COGNITIVA 85

Paris, 4-7 Juin 1985

PARALLEL STRUCTURES IN HUMAN AND COMPUTER MEMORY

Pentti Kanerva

Center for the Study of Language and Information
Ventura Hall, Stanford University, Stanford, CA 94305, U. S. A.

RESUME

SUMMARY

If we think of our experiences as being recorded continuously on film, then human memory can be compared to a film library that is indexed by the contents of the film strips stored in it. Moreover, approximate retrieval cues suffice to retrieve information stored in this library: We recognize a familiar person in a fuzzy photograph or a familiar tune played on a strange instrument.

This paper is about how to construct a computer memory that would allow a computer to recognize patterns and to recall sequences the way humans do. Such a memory is remarkably similar in structure to a conventional computer memory and also to the neural circuits in the cortex of the cerebellum of the human brain.

The paper concludes that the frame problem of artificial intelligence could be solved by the use of such a memory if we were able to encode information about the world properly.

The research reported in this paper was conducted at the Center for the Study of Language and Information and was made possible by a gift from the System Development Foundation.

Pentti Kanerva

INTRODUCTION

The title of my paper suggests two things: that a memory is a parallel processor and that human and computer memories are structurally similar. I will discuss both of these topics.

The memory model discussed in this paper was developed by me (Kanerva, 1984), and I will simply refer to its properties as needed. It resembles the associative-memory models of Marr (1969, 1970, 1971), Kohonen (1977, 1984) and his coworkers (Kohonen, Oja, & Lehtiö, 1981), and Willshaw (1981). The detailed comparison to computer memory is unique to my work.

It is nice to begin with something about which we can say that we truly understand it. We can say that about a computer memory because we specify it in minute detail and build it from very simple components. The point of my talking about computer memories, however, is that their organization is remarkably similar to my model of human memory. A comparison of the two can be useful in several ways. First, it can help us understand the proposed memory model. Second, the differences in the two can give us insights into intelligence, both natural and artificial: Why are some things so easy for us and so hard for computers? This, in turn, can guide research in artificial intelligence. Finally, it can suggest ways to build computer memories for artificial intelligence.

THE ORGANIZATION OF COMPUTER MEMORY

The random-access memory of a computer is an array of storage locations or registers. A location is identified by its position in the array—a sequence number—which is called the address of the location. A location stores information: a fixed-length vector of bits, a binary word. The word stored in a location is called the contents of the location. For example, today's small computer may have a memory with 100,000 locations, each with a capacity of 8 bits, and a large computer may have several million 64-bit locations. The main parts of a computer memory appear on the right in Figure 1.

Storing information (a 32-bit word) in memory is called writing and retrieving it is called reading. They involve two special registers: the address register, to hold the address of some memory location, and the datum register, to hold a word that is being transferred into or out of the memory. In writing, the contents of the addressed location (exactly one location) are changed (the datum register is copied into the addressed location, replacing the location's old contents), and in reading, the contents of the addressed location are retrieved (they are copied into the datum register).

To find the addressed location in the memory array, the memory has a network of circuits called address decoders. In principle, each storage location has its own address decoder, which will recognize the location's address and no others. When an address is placed in the address register, it becomes available to all the address decoders, but only one will recognize it as its own and make that location available for a subsequent transfer of a word.

In addition, a computer memory has circuits to control the timing of data transfers. They are included in Figure 1 for completeness.

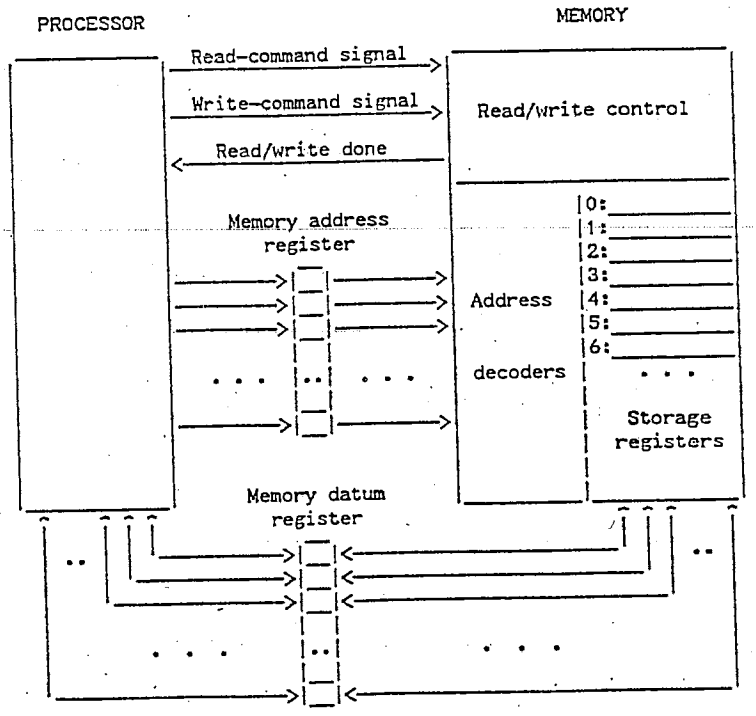

FIGURE 1. A computer organized as a processor and a memory.

Pentti Kanerva

PARALLELISM IN COMPUTERS

We usually think of the computer as operating in serial. The processor executes a program one instruction at a time, and words (data) are read from or written into memory one at a time. The nervous system, by contrast, appears to be highly parallel in its operation. This serial versus parallel operation is commonly taken to be a major difference between computers and brains. Perhaps too much has been made of that difference.

First, some parallelism is apparent even in the schematic picture of a computer in Figure 1: The bits of an address, and of a word being written or read, are transferred from one part of the computer to another in a single swoop—in parallel—instead of bit after bit. Such parallelism is present also in my model for human memory, only the model memory operates with 1,000-bit words compared to the computers' words of, say, 32 bits. Parallelism of this kind—of data paths—appears common also in the brain.

Less apparent in Figure 1 but exceedingly important in reality is the parallelism in the working of the address decoders. Recall that the function of the address decoders is to find the addressed location in the memory array. We may think of it as one primitive operation, but, in fact, a very large selection network is active in parallel and, as a result, memory access is fast (the time to find a location grows with the logarithm of memory size). Address decoding has this same parallelism in my memory model. The counterpart of address decoding in human memory is pattern recognition, which likewise appears to involve much parallel processing similar to address decoding.

By contrast, in a truly serial computer—a machine with tape for a memory, such as a Turing Machine—the memory locations are examined one after another, and it takes a long time to find the addressed location (the time grows linearly with memory size, i.e., the length of [the active part of] the memory tape). The absence of parallelism in such machines makes the accessing of memory very slow.

When computer professionals speak of parallel processing, they tend to overlook address decoding as an important instance of it and think only of many programmable processors (see Fig. 1) being active at once, working on different parts of a problem. In an extreme form of such parallel processing, the processors operate in lock-step on different pieces of large, homogeneous data. Matrix calculation is a good candidate for such parallel processing. In a more general form of parallel processing, the processors are coupled loosely to one another and can work quite independently of each other. Address decoding is a variant of the first, more restricted form of parallel processing: All the individual, primitive processors—the logic gates—are performing the same selection function at once, only they are doing it relative to different parts of the memory array.

A final note about parallelism: It is a practical issue for computers and, presumably, also for brains. It provides a way to speed up computation—even millionfold—and to increase a system's reliability, but it does not affect the class of things that a computer, in theory, is capable of.

SPARSE, DISTRIBUTED MEMORY AS A GENERALIZATION OF COMPUTER MEMORY

In this section, I will transform a fairly large computer memory into a model of human memory.

A storage location. Instead of the computer's 32-bit words, the model memory will have 1,000-bit words; instead of 20-bit addresses—enough to address one million locations—the model memory will have 1,000-bit addresses; and instead of each bit slot of a storage location containing either 0 or 1, the 1,000 bit slots of a storage location in the model memory each contain a count, a small integer. Initially, all counts are zero.

Two locations are said to be h bits apart if their addresses differ by h bits. Such a distance is called the Hamming distance; it is the number of bit positions at which two binary words differ. The number of n-bit words that are h bits away from an arbitrary n-bit word is given by the binomial coefficient 'n choose h'. Hence, the distribution of distances from an arbitrary address to all possible addresses is the binomial distribution with parameters $n = 1{,}000$ and $p = 1/2$, which is approximated by the normal distribution with mean 500 and variance 250. For example, in the model memory, the maximum distance between two locations is 1,000 bits, the mean distance is 500 bits, and .998 of the locations are at least 451 bits and at most 549 bits away from any given location.

Sparse memory. A computer memory with 20-bit addresses usually has 2^{20} locations (about one million), so it is natural to assume that a memory with 1,000-bit addresses should have $2^{1{,}000}$ locations. But that is an enormously large number; there are not that many elementary particles in the known universe. Fortunately, the memory can be made to work with a relatively small number of locations. We will assume that the model memory has 1,000,000 actual locations and that their addresses are a random sample of the $2^{1{,}000}$ possible addresses. Thus, the memory is very sparse: The median distance from a storage location to its nearest neighbor is 424 bits.

Distributed memory. When a computer memory is addressed for writing or reading, exactly one location is selected, and that location either receives or emits a word. But when a sparse memory is addressed, practically never is there an actual location with that exact address. A way to access a sparse memory is to select many locations at once. Given an address for writing or reading, select the locations that are within 450 bits of that address. In this way, nearly 1,000 memory locations closest to the read or write address will be selected (.001 of the address space lies within 451 bits of any given address).

Writing. A 1,000-bit word is stored in the model memory by storing it in all the locations that are within 450 bits of the write address. A word is stored in a location by incrementing and decrementing the location's 1,000 counters. To store 1, one is added to the appropriate counter; to store 0, one is subtracted from it. Thus, writing a word causes nearly 1,000 times 1,000 counters to be incremented or decremented. These nearly million operations are carried out in parallel.

Reading. A word is retrieved from the model memory by pooling the contents of the locations that are within 450 bits of the read address and

then finding a word that represents the pooled data. Each bit of this 1,000-bit word is determined by the majority rule: If, in the words written into the pooled locations, the bit was more often 0 than 1, the bit read will be 0; otherwise it will be 1. In practice, we add the counters across the pooled locations—in parallel—and compare the resulting 1,000 sums to zero. Notice that the read word is a statistical reconstruction and it is not necessarily identical to any of the words that has been written in memory.

Capacity. After a word has been written in a location in computer memory, it can be read by specifying the location's address. Is this also true for a sparse, distributed memory? If a word W has been written with A as the write address, can it be read by using A as the read address? It can, with a very high probability, if the total number of words written in memory is not too large—100,000 for our model memory—and if another word has not been written with an address very similar to A (two addresses are similar to each other if the Hamming distance between them is small). So even though the model memory has 1,000,000 locations, its capacity is somewhat less than 100,000 words.

CONVERGENCE TO THE BEST-MATCHING WORD AND SEQUENCE

The most significant property of the model memory is that it is sensitive to similarity. In other words, approximate retrieval cues can be used to retrieve exact information. This makes the memory a candidate for a model of human memory.

For the model to work in this way, it is necessary that a word read from memory can be used to address the memory. I call this the unifying principle of the theory. It is also necessary that the memory has not been filled to capacity. In the examples below, I will assume that 10,000 words have been written in memory. Since each word is written in nearly 1,000 locations, nearly 1,000 times 10,000 copies have been written in the million locations, or about 10 words per location.

Assume that the word X has been written with X itself as the address. That means that all locations within 450 bits of X (nearly 1,000 locations) will store one copy of X each—their counters are incremented and decremented according to the bits of X. We have already implied that reading with X as the address will retrieve X with very high probability. The reason is that we get back the nearly 1,000 copies of X, and they reinforce each other in the grand sum, plus nearly 1,000 times 10 copies of other words written in those same locations, and they mostly cancel out each other.

The interesting case is when some word X′ that is sufficiently similar to X (within 209 bits of X) is used as the reading address, because the read word X″ will be more similar to X than X′ is. Reading then with X″ as the address will retrieve the word X‴ that is even more similar to X. Fewer than ten iterations of this kind will retrieve X. The statistical argument here is similar to the one above: When the memory is read with an address X′ that is sufficiently similar to some previous write address X, writing and reading will access many common locations, each one of which holds a copy of X, and these multiple copies reinforce each other in reading. When the distance is sufficiently large (over 209 bits), the locations by which writing and reading overlap are too few to let the word X stand out against the background noise from the other words in the pooled data. In such a case, iterated reading will result in a sequence of random 1,000-bit words with little resemblance to any of the words written in memory.

A sequence of words X, Y, Z, ... can be stored in memory as a linked list by using X as the address to write Y, Y as the address to write Z, and so forth. If we then read with X as the address, we will retrieve Y, then read with Y as the address, we will retrieve Z, and so forth. In other words, we can read back the sequence by starting with its first member, just as we would with linked lists stored in conventional computer memory.

Similarity works here as well. Assuming that the sequence has been stored as a linked list and that we read with address X′ that is sufficiently similar to X, we will retrieve a word X″ that is more similar to Y than X′ is to X. With X″ as the address we will read a word X‴ that is even more similar to Z, and a few more iterations will suffice to read exact words of the stored sequence. The statistical argument for this is the same as it was above.

NEUROPHYSIOLOGICAL PARALLELS

Most interesting about the memory model is that realizing it in hardware requires components and circuits that resemble common neural components and circuits in the brain.

An address decoder that responds to a set of addresses that are within a certain distance from a specified address can be realized by a linear threshold function. Linear threshold functions, in turn, have been used commonly to model neurons, and, in fact, a neuron appears to be an ideal address decoder for a storage locations.

A storage location is made of 1,000 counters, the values of which are incremented and decremented to store 1s and 0s. Accordingly, modifiable synapses along the axon of an 'address-decoder neuron' could constitute a storage location.

In reading from memory, corresponding bit locations of many storage locations are pooled to form a single bit of output. An ideal neural structure for that would be a large, flat dendrite plane of a 'read-out neuron', perpendicular to the axons of the address-decoder neurons. It would correspond to a bit plane in computer memory.

In writing into memory, to cause a bit to be written in the very bit locations that are pooled for a single bit of output upon reading, there would have to be a 'write-in neuron' with a branching axon tree that matches the dendrite tree of a read-out neuron.

The structure of the cerebellar cortex follows very closely the above plan. The granule cells correspond to the address-decoder neurons. Their axons, called the parallel fibers, are perpendicular to the flat dendrites of the Purkinje cells, and the Purkinje cells provide the only output of the cerebellar cortex. So the Purkinje cells correspond to the read-out neurons, and the synapses of the parallel fibers with the Purkinje-cell dendrites correspond to the bit locations. Finally, the climbing fibers correspond to the axons of the write-in neurons, as they pair up with the dendrites of the Purkinje cells.

Whether the cerebellum actually works as a memory is not clear to me. However, climbing fibers similar to those in the cerebellum are common in the brain. In general, any place with a climbing fiber matching the dendrite tree of another cell is a good candidate for a pair of write-in and read-out neurons and hence a good starting point in trying to interpret the function of a neural structure.

SPARSE, DISTRIBUTED MEMORY AS A MODEL OF HUMAN MEMORY

I assume that the function of memory is to store a model of the world for use by an individual in dealing with the world. The usefulness of the model memory for this function will be discussed in the next two sections.

The 1,000-bit words on which the model memory operates can be thought of as patterns of 1,000 abstract features. We have established that it is possible to store such patterns and sequences of them in the model memory and then retrieve them by cueing the memory with those same patterns or with ones similar to them.

To apply the model to human memory, we will identify a pattern with a moment of an individual's experience. A graphic, even if crude, way to think about it is that an individual has a thousand special neurons in the brain—the equivalent of the 1,000-bit memory address register—and the momentary state of those neurons encodes the individual's subjective experience of the moment. I will henceforth call this address register the mind's focus (actually, it is a combined datum and address register; see Fig. 1). The individual experiences things through the focus. To attend to something, that something (its encoding) has to be in the focus.

The individual is coupled to the world through the senses. The senses feed into the focus, and memory storage and retrieval are through the focus. This is illustrated by Figure 2.

A succession of moments—a section of an individual's life—is then represented by a sequence of patterns. It is natural to store the sequence in memory as a linked list, because it can then be retrieved later, as has already been described. The usefulness of such storage will become clear in the next section.

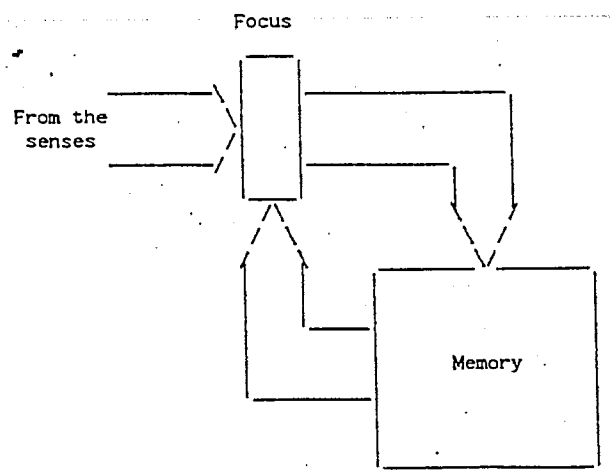

FIGURE 2. The coupling of memory with the world.

APPLICATION TO THE FRAME PROBLEM

As an application of the memory model, consider the frame problem of artificial intelligence, particularly, of robotics. I will explain the frame problem with an example.

A robot lives in a world. To function in it, the robot maintains an internal model of the world—a data base. In the data base are represented objects of the world (e.g., the robot, a cart, box 1, box 2, room 1, room 2, a rope), properties of the objects (e.g., all rooms are stationary, the cart is movable, box 1 is red, box 2 is green), and relations between the objects (e.g., the cart is in room 1, box 1 is on the cart, box 2 is on box 1).

In addition, the model of the world must specify the ways in which things interact when the robot acts on the world, say, moves the cart from room 1 to room 2. What, besides the cart (and the robot), will end up in room 2—what entries in the data base, other than the one for the cart (and the robot), must be updated? Naturally, entries for all the things resting on the cart, except those tied by a short rope to the wall—box 1, say—as they and things resting on them will fall on the floor of room 1 and will no longer be on the cart (nor on top of one another). The story can be made as complicated as one wishes.

Updating the data base as the robot interacts with the world is known as the frame problem, and it is as yet unsolved in robotics.

How do the higher animals and we humans handle the frame problem? According to the computational view of mental functions, the problem is as real to us as it is to the robot. An easy answer is that we have common sense, which the robot lacks, and we have gained it through experience. But how does common sense work? How is experience acquired and how is it used?

The memory model suggests a way to answer these questions. Assume that the situation in which the individual is at present resembles ones encountered by it in the past. The consequences of those past situations are then likely to predict what is about to happen this time. So the memory should allow the retrieval of those consequences. According to the memory model, they will be retrieved automatically if the individual's record of experiences has been stored as a linked list. Furthermore, due to the statistical nature of memory storage and retrieval, the common parts of those past consequences will reinforce each other and thus will stand out in what is retrieved from memory, the other parts being blurred away. Thus, the likely consequences of the present—a statistical abstraction based on the individual's past experience—are automatically brought to the focus.

We are still left with a major problem, namely, encoding. On that, the memory model suggests the following: First, the things that come to the focus from our senses are the very things written in memory—encoding happens outside the focus-memory loop (abstraction happens in it, as mentioned above, and encoding can be affected by what is retrieved from memory). Psychological experiments on encoding support this view, as does our subjective experience. Consider a familiar task, such as driving to work. In doing it we are constantly cued by the environment but fill in most of the detail from the inside, from memory. As long as there are no surprises—as long as what comes from the outside agrees with what

comes from memory—we are hardly aware of where the information is coming from and start paying attention only when the two disagree. Experiences that are controlled almost entirely from the inside, such as dreams and hallucinations, provide further support to this view about encoding. The subjective experience in such cases can be very real and it can be accompanied by physical signs of pleasure or fear, for example. In extreme cases it may be hard to tell whether the thing actually happened to us or whether we just "made it up."

Second, the entity in the focus is a high-dimensional vector of features (a very "large" pattern, a point of an abstract multidimensional space) that encodes everything about that moment, that is, any specific things that the individual may be attending to as well as the overall context. In that sense the memory is holistic, and whatever is retrieved from it is affected strongly by the context. This agrees with memory experiments with human subjects that have shown conclusively that recall and recognition are sensitive to manipulations of context.

Assuming that the function of memory is to store a model of the world for later reference, we can now see that this model is dynamic: The present situation (its encoding) brings to focus the consequences of similar past situations—the organism makes use of its experience. Referring back to the frame problem, the memory predicts continuously and automatically what is about to happen.

Notice, however, that this does not solve a basic problem of robotics, but it only shifts it. Whereas, before, we had chosen an encoding of the world in a data base but had a problem updating the data base (i.e., the frame problem), we now have an idea of how to make the memory work (how to maintain the data base) but are left with the problem of encoding the data. So has there been any progress? Possibly. Whereas, before, we knew neither how to encode information about the world nor how to manipulate this information, we now have a reasonable candidate for the latter. Before, we had just assumed that we can encode the information in object and property lists and in rules of manipulating the lists, but that approach led to the frame problem. The present research suggests another approach: Assuming the memory dynamics to be known, how to encode the data?

SUMMARY AND CONCLUSIONS

I have tried to describe a memory model in sufficient detail to give an idea of how it works. At this level of description the model has problems, some of which are avoided by more complete descriptions of it. Particularly worth pointing out is that the theory is valid mathematically over a very wide range of dimensions: It works for patterns with as few as 100 and as many as 100,000 components. Therefore, the theory could apply even if a pattern encoding a moment of human experience were to have 10,000 components (10,000 bits of information) instead of 1,000 and the total number of memory locations were in the hundreds of millions instead of a million. Furthermore, such large models would be practical because of the massive parallelism in writing and reading, whereas simulating them on a conventional computer would be utterly impractical.

The real story about human memory will be much more complicated than any of our models of it so far. However, it helps to understand simple models of the right kind if we want to develop more comprehensive models and eventually to understand the real phenomenon of memory itself. I have used a similar strategy by describing my memory model in terms of a computer memory, and, presumably, that helps people who already know how computer memories work.

Below are some statements about human memory that I believe to be true and that are supported by my model.

1. The mind has a focus. It is associated with the states of a set of neurons. This set is a very small compared to the total number of neurons in the brain.

2. At any moment, the information in the focus is a minute fraction of the total information stored in memory.

3. The contents of the focus serve as an address to the memory.

4. Retrieval of information from memory is iterative through the focus, making possible the retrieval of information by approximate retrieval cues.

5. We are aware of (conscious of, attend to, perceive) only things that have been brought to the focus. Stable and well-behaved states of the focus correspond to clear mental images, thoughts, and actions.

6. Some associations are inherent—they are based on form (similarity in the model)—and others are learned (frequent juxtapositions in time, experience).

7. The memory (storage) is highly distributed, and images are reconstructed statistically.

8. Writing in and reading from memory are highly parallel.

Besides possibly helping understand human memory, the present research suggests a new construction principle for computer memory: We could build a sparse, distributed memory. It would be a random-access memory that operates on patterns with a very large number of features—i.e., points of a high-dimensional, abstract space—and it would have dynamic properties resembling those of human memory. To use such a memory in robots, we would have to learn to encode information about the world into high-dimensional feature vectors. I conclude with the suggestion that studying perception and its encoding could be a fruitful line of investigation.

REFERENCES

Kanerva, P. 1984. Self-propagating search: A unified theory of memory (Rep. No. CSLI-84-7). Stanford: Center for the Study of Language and Information. To appear as a book by Bradford Books/MIT Press.

Kohonen, T. 1977. Associative memory: A system-theoretic approach. New York: Springer-Verlag.

Kohonen, T. 1984. Self-organization and associative memory. 2nd ed. New York: Springer-Verlag.

Kohonen, T., Oja, E., and Lehtiö, P. 1981. Storage and processing of information in distributed associative memory systems. In G. E. Hinton and J. A. Anderson (Eds.), Parallel models of associative memory. Hillsdale, N.J.: Lawrence Erlbaum Associates. Pp. 105–143.

Marr, D. 1969. A theory of cerebellar cortex. Journal of Physiology 202:437–470.

Marr, D. 1970. A theory for cerebral neocortex. Proceedings of the Royal Society of London B–176:161–234.

Marr, D. 1971. Simple memory: a theory for archicortex. Philosophical Transactions of the Royal Society of London B–262:23–81.

Willshaw, D. 1981. Holography, associative memory, and inductive generalization. In G. E. Hinton and J. A. Anderson (Eds.), Parallel models of associative memory. Hillsdale, N.J.: Lawrence Erlbaum Associates. Pp. 83–104.