Memory and Algebra

C. Jousselin J-P. Moskowitz

Laboratoires d'Electronique et de Physique appliquée *

94450 Limeil-Brévannes

France

Abstract

Memory in the von Neumann computer is usually viewed as a linear array. We prove that this view does not follow from the consecutive nature of this memory, but from the group structure of the law performed in the address arithmetic unit. By changing that law, we can get a memory with a non commutative access. As an example we describe the metacyclic memory.

1 Introduction

The classical von Neumann computer is made of three parts: a central unit, a connection unit and a memory unit. The role of the central unit is twofold: it fetches data or instruction words from memory, and then processes them. The connection unit consists of the address bus and the data bus. Every piece of information flows accross these channels, called the von Neumann bottleneck. In such computers, performance is achieved through efficient information processing combined with fast retrieval of pieces of information. Therefore studies have traditionally been conducted in two distinct fields: on a software level, research has focused on languages, ways of programming and algorithms that manipulate data structures. On a hardware level, industry has provided devices where data structures are eventually stored.

Traditionally memory is viewed from the central unit as a linear array, each memory cell having a unique successor; unidimensional arrays are thus the only straightforwardly storable data structures. To cope with this allegedly fixed memory structure, D.E. Knuth proposed the linked allocation: *"we see that the linking technique, which frees us from any constraints imposed by the consecutive nature of computer memory, gives us a good deal more efficiency in some operations"* [9, page 253].

A.L. Rosenberg also deplores that *"when one implements algorithms, one is impelled by the architecture of conventional computers to impose a (possibly artificial) order on the successors of a given data cell"* [11]. He considers memory as a graph and highlights the concept of successor of a memory cell. To adapt data to its storage environment, he encodes a structurally complicated data graph in a simpler one, but the memory graph is not changed. Storing complex data structures in memory is then equivalent to embedding a data graph in the memory graph, by means of graph transformations or use of pointers.

*A member of PHILIPS Research Organization.

Use of pointers requires more indirections than necessary to compute addresses. The consequence was outlined by J. Backus: *"ironically, a large part of the traffic in the bottleneck is not useful data but merely names of data, as well as operations and data used only to compute such names"* [3, page 615].

The linear aspect of memory is assumed to be inherent to the von Neumann computer. We propose a model which proves that linearity is not the unique possible structure of memory. It is a consequence of the use of integer addition to manipulate addresses. The use of another address manipulation law allows us to consider the memory graph as a Cayley graph.

While the model of memory has remained unchanged, the objects needed by high level languages have become more and more complex. Therefore, there is a growing gap between complex data structures and their hardware implementation. In the past there have been numerous techniques proposed to reduce that semantic gap. P.M. Fenwick noticed in [5] that data accessing has been too often neglected to the benefit of data manipulation. According to him, data accessing *"traditionally has been done by applying normal arithmetic operations to addresses considered as numeric data"*. To restore some balance between data accessing and data manipulation, he proposed to handle most data accessing algorithms into separate code streams which run in a special address processor, with a dedicated instruction set.

More recently, a kind of memory manipulation coprocessor has been proposed. Its aim is *"to reduce the semantic gap between the high level languages and the hardware"*. In the proposed system, *"the processor's view of memory is not a linear array of bytes as in conventional computers, but as a collection of objects of data structure"* [2].

These two approaches artificially modify the processor's view of memory, by adding some control functions near the memory unit. Our model shows that the central unit does not *view* the memory structure, but *imposes* a structure on memory.

Section 2 reminds a few useful mathematical results. We prove in section 3 that the linear structure of memory is imposed by the law of the addressing unit; since this law is usually the law of addition of integers, the memory has the structure of a cyclic group. Section 4 emphasizes the group structure of paginated memory. In section 5 we propose a simple example of a non-commutative memory structure, along with an application to the composition of plane transformations.

2 Preliminaries

This section introduces some useful group-theoretic terms. Most notions come from [4,7,10]. We consider abstract groups presented by generators and relations fulfilled by them. In the remainder of this paper every group will be finite. By convention, we will multiply elements of a group from left to right.

Group presentation

A group G is generated by a subset X if each of its elements can be expressed as a product of members of $X^{\pm 1}$. Such a product is called a word, and a relation is an equation between two words. A set R of relations that hold in G defines the group if every relation that holds in G is a consequence of R. When this happens, we say that G is presented by X and R. We will note it $< X;\ R >$.

Graph of a group

We can represent the multiplication table of a group $(G, .)$ by a graph, also called a *Cayley diagram*, having a vertex for each element of the group. With a set of generators s_i, we associate a set of *directed* edges, say $s_i - edges$, which we assume to be indicated by a specific color C_i. Two vertices v and w are joined by a $s_i - edge$, directed from v to w, whenever $w = v.s_i$. Thus exactly one positively oriented edge of each color begins at every vertex, and exactly one positively oriented edge of each color ends at every vertex.

Any path along the edges of the graph corresponds to a word in the generators. From a graph, it is straightforward to see whether a group is Abelian or not.

Successors and predecessors

Let $(G, .)$ be a group having $\{s_1, \ldots, s_n\}$ as generating set. Let a be an element of G. The $s_i - successor$ of a is the element b of G such that $b = a.s_i$. The $s_i - predecessor$ of a is the element c of G such that $c = a.s_i^{-1}$.

3 Linear memory

Real memory consists of a set of memory cells in which a piece of information is stored. Each cell has a unique identifier called its address, which is a symbolic name. The set of these symbolic names is called the address space. Today's technological constraints impose a binary encoding of these words, but there is no arithmetic link *a priori* between the addresses 00, 01 and 10. In this section, we show that what people call *the processor's view of memory* is actually the structure of the address space; this structure, as well as the linear order of memory cells, appear as a consequence of the choice of the address arithmetic unit.

In the von Neumann computer the central unit delivers the address words on the address bus by means of addressing modes. An indexed addressing mode is used to access a piece of information relatively to another one. As an example, let us look at the interpretation of the following instruction (for a Motorola 68000 microprocessor):

$$move \;\; D_1,\; d(A_1, D_2)$$

The information stored at address D_1 is transfered to the effective destination address, computed in the address arithmetic unit as:

$$d + (A_1) + (D_2)$$

In this context addresses are considered as integers. Properties of the integer addition law "+" provide the finite set of n addresses with a cyclic group structure, isomorphic to Z_n. Such a law in the address arithmetic unit imposes the simplest commutative group structure on memory.

The address arithmetic unit gives its structure to the address space, because most addressing modes use the address composition law to travel in the address space. Subgraphs of the memory graph are the skeletons of the naturally storable structures, i.e. structures which can be directly mapped onto memory.

The Cayley graph of a cyclic group being a circle, the traditional memory map should rather be called the *cyclic memory.* In such a memory, the naturally storable structures are based on arcs of circle, i.e. unidimensional arrays. To embed more complex data structures, pointers must therefore be used.

The law of memory

By *law of memory* we mean the law of the group of the associated address space. The presentation of a finite cyclic group in terms of generators and defining relations is:

$$< t; t^n = t^0 >$$

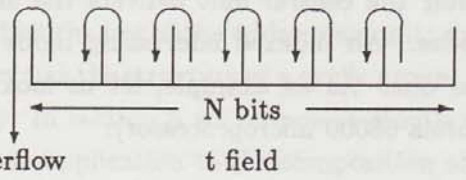
An address will be denoted t^i, where t stands for the address field, and i is the value of the address field that is sent on the address bus; i will be called the **address coordinate**.

The address composition law will be denoted ".". The coordinate composition law is the addition of integers $+$. The addressing unit achieves the composition of two input address coordinates i and j to yield the output address coordinate $i + j \ mod \ n$. In the address space, the isomorphic composition is:

$$t^i . t^j = t^{i+j}$$

A successor function defined on cells is induced by the successor function of integers. It is used for example with the program counter before fetching the next instruction ($PC \leftarrow PC + 1$).

The relation $t^n = t^0$ implies two remarks: the address coordinates are encoded over $N = log_2 n$ bits, therefore the address bus is N-wire wide; the successor address of t^{n-1} is t^0, thus during the computation of the output address coordinate, the overflow bit is not used. We shall symbolize that with the following figure:



where the arrows show the internal carry propagations inside the field t due to the implicit integer addition law.

4 Paginated memory

A paginated memory is a structure composed of m pages of n words each. The address arithmetic unit, which is either a part of the central unit or of the Memory Management Unit, provides such a memory with a structure of a direct product of cyclic groups $Z_m \times Z_n$. Address words are actually made of two fields: the s field dedicated to the page number and the t field to the offset number.

A memory cell is characterized by its address $s^i t^j$, where i is its page coordinate and j is its offset coordinate. The address composition law is the law of the group $Z_m \times Z_n$:

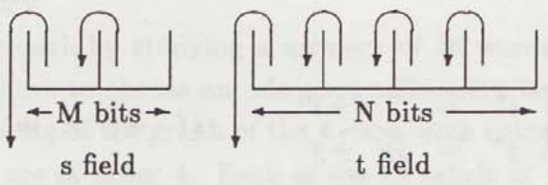
$$s^i t^j \ . \ s^{i'} t^{j'} = s^{i+i'} t^{j+j'}$$

It is realized in the addressing unit by two concurrent operations acting on the coordinates: $i + i' \bmod m$ and $j + j' \bmod n$.

The presentation of the group $Z_m \times Z_n$ in terms of generators and defining relations is:

$$< s,\ t\ ;\ \begin{array}{rcl} t^n & = & t^0 \\ s^m & = & s^0 \\ t^1 s^1 & = & s^1 t^1 \end{array} >$$

We can represent the first two relations by means of a symbolic scheme as follows:



The third relation shows the commutativity of the addressing law, and is not visible in the picture.

The Cayley graph of this direct product of finite cyclic groups can be mapped onto a torus. Hence we could call this paginated memory a *memory torus.* In such a memory, the naturally storable structures are bidimensional arrays. As in the cyclic memory, pointers must be used to embed more complex data structures.

5 A non commutative memory

We have exhibited the group properties of two useful memory structures. To obtain a more complex memory graph while preserving the fundamental properties of memory, it will be sufficient to use another address composition law with a less trivial group.

To enlarge this memory model we choose the metacyclic group family which includes the two previous commutative groups and allows us to describe a particular type of non commutative memories.

5.1 Metacyclic group

Let us first recall the definition of a metacyclic group, and give its presentation in terms of generators and relations.

Definition *A group G is called* metacyclic *if it has a normal subgroup H such that both H and the factor group G/H are cyclic.*

A finite metacyclic group has a presentation of the form:

$$< s,\ t\ ;\ \begin{array}{rcl} t^n & = & t^0 \\ s^m & = & t^p \\ t^1 s^1 & = & s^1 t^y \end{array} >$$

where the parameters $n, m, p, y \in N$ verify the following constraints:

$$\begin{array}{lcl} p, y & \leq & n \\ y^m & = & 1 \quad mod \quad n \\ yp & = & p \quad mod \quad n \end{array}$$

We consider each element g of G written with its canonical form in s and t: $s^i t^j$ with $i, j \in Z$. From the defining relations we derive the composition law "." acting on elements of the group:

$$s^i t^j \; . \; s^{i'} t^{j'} \; = \; s^{i"} t^{j"}$$

with

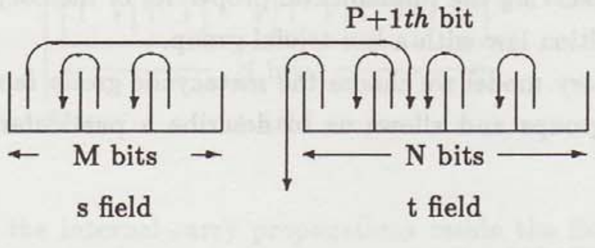
$$\left\{ \begin{array}{l} \left\{ \begin{array}{lcl} i" & = & i + i' \\ j" & = & jy^{i'} + j' \end{array} \right. \quad if \ i + i' \ < \ m \\ \\ \left\{ \begin{array}{lcl} i" & = & i + i' - m \\ j" & = & jy^{i'} + j' + p \end{array} \right. \quad if \ i + i' \ \geq \ m \end{array} \right.$$

These two equations show the law acting on the address coordinates, which is implemented in the address arithmetic unit.

5.2 Interpretation of relations

The binary encoding of coordinates imposed by today's technological constraints imply that n, m and p be powers of 2. We note N, M and P their base 2 logarithms.

We can represent the first two relations by means of the following symbolic scheme:



The third relation of the presentation shows the non commutativity of the addressing law (when $y \neq 1$), and is not implied by the picture. We call such a memory a *metacyclic memory*. The subgraphs of its Cayley graph form the bases of the naturally storable data structures.

This model appears as an extension of the traditional memory, since we can retrieve the previous memories by setting the parameters as follows:

$$y \; = \; 1 \; and \; p \; = \; 1$$

leads to the cyclic memory $Z_{n \times m}$, and

$$y \; = \; 1 \; and \; p = \; n$$

leads to the memory torus $Z_n \times Z_m$.

5.3 An easy non commutative case: $y = -1 \; mod \; n$

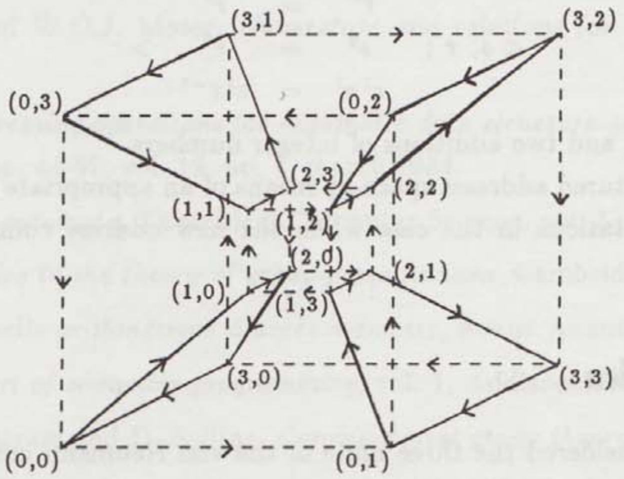
The address arithmetic unit must perform the law of composition of coordinates, which has to raise y to a certain power followed by a multiplication: $jy^{i'}$. The case $y = 1$ corresponds to a commutative memory. The next simple case is with $y = -1 \; mod \; n$, which gives a non commutative structure to the memory. The computation is reduced to a parity check: $j(-1)^{i'}$, and the determination of j'' consists of an addition or a subtraction. The complete calculation is not more complex than when the integer addition law is used.

The hypercubic memory

Let us illustrate this approach by studying a memory of 16 words which we want to appear as an oriented 4-cube. We have to choose an adequate addressing law for this *hypercubic memory*. We consider a 2-edge coloring of the graph of the 4-cube, each color corresponding to a generator. Both generators s and t are of order 4. Each of the 16 labels of the nodes is composed of two fields, the s-coordinate and the t-coordinate. The parameters of the associated metacyclic group are $(n, m, p, y) = (4, 4, 0, -1)$, i.e.:

$$< s,\ t\ ; \quad \begin{array}{ccc} t^4 & = & t^0 \\ s^4 & = & t^0 \\ t^1 s^1 & = & s^1 t^{-1} \end{array} >$$

The Cayley diagram of this group, as well as the coordinates of the addresses are given in the following figure:



where the dashed lines stand for the $t - edges$ and the solid lines for the $s - edges$. The address s^0t^1 has as $s - successor$ the address s^1t^3 and as $t - successor$ the address s^0t^2.

Main features of this hypercubic memory are a regular graph, two successors and two predecessors per memory cell, a natural orientation of orbits, and are issued from the group structure of the hypercube. This fact has already been highlighted for networks of processors by S.B. Akers and B. Krishnamurthy [1].

Dihedral memory

Data and instruction words are stored in a von Neumann architecture in the same way. Up to this point we have focused our attention on data structures. Let us now consider a simple case where the set of instructions has a mathematical group structure. The set of plane rotations and symmetries forms a group isomorphic to a dihedral group D_n. We choose to generate it with one symmetry s around an axis and one elementary rotation r of angle $2\pi/n$.

One has to be aware that there are 4 different spaces:

- the space of points P whose cartesian coordinates are (x,y);
- the space of plane transformations τ_i which are applied to the points;
- the space of matrices M_i which define them, and are applied to the points coordinates;
- the address space, in which $s^i r^j$, $i \in \{0,1\}$, $j \in \{0,\ldots,n-1\}$ uniquely defines one of the $2n$ plane transformations and its associated matrix.

To compute the image of a given point P after k successive transformations $\tau_1,\ldots,\tau_k$, one has to determine the resulting transformation $\tau = \tau_1 \circ \ldots \circ \tau_k$, and then multiply its associated matrix M by the coordinates (x,y) of P. The transformation τ can be determined either in the matrix space or in the address space. The composition of square 2 by 2 matrices requires eight multiplications and four additions of real numbers; the composition of addresses uses the simple law of the dihedral group, which is metacyclic and whose parameters are $(n,m,p,y) = (n,2,0,-1)$, i.e.:

$$< s,\ r\ ;\quad \begin{array}{ccc} r^n & = & r^0 \\ s^2 & = & r^0 \\ r^1 s^1 & = & s^1 r^{-1} \end{array} >$$

It requires one parity test and two additions of integer numbers.

The use of a well structured addresss space by means of an appropriate addressing unit reduces the complexity of computations in the case when the new address composition law is easy to implement.

6 Final remarks

In this paper we have considered the three units of the von Neumann computer as a whole, and highlighted the hidden group structure of the address space. We have shown that when the integer addition law is used to manipulate addresses, this space is a cyclic group, and memory is seen as a linear array. We showed that when another group law is performed in the address arithmetic unit, the memory graph is a Cayley graph of that group. From a practical point of view, the description of the group by means of a presentation leads directly to the interpretation of the address words; each generator is associated with an address field, and relations between generators show carry propagation relations. The composition law indicates the operations to be performed on the address coordinates. A hypercubic memory was presented as an illustration, and the use of a dihedral group of addresses enables to compose plane transformations without any matrix multiplication.

The connection with abstract models of computation could be established, having in mind the work of M. Garzon, who defined an extension of Turing Machines called "Cayley machines" [6]. He noticed that the *right* and *left* moves of a Turing machine define a single direction of motion with direct and inverse orientations. This provides the tape with an infinite cyclic group structure. He then proposes to increase the number of moves of the abstract machine tape; the forward and backward directions in each available move give a group structure to the tape.

Address arithmetic units based on groups with more than two generators should be studied in order to have new memory graphs. Future work should lead to a systematic investigation of classical data structures, to find which groups accept them as naturally storable structures. A criterion to evaluate the cost of address coordinates computation should be defined to select the simplest addressing unit.

References

[1] S.B. Akers and B. Krishnamurthy, *Group graphs as interconnection networks*, proc. 14th IEEE symposium on Fault-Tolerant Computing, pp. 422-427, june 1984.

[2] A. Asthanen, H.V. Jagadish, J.A. Chandross, D.Lin, S.C. Knauer, *An intelligent memory system*, Computer Architecture News, pp. 12-20, vol. 16, no. 4, september 1988.

[3] J. Backus, *Can programming be liberated from the von Neumann style? A functional style and its algebra of programs*, CACM, vol. 21, no. 8, pp. 613-641, august 1978.

[4] H.S.M. Coxeter and W.O.J. Moser, *Generators and relations for discrete groups*, Springer-Verlag, 1957.

[5] P.M. Fenwick, *Addressing operations for automatic data structure accessing*, Computer Architecture News, pp. 44-57, vol. 12, no. 1, march 1984.

[6] M. Garzon, *Cyclic automata*, Theoritical Computer Science, vol. 53, pp. 307-317, 1987.

[7] D.L. Johnson, *Topics in the theory of group presentations*, Cambridge University Press, 1980.

[8] C. Jousselin, *Nouvelle arithmétique d'accès mémoire*, Revue Annuelle LEP, 1987.

[9] D.E. Knuth, *The art of computer programming*, vol. 1, Addison-Wesley, 1968.

[10] W. Magnus, A. Karrass and D. Solitar, *Combinatorial group theory*, Interscience, 1966.

[11] A.L. Rosenberg, *Data graphs and addressing schemes*, J. Comp. Syst. Sci., vol. 5, pp. 193-238, 1971.

Lecture Notes in
Computer Science

Edited by G. Goos and J. Hartmanis

379

A. Kreczmar G. Mirkowska (Eds.)

Mathematical Foundations of Computer Science 1989

Porąbka-Kozubnik, Poland, August/September 1989
Proceedings

Springer-Verlag