A Comparison of Data Structures in Computer Graphics

KOICHI FURUKAWA*

1. *Introduction*

Various kinds of data structures [1] which have been so far developed in Computer Graphics to describe data with complex relations are investigated mainly in view of their abilities to represent the generalized directed graph. They are classified in terms of their capabilities. It is described that the data structure based on the associative triple is equivalent in its ability to the most complex ring-based data structure such as ASP.

Detailed comparisons between the two structures are made with respect to memory space and search time.

2. *Data structures based on linked list*

There exists a problem how to represent directed graphs by data structure.

A directed graph is defined as a set of nodes and a set of arcs, with direction, each of which connects two nodes. The directed graph defined above is not sufficient to represent many problems, therefore two more attributes are needed. One is some data associated with each node and the other is type information associated with each arc. The augmented directed graph is called a generalized directed graph (GDG).

A model to be represented was fixed so that differences among data structures could be derived easily and is shown in Fig. 1. $N1 \sim N4$ are names of nodes and $a1$, $a2$ are type information.

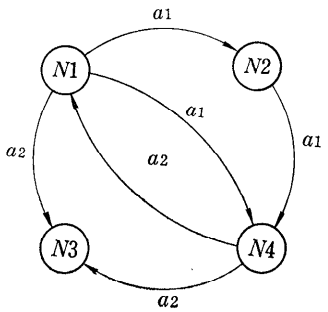

Fig. 1. A GDG model.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 11, No. 9 (1970), pp. 523-532.
* Electrotechnical Laboratory.

All nodes are given storage blocks. Each contains all of data associated with the node and some information about the arcs linked to it.

2.1 *The extention of plex* [2]

A storage block has one type information and one link pointer for each arc which starts from the node, as shown in Fig. 2. This structure is the most simple

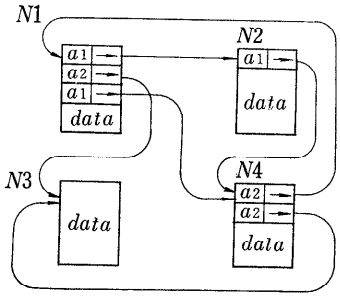

Fig. 2. A model representation by the extention of plex.

one and its accessibility is limited. For example, node $N1$ cannot be reached from node $N3$ through an arc with $a2$.

2.2 *The ring-based data structure*

To make up for the above defect, the idea of ring structure has been introduced. Here, the ring means a circular list with a head cell. The head cell is called a ring-start and the other cells are called ring-ties.

For each type of arcs starting from a node, put a ring-start in the corresponding storage-block, and for any destination of arcs with that type starting from the node, put a ring-tie in the destination storage block. Fig. 3 shows this structure.

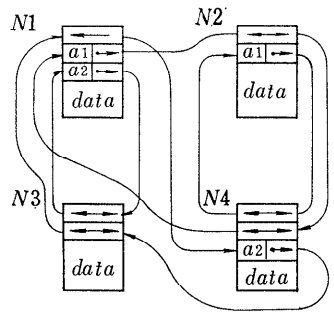

Fig. 3. A model representation by the ring-based data structure.

APL [1], [3] (Associative Programming Lang.), CORAL [1] (Class Oriented Ring Association Lang.) and SKETCHPAD [1] have this structure.

This structure has almost complete accessibility. But it has a serious drawback about the flexibility. If one fixes the format of blocks, GDG's representable

by this structure are limited.

This drawback can be removed by slight modification of this structure.

2.3 *The modified ring-based data structure*

To allow that any number of ring-starts and ring-ties may be associated with any node, a ring of ring-starts and a ring of ring-ties have been provided. They are called lower ring (L-ring) and upper ring (U-ring) respectively. The ring-starts of the above two rings are put in the storage blocks.

So, this data structure consists of three different kinds of rings, i. e., L-rings, U-rings and the rings which appeared in the previous section. The last ring is called a center ring and denoted by C-ring. Fig. 4 shows this complex structure.

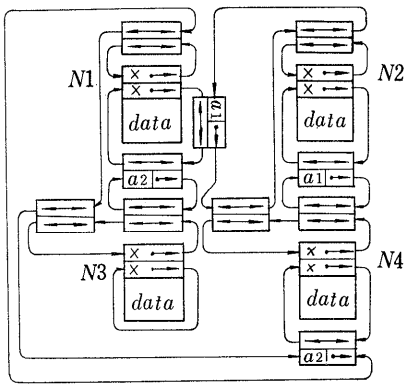

Fig. 4. A model representation by the modified ring-based data structure.

ASP [1], [4] has this structure. A ring tie which lies on both C-ring and U-ring is called an associator.

3. *Data structure based on associative triple*

All facts or information can be expressed by a set of triples which consist of Object, Attribute and its Value, and so a triple can be taken as a unit of information. It is denoted as $A(O)=V$ or $\langle A, O, V\rangle$.

Any GDG can be represnted by using this triple form. For each arc a triple is constructed by assigning the type of the arc to A, the starting node to O and the destination to V. A set of triples represents a GDG.

The basic calculation on the set is to derive one or two components from the other components of triples. There are eight basic forms, as follows.

$F0: A(O)=V$ $\quad$ $F1: A(O)=x$

$F2: A(x)=V$ $\quad$ $F3: A(x)=y$

$F4: x(O)=V$ $\quad$ $F5: x(O)=y$

$F6: x(y)=V$ $\quad$ $F7: x(y)=z$

The form $F0$ asks whether the specific triple exists or not. For the other forms $F1\sim F7$, the x, y, z represent variables which take a set of items (possibly

empty) as their values.

The perfect accessibility of this structure may be attained if these eight forms operate very fast. To permit such quick response, the hash addressing technique [5] is used. Various kinds of functions are applied to perform the hash addressing. The most simple function is the module calculation by the table size. In the case of $F1: A(O)=x$, an "exclusive or" operation is applied to the pair (A, O) and then its modulo by the table size is calculated. As a result a cell index is obtained and the triple $\langle A, O, V\rangle$ is put in the cell. The transformation from (A, O) to the cell index is not one-to-one, so if the different pairs yield the same index, they are linked in a chain. If there are more than one item which satisfies the form $A(O)=x$, they are put in a ring.

This structure is not sufficient for forms which contain two variables. In the case of $F3: A(x)=y$, all triples which satisfy this form must be put in a ring. The ring-start of this ring is placed in a cell which is obtained by applying another hash function to A.

For the other forms, the triple must be put in other tables. As the entire set of forms is symmetric with respect to A, O and V, three different tables, each called A-page, O-page and V-page, have to be provided. Besides the above three tables, the item table and the data table have to be provided.

The SAIL system is an ALGOL extended language and contains this kind of data structure. The statements which deal with the associative triples are called LEAP statements. Henceforth, we refer this data structure simply to LEAP [6].

Fig. 5 shows a concrete expression by this structure (not all but only A-page).

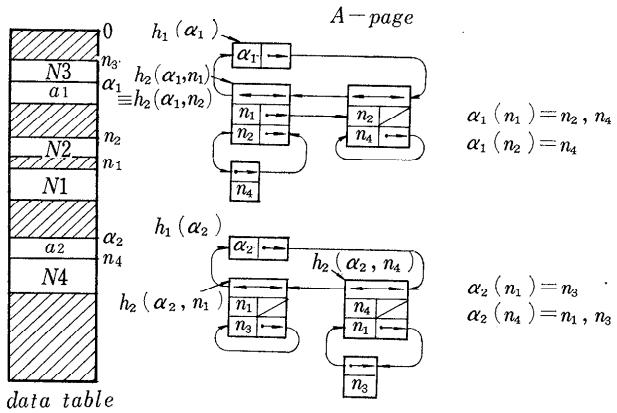

Fig. 5. A model representation by the associative triple-based data structure.

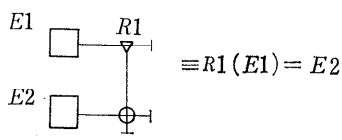

Fig. 6. The correspondence between ASP and LEAP.

The ASP and LEAP are transformed from one to another as shown in Fig. 6.

4. *The Comparison between ASP and LEAP*

ASP is compared with LEAP in their operation times (search time and construction time) and in their storage sizes.

4.1. *Operation times*

At first, search algorithms in ASP, which correspond to each form in LEAP, are provided.

In the case of $F0: a1(N1)=N2$, the search procedure starts at the given node $N1$ through the L-ring of node $N1$. If $a1$ is found, then the search goes through the C-ring. If the node $N2$ is found, the answer to $F0: a1(N1)=N2?$ is "yes". Otherwise, the answer is "no". In the case of $F1: a1(N1)=x$, the procedure is almost the same as for $F0$. The only difference is to pick up all the associators in the C-ring for $F1$. This procedure is shown as Fig. 7. In this figure, every

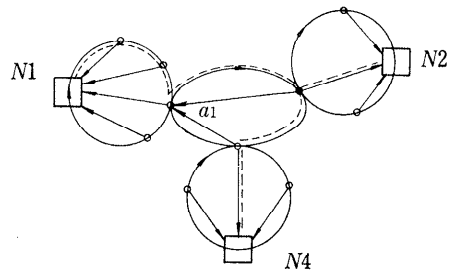

Fig. 7. The search algorithm for $F1: a1(N1)=x$.

ring-tie points to their ring-start. These pointers shorten the search times in many cases.

In the case of $F2: a1(x)=N2$, the procedure starts from the node $N2$ and traverses the three rings in reverse order. In traversing the U-ring, one must pick up all the associators whose ring-starts have the property $a1$.

In the case of $F4: x(N1)=N2$, the first step is to pick up all the elements on the L-ring of the node $N1$ and on the U-ring of the node $N2$. The answer is the intersection of these two sets.

Other search algorithms are given by the combination of the above four algorithms. It is concluded that LEAP is much better than ASP in search efficiency.

The comparison in modifying times is rather difficult because each structure has its own drawback. ASP requires search proccdures before modification. On the other hand, LEAP has to change three pages at a time to modify one triple.

4.2 *Storage sizes*

The storage sizes required by link pointer fields increase proportional to the complexity of structures.

Suppose that arc type information and a link pointer require one word each, and that there is a link pointer from each ring-tie to its ring-start.

Then the storage size required by each data structure in the previous example is,

APL: 20, ASP: 42, LEAP: 96.

In LEAP, if the link pointers for forms with two unknown variables (the *A*, *O*, *V* list in each page) are subtracted, the storage size reduces to 64.

This size is still bigger than the size required in ASP. The difference is mainly due to the conflict pointer fields in LEAP.

5. *Conclusion*

It has made to be clear that

(1) LEAP is more efficient than ASP in retrieval.

(2) LEAP requires more than twice storage space as ASP. And also ASP requires about twice storage space as APL.

(3) If the model to be represented is simple, rather simple data structures such as APL are more suitable than ASP or LEAP.

ASP represents a model by itself, that is, its structure has one-to-one correspondence to the model, deffering in their philosophies from the fact that LEAP represents a model by a set of its local properties. This difference may affect the way how to use them (in the level of programming).

The future development of hardware may repeal the whole arguments dicussed above. For instance, list processing hardwares and associative memories will reduce the search times in certain situations.

It is the most important for hardware and softwarc how to sharc tasks bctween them. In other words, the basic functions which are to be processed by hardware should be pursued.

Acknowledgement

The author wishes to express his sincere thanks to Dr. H. Nishino who is the head of the Computer Science division of Electrotechnical Laboratory.

References

[1] Gray, J. C.: Compound data structure for computer aided design; a survey. *Proc. ACM National Conference* (1967).

[2] Ross, D. T. and J. E. Rodrigues: Theoretical foundations for the computer aided design system. *SJCC* (1963).

[3] Dodd, G. G.: APL-a language for associative data handling in PL/I. *FJCC* (1966).

[4] Lang, C. A. and J. C. Gray: ASP-a ring implemented associative structure package. *CACM, 11*, 8 (August. 1968).

[5] Morris, R.: Scatter storage technique. *CACM, 11*, 1 (Jan. 1968).

[6] Feldman, J. A. and P. D. Rovner: An algol-based associative language. *CACM, 12*, 8 (August. 1969).