this structure by (1) End Item, (2) Assembly, (3) Sequence Code, and (4) Constituent.

Fig. 6) Next, we start with the common ancestor and trace the tree downward until all branches have been traced and output skeleton lines of our final report.

(Fig. 7) These skeleton lines are then sorted by (1) Constituent, (2) Assembly, and (3) End Item.

(Fig. 8) ^{and} These lines are then matched with the master file and filled in with indicative information.

(Fig. 9) Lastly, these completed lines are then sorted into report sequence.

Now, let's look at Step 3 in more detail.

⁵
(Fig. 8) Tracing Logic #1 receives, as input, a tape containing the basic relational information sorted (1) by common ^{End Item} ancestor, (2) by drawing number, (3) by a relative sort code and (4) by constituent drawing number. It is important at this point for the constituent drawings to be sequenced by an artificial relative sequence code. Since we desire certain categories of drawings to be arbitrarily listed before other categories when they occur at the same structural level we preserve the relative sequence in our trace through this sequence code.

After tracing out the structure and outputting skeleton Indentured Drawing List lines we then sort these lines by (1) Constituents, (2) Next Assemblies, (3) End Items. (Fig. 6) We then match these skeleton lines with our master file and extract indicative information. ^{Fig. 9} (Fig. 7) After filling in the needed indicative information we then sort the lines back into report line sequence by (1) End Item and (2) by Line Number. (Fig. 8)

(Fig 6)

We are going to concentrate on the part of the procedure back on slide 6 that syas trace the tree downward until all branches have been traced. The sorted structure information from previous passes constitutes input to this pass.

We feel that this total proposal is a reasonable solution to the large file problem where random access capability is limited. Some time saving could be gained if the limited random access capability were used for the multiple pass trace of the assembly file. This proposal is described more fully in Reference 1.

Now this is just raw bones information and is presently set up to accommodate drawing numbers of 18 characters each. These records are 10 words long and blocked 100 records to the block. We presently are loading up to 14 such blocks into a 14,000 word array giving us the present capability of tracing any end item that is made up of no more than 1400 drawings. However, we feel that we have considerable room for expansion--this happens to be the most we expect on any one end item.

There are two ways to expand: (1) use more memory space, or (2) compress the information tighter into present space. We can discuss these in more detail later. (Fig. 10) The features that I think you might find interesting are: (1) the loading of blocked records into the 14,000 word array in a way so as to minimize processing time, (2) the way we handle tracing and output without moving this information once loaded, (3) the use of so-called binary or logarithmic searching techniques again to minimize processing time, and (4) loop termination.

(Fig. 11) First we have our inputs and outputs buffered so that we can benefit from asynchronous overlapping of input, output and processing. Here (Fig. 13) we have hypothetical data for two end items: N and T. These structure records were triggered

by an update to drawing H which appears in both N and T. (Fig. 12) Through the buffer system we load a block of records (presently 100 records) into our 14,000 word array. (Fig. 13) We look at the end item field of the first record and set START = index of the first record. Next we look at the end item field of the last record in the block and compare it with that of the first record. If they are different we know that the break point between end items occurs within the block and binary searching will quickly locate the break. (Fig. 14) If the end item fields match then a low value index is established at the last record of the block just read in and another block is loaded from the buffer and the end item field of the last record of it, is compared with that of the first record of the first block. If they are not equal then we have the breakpoint bracketted between the low value index and the top of the current block. This process repeats until the breakpoint is found or the 14,000 word array is full (which means that we have outgrown the array size). When the breakpoint is found then STOP is set equal to the array subscript representing the last word of the last record still within the end item. Thus START and STOP define that portion of the array within which we will be searching to trace the entire structure of the end item. (Fig. 15) Once we have traced the first end item, we set START equal to the previous STOP and search for the breakpoint of the second end item. When loading blocks of information, if the end of the array is reached, then loading continues "around the corner" to the beginning, thus once information is loaded it is not moved.

(Fig. 16) Now having isolated START and STOP we have information arranged like this.

We now write an output line for the end item and give it a line number, say, 1. We now look for the first appearance of the end item number in the Assembly Column. We do this by a binary search; i.e., we look first in the first line then in the last line then 1/2 way between. When we find a match we then have to find the first line that matches. We do this with another binary search. Having found the first line of end item assembly we store an index to that number in an array we call KRAY (1) and keep track of generation level in a cell called LEVEL. LEVEL is now equal to one. We store the line index for the line in which we found a match in KRAY (2) and index our level register. (Fig. 17) KRAY coupled with the level register acts like a push down list. That is, LEVEL tells you in which cell of KRAY to find the last entry. It also tells at what level in the structure tree you are presently tracing. We terminate this downward trace whenever we reach an item that is not an assembly (doesn't appear in the assembly column of JRAY) or if we reach another end item identified by a (P). To continue a trace having found a matched line the line index is loaded into KRAY and the constituent of that line is then sought in the Assembly Column. (Fig. 18) When found, it in turn is indexed in KRAY. This is continued until that particular branch of the tree is terminated (as above). When a termination is reached then we check to see if there are any more constituents on the assembly just traced by looking at the next line to see if the assembly is the same. If the assembly is the same, then we trace the constituent of that line. (Fig. 19) This continues until all constituents at the current level are traced then LEVEL is reduced by one and we return to the last constituent traced at this next lower level (Indexed by the entry in KRAY). Its next assembly is checked for more constituents (by comparing next assembly with next assembly of next line). This continues until we return to level 1, at which time the entire tree has been traced and skeleton records have been outputted for each line of the indented drawing list. This skeleton is then filled out by matching against the master file as in Figure 7.

(Fig. 7)

Now, what happens if through error (or design) we should encounter a loop (sometimes called a re-entrant tree structure). If we do not sense this we would continue in the loop indefinitely. Our solution is to check for a loop each time we enter an index into KRAY. (Fig. 20) We compare the constituent indexed by the new entry in KRAY with each of the constituents indexed at a lower level by KRAY. If we get an equal it means we are re-entering the tree in an area already traced. We output a line for the redundant item (and we could give it a special mark to indicate that it has already been traced, but we don't presently).

We have written this program in Sandia's version of Fortran II and we have defined it as a bunch of subroutines:
(Fig. 21)

1. BLD reads input tape, loads JRAY, and finds START and STOP.
2. TRACE is the main tracing subroutine and calls on other subroutines as needed.
3. FILL loads indexes into KRAY increments LEVEL and calls LOOP.
4. LOOP checks for loops.
5. OUT writes traced output lines.
6. SEARCH looks for the first line of the assembly to match a given constituent.
7. MOR checks for more constituents at the present level and adjusts arguments accordingly.
8. LIMBO writes out any records not included in the traced tree.

These are all running programs. Processing time for this pass has been running approximately 5000 lines of output per minute with up to 20 levels. I don't know if that's good or bad but it's better than we had expected. The lines we actually put out are not the simple ones I have shown but are variable length records and range from 11 words to 76 words. They include all intermediate steps or levels giving the complete chain or branch from the end item down to the current item. This is because at one time we thought we needed all this structure to resequence our lines for final report. Since that time we hit upon the idea of a line number to do this sequencing, and we could now go to fixed length, 11 word, blocked records and save some more time. At any rate we are quite pleased with this package.

I have not talked very much about what goes on before this tracing pass nor about what goes on after it. I merely said we separate out the structure information and sort it before the trace and that after the trace we sort the traced lines, match them with indicative information and resort them into report sequence. The reason I haven't talked much about this is that these portions are relatively inefficient interim programs. They are simple and they work but they take lots of machine time.

III. EXTERNAL TRACING

The second logic, I shall discuss only briefly; first, because we have not programmed it and therefore do not have it running and second, because my time is running short. But briefly the concept is this. If you have a file of 250,000 drawings or less (actually 262,144) and if you are willing to code the drawing numbers (18 bits per number), (Fig. 22) then the structure information can be squeezed into a couple of words and records

can be blocked so that the structure files are quite short on tape. Multiple passes and sorts would then be made on these dehydrated and highly compressed tapes. The main indicative file would be passed only once for updating, extracting and decoding. A final large sort on the output report lines would then organize the output reports. It would look something like this. (Fig. 23)

This scheme will give us three master files: Used-on, Assembly and Indicative. (Fig. 24) If coding is used for the first two we estimate their sizes to be as shown in Figure 24. The used-on file needs to be passed twice, the assembly file once for each generation of trace, and the indicative file once only.

IV. SUMMARY

We have gone rather hurriedly through two passes of a multiple-generation structure tracing scheme for use when for one reason or another it is desired to keep with a tape system. To keep tape passing time to a minimum coding, packing, blocking, and buffering have been suggested. Part of this scheme, we have programmed and running; the rest we feel is practical and we plan to implement it.

References

1. A Feasibility Report and Recommendation for Maintaining and Controlling a Sandia Corporation Product Definition System by the Use of Large Scale Data Processing Equipment by Electronic Data Processing, Department 3450, August 1963.
2. Software for Random Access Processing, by Charles W. Bachman, Datamation, April 1965.
3. General Information Manual, Aerospace Information and Control Systems, Engineering Data Control, IEM No. E20-8115.
4. Integrated Engineering Configuration Data Lists at TRW Space Technology Laboratories by Frank L. Czerwinski and John J. Rosati, Engineering Data Systems Workshop, November 30 - December 1, 1964.

Large File

11-12 Reels
130,000 Drawings
15,000 End Items

4 Disk Units if
full
7 Disk Units if
60% full

Small File

1-3 Reels
10,000-40,000 Drawings
600 End Items

1-3 Disk Units

Figure 1

2. SORT STRUCTURE INFORMATION

BY

1. End Item 3. Sequence Code
2. Next Assembly 4. Part

②

①

③

Part	Next Assy	End Item	Qty	Sort Code
Q	E	N	2	54
W	H	N	2	83
P	H	N	8	92
Z	H	N	6	92
E	J	N	2	54
U	J	N	1	54
V	Y	N	1	92
W	H	T	2	83
Z	T	T	5	92

Figure 5

புதுமை

2. Common End Item

1. Common End Item

TWO LOGICS FOR TRACING

TRACING LOGIC #1
(Common End Item Known)

1. Separate Structure Information from Indicative Information
2. Sort the Structure Information
3. Trace the Structure Information downward
4. Sort the Traced Lines
5. Match the Traced Lines with the Indicative File
6. Sort the Completed Lines into Report Sequence

Figure 3

1. SEPARATE STRUCTURE INFORMATION
FROM INDICATIVE INFORMATION

Figure 4

3. START WITH COMMON END ITEM AND TRACE TREE DOWNWARD UNTIL ALL BRANCHES HAVE BEEN TRACED

Output				
Level	Part	Next Assy	End Item	Line No
1	M	b	N	1
2	H	N	N	2
3	W	H	N	3
3	P	H	N	4
3	Z	H	N	5
2	F	N	N	6
2	J	N	N	7

Figure 6

5. MATCH WITH THE MASTER FILE AND EXTRACT INDICATIVE INFORMATION

Part Dwg No	Title	Next Assy	End Item	Qty	Next Assy	End Item	Qty
A	MOULDING	N	N	2	T	T	4

Level	Part Dwg	Next Assy	End Item	Line No	Title	Qty
2	A	N	N	2	MOULDING	2
2	A	T	T	2	MOULDING	4

Figure 3

4. SORT TRACED LINES

1. Part Drawing Number
2. Next Assembly
3. End Item

Level	Part No	Next Assy	End Item	Lane No
2	A	N	H	21
2	A	J	P	12
3	B	J	N	18
3	B	K	T	8
5	B	Y	M	12
5	B	Y	M	16
3	C	K	J	9
3	F	J	N	6
2	F	N	N	6
2	F	N	T	2
2	H	N	T	2
2	H	J	N	19
2	J	N	N	7
3	...	...	...	5
3	Z	...	...	11
3	Z	...	...	18

Figure 1

INPUT TAPE
(Sorted)

④	②	①	③
Q	E	N	54
W	H	N	83
P	H	N	92
Z	H	N	92
E	J	N	54
U	J	N	54
B	J	N	64
I	J	N	64
V	J	N	92
H	N	N	21
F	N	N	32
J	N	N	45
A	H	N	83
S	H	N	83
P	N	N	92
U	Q	N	54
Y	U	N	54
B	Y	N	64
V	Y	N	92
A	H	T	83
P	H	T	92
Z	H	T	92

B	K	T	64
C	K	T	64
W	K	T	83
Z	K	T	92
H	T	T	21
F	T	T	32
K	T	T	45
A	E	T	83
W	E	T	83
P	E	T	92
V	E	T	92
Z	E	T	92

Figure 12

6. SORT COMPLETED LINES INTO REPORT SEQUENCE

BY

1. End Item
2. Line Number

①

②

Level	Part Dwg	Next Assy	End Item	Line No	Title	Qty
1	N	b	N	1	AUTO	REF
2	H	N	N	2	BODY	1
3	W	H	N	3	MOULDING	2
3	P	H	N	4	BOLE	8
3	Z	H	N	5	BOLE	6
2	F	N	N	6	ENGINE	1
2	J	N	N	7	TRANSMISSION	1

Figure 9

FEATURES OF INTEREST

1. Loading of Blocked Input Records
2. Tracing and outputting without moving information in memory, by using indexes (or pointers)
3. Binary (or logarithmic) searching
4. Loop Termination

Figure 10

for Asynchronous overlapping

Figure 11.

LOADING OF BLOCKED RECORDS

Figure 15

TRACING THE TREE

Finding the End Item Assembly

Figure 16

Figure 17

TRACING THE TREE

Finding the two-level Assembly

JRAY

START

LEVEL 1/2
KRAY
(1) 93
(2) 90
(3)

0 Q E N
10 W H N
20 P H N
30 Z H N
40 E J N
50 U J N
60 B J N
70 I J N
80 V J N
90 H N N
100 F N N
110 J N N
120 A N N
130 S M N
140 P N N
150 U Q N
160 Y U N
170 B Y N
180 V Y N

Output

Level Part Next End Line
Assy Item No

1 H 6 N 1
2 H N N 2

Figure 17

Figure 24

File Name
Used-On File
Assembly File
Indicative File
1/3 reel
1/2 reel
5 1/2 reels

FILE SIZES
(Coded and Blocked)

TRACING THE TREE

Finding the Three-level Items

JRAY		START	LEVEL	1 2 3
0	Q E N			
10	W H N		KRAY	
20	P H N		(1)	93
30	Z H N		(2)	90
40	E J N		(3)	10 20 30
50	U J N		(4)	
60	B J N		(5)	
70	I J N		(6)	
80	V J N		(7)	
90	H H N			
100	F H N			
Output				
Level	Part	Next	End	Line
Assy Item No				
1	N	D	N	1
2	H	H	N	2
3	W	H	N	3
3	P	H	N	4
3	Z	H	N	5

Figure 18

Figure 13

LOADING OF JRAY
First Loading

LOOP TERMINATION

KRAY

(1)	(2)	(3)	(4)	(5)
93	120	50	0	150

Figure 20

CODED RECORDS

Used-On File:

Code No of Assy	Code No of Part	Code No of End Item	Qty	Seq Code	(P)
1 word		1 word			

Assembly File:

Code No of Assy	Code No of Part		Qty	Seq Code	(P)
1 word		1 word			

Indicative File:

Code No of Drawing	Control Information	Drwg No	Title		
1 word		3 words		16 words	

Figure 22

TRACING LOGIC #2
(Common End Item Not Known)

1. Separate the Update Structure from the Update Indicative Information
2. Match the Used-On File with the Update Structure
3. Sort the Assembly Update Information
4. Update the Assembly File and make a multiple-pass trace of the Assembly File
5. Sort the Used-On Update Information
6. Update the Used-On File and extract Structure for internal trace
7. Sort the Structure Information
8. Trace the Structure
9. Sort the Skeleton Lines
10. Update the Indicative File and fill in the Indicative Information for the finished report and decode the coded drawing numbers
11. Sort the output lines into report sequence

Figure 23

SUBROUTINES

1. BLD reads input tape, loads JRAY, and finds START and STOP

2. TRACE is the main tracing subroutine and calls on other subroutines as needed

3. FILL loads indexes into KRAY, increments LEVEL, and calls LOOP

4. LOOP checks for loops

5. OUT writes the traced lines as output

6. SEARCH looks for the first line of the assembly to match a given part

7. NOR checks for more parts at the present level and adjusts arguments accordingly

8. LIMBO writes out any records not included in the traced tree

Figure 21