N87-26539

SIMULATION OF AN ARRAY-BASED NEURAL NET MODEL

John A. Barnden
Computer Science Department
Indiana University
Bloomington, Indiana

ABSTRACT

Research in cognitive science suggests that much of cognition involves the rapid manipulation of complex data structures. However, it is very unclear how this could be realized in neural networks or connectionist systems. A core question is: how could the interconnectivity of items in an abstract-level data structure be neurally encoded? My answer appeals mainly to positional relationships between activity patterns within neural arrays, rather than directly to neural connections in the traditional way. The new method was initially devised to account for abstract-symbolic data structures, but it also supports cognitively useful "spatial analogue", image-like representations. As the neural model is based on massive, uniform, parallel computations over 2D arrays, the MPP is a convenient tool for simulation work, although there are complications in using the machine to the fullest advantage. An MPP Pascal simulation program for a small pilot version of the model is running.

Keywords: neural networks, connectionism, representation theory, data structures, mental models, imagery, spatial analogues.

INTRODUCTION

Artificial Intelligence and Cognitive Psychology have provided the most advanced and detailed ideas about the nature of the information processing needed in cognition. Information processing as studied in those fields typically involves the rapid manipulation of complex, very-short-term symbolic data structures. The data structures may be representations of the world, semantic structures derived from natural-language utterances, descriptions of plans, goals, and their relationships, and so on. Such structures must be created, updated, analyzed, compared, modified, committed to memory, and retrieved from memory in rapid and complex ways. AI has been much concerned with the abstract nature of the data structures and manipulations, and with their implementation on conventional computers.

I therefore adopt the working hypothesis that much of human cognition requires *rapid computational processes involving complex, temporary structures* that hold information. These structures do not, however, have to be very similar to the particular data structures proposed in current AI and cognitive science.

Now, in recent years there has been great interest in the question of how "brain hardware" operates so as to support cognitive functions. The usual assumption is that the proper hardware level at which to study cognitive functions is that of the dynamics of neural *networks*, in which individual neurons perform only quite low-level functions and communicate with each other by sending simple signals along fibres. I take this assumption as a second working hypothesis, even though I am intrigued by suggestions that other effects may be important; that, for instance, individual neurons may be capable of performing major pieces of computation in their own right.

The two working hypotheses lead to the central question addressed by the project:

> CENTRAL QUESTION
> How can rapid processing of complex, temporary information structures, of some sort or other, be accomplished by "standard" neural net circuitry?

The emphasis on *short-term* structures and processing is to be noted. The project is not at present concerned with matters of long-term memory or learning. A late stage of the project will look at such matters.

Limitations of Prevailing Ideas

Most of the research on the cognitive capabilities of neural networks has centered on visual pattern recognition, lexical processes in language understanding, the "associative" retrieval of information from long-term memory, and restricted forms of learning. But for us to progress beyond these particular specialized

functions, there must be a more thorough and general examination of how neural networks can represent and manipulate information. As is widely admitted, it is very unclear how we could answer the Central Question by applying the main current ideas about neural networks.

Most work on neural nets in AI and cognitive science is within the so-called "connectionist" paradigm [Refs. 4, 6, 8, 11, 14]. Some connectionists have recently begun to look at aspects of the central question [e.g. Ref. 13]. However, the work is in its infancy and has barely begun to address the real issues. There is reason to believe [Ref. 1] that it will be difficult to deal with these issues without adding a great deal of extra complication to existing connectionist models.

My own neural net model is a more thoroughgoing attempt to confront the issues than has previously been made. In this confrontation, I have found it beneficial to introduce some novel ideas about neural representation. The theory is thus highly atypical as a connectionist theory if one defines connectionism by appeal to existing research, although it is still connectionist in that it relies on networks of processing units that are like simplified neurons and send only simple messages to each other. The model is conjectural, though respectably in tune with neurophysiological evidence [Ref. 2]. An important feature is that it serves as a brake on uncritical acceptance of the (similarly conjectural) existing views of neural-net functioning.

The Central Representational Tenets

Here I specify the basic, general representational principles underlying the project. They could conceivably by instantiated in detail in numerous different ways. One specific instantiation that I am studying and simulating will be outlined below. However, the general principles would still be of interest even if this instantiation were abandoned.

PRINCIPLE 1

> The neural realization (implementation) of a short term data structure is composed of localized patterns of neural activity. These localized patterns are *positioned occurrences*, in some neural medium, of patterns of neural activity that are location-unspecific in that they can be instantiated at any position within that medium.

Analogy: A piece of natural language text is a data structure (in a paper/ink medium) whose parts are localized occurrences of location-unspecific visual patterns (words).

PRINCIPLE 2

> One important way in which those neural activity-pattern occurrences are "glued" together so as to form structures is to be in appropriate *relative positions* in the neural medium.

Analogy: In a piece of text the word occurrences are structured by being put into appropriate relative positions.

PRINCIPLE 3

> Activity-pattern occurrences that act as parts of an implemented data structure can also be associated with each other by being similar in some particular sense.

Analogy: a textual label used in a diagram and in a legend attached to the diagram serves to associate a part of the legend with a part of the diagram.

There is a fourth principle that I state later.

Neural Arrays

Principles 1 and 2 mentioned neural-net media in which patterns of neural activity can be given instantiations in particular positions. Thus, I am assuming that the media are such that one can make sense of the notions of "instantiation", "position" and "relative position" within a medium. To this end I have adopted the following simple assumption in my model: each medium is a *2D square array of small neural circuits.* All the arrays are of the same size, and all the small neural circuits (the array elements) are similar to each other. Each array element supports at any given time a particular pattern of neural activity, and, since the array elements are isomorphic circuits, the supported patterns can be viewed as positioned instantiations of location-unspecific patterns.

A short-term data structure, then, consists at the hardware level of configurations of pattern instantiations in one or more of the neural arrays. To continue the text or diagram analogies mentioned above, an array can be likened to a piece of paper on which words can be put in particular places.

In theory the arrayness of my neural media could be purely abstract, with no implication that they are laid out in space as geometric arrays. Nevertheless, I

do in fact assume that they are so laid out, at least approximately. This is convenient because the processing applied to the media requires adjacent array elements to communicate. Although there is some degree of broadcasting within arrays, it is overshadowed by the neighbor-neighbor interactions.

The 2D geometric arrayness of the media fits in well with current ideas about the functional structure of cortex. I say more on this later.

Spatial-Analogue Representation

Now that we have proposed an array-based computational architecture, it is a short step to the following proposal: that bodies of information that can abstractly be cast as arrays could in some cases be implemented in the architecture by adopting a natural mapping from the abstract arrays down to our neural arrays. (Cf. the standard implementation of a 1D abstract array by sequential allocation in a conventional computer memory, viewed as a 1D array of cells.) The point of this sort of "natural" implementation of abstract arrays is that it can render certain sorts of operation on the abstract information more efficient or more convenient than they would otherwise be.

There is one specific sub-proposal that much of my project is concerned with: that *2D "spatial" arrays could be naturally mapped onto the neural arrays.* By a " 2D spatial" array I mean an abstract array in which element (i,j) is meant to correspond to position (or subregion) (i,j) in some 2D projection of space. Thus, in a neural array that is being used to support a spatial array, the element (i,j) would correspond to and hold information about position (i,j) of the 2D projection of space.

Of course, having arrived at this point we observe that a neural array could be used to naturally map an *abstract* 2D space as well as a geometric one. Altogether, we are led to the following:

PRINCIPLE 4

> Relative position of activity-pattern occurrences in a neural array may (from time to time) serve to encode relative position in (geometric or abstract) spaces.

This principle as stated is very general. One important possibility that it allows is for spatial-analogue representation to be mixed with other sorts of representation in the very same neural array. Such hybrid types of representation are a major focus of my project.

I view the range of possibilities allowed by Principle 4 as constituting a neurally-explicated capability for a form of *imagery.* I use the term "imagery" to link the work to studies of imagery by psychologists and philosophers [Refs. 3, 7, 10]. Imagery is held to confer processing advantages in various sorts of mundane cognition, including some types of problem solving, although there has been much controversy over the various claims made, and indeed over what imagery is in the first place. On the other hand, although I find it convenient to use the term "imagery" in talking about my model, I emphasize that I do not imply that human beings have any conscious access to the "images" I postulate. The images are simply a certain class of data structures, processsed in a certain way.

THE SPECIFIC MODEL

I now sketch a particular, detailed model that instantiates the data structuring principles presented in the previous section. A longer discussion is provided elsewhere [Ref. 2].

The role that this particular instantiation of the principles plays in the research should be carefully understood. I view it as just one of the many conceivable detailed models that conform to the general principles. It is this class of models that is my real interest. Thus, it is possible that the present detailed model may later be overthrown in favour of another. In any case, the present model contains several deliberate over-simplifications.

The instantiation takes the form of an information-processing model specified at a level slightly above that of detailed circuitry. That is, the model is composed of building blocks that can clearly be given a straightforward implementation in terms of idealized neurons that typically act like logic gates.

The model is based on a set of neural arrays of the sort described above. I henceforth call them *configuration matrices* (CMs). A short-term data structure is set up by putting appropriate CM elements into appropriate activity states (and letting the others be in a "null" or "resting" state). The elements used in a data structure need not all be in the same CM.

Structure per se consists partly in the relative positioning within individual CMs of the CM-element states. (See Principle 2.) This form of structure is backed up by a version of Principle 3: if two CM elements are in states that are similar in a certain way then they are taken to be identified with each

other; that is, certain processing mechanisms treat them as if they were the same element.

A CM is laid out on the cortical surface as a geometrical 2D array, at least approximately. Each of the CM elements composing a CM is connected to its eight neighbors, to enable certain basic processing operations. The fact that the theory proposes several CMs rather than just one larger one reflects a concern with physiological plausibility, but there is no space here to go into this matter.

The total set of configuration matrices is divided into a small set of "principal CMs" and a possibly large set of "storage CMs". The activity configurations (i.e. implemented data structures) in the principal CMs can be acted on by condition-action rules. These rules are implemented in neural circuitry that is for the most part outside the CMs themselves but is also dependent on connections between CM elements. Rules do such things as responding to, analyzing and modifying the data structures in principal CMs, copying them into and out of storage CMs, organizing transfers of information between CMs and long-term memory, and sending signals to mechanisms external to the CM model itself.

I emphasize that *the data structures in CMs are short-term.* They are constantly being modified, in the service of cognitive tasks such as language processing, commonsense problem-solving, planning and so on. The nature of long-term memory in the theory will only be investigated at a late stage in the project. (I certainly do not assume that a long-term encoding of a CM activity-configuration is itself a pattern of neural *activity*.)

The state of a CM element at any moment t is an ordered pair (B_t, H_t). The component B_t is one out of a finite set of values called "basic symbols". This set of possibilities is the same for every CM element. Any number of CM elements can simultaneously have the same B_t value.

Most basic symbols will, at least in the initial stages of the project, be neural activity patterns that are internal "names" for entities such as particular people, classes of objects (e.g. the class of all tables), classes of situations or events, etc. When a CM element holds such a basic symbol then we regard the element as (*currently*) representing the entity named by the symbol.

The H_t component of a CM element's state is an ordered tuple of ON/OFF flag values. For intuitive convenience, each tuple position is identified by the name of a color. When, say, the "red" position contains the value ON we say that the CM element is (*currently*) "highlighted in red", or just "red".

Data structures are created by placing particular basic symbols in various CM elements and/or by putting CM elements into particular highlighting states. The absolute positions of basic symbols and highlighting states in a CM are irrelevant in the case of data structures with no spatial-analogue aspect — only the relative postioning is significant. For the purposes of illustration, suppose there is a basic symbol that is the internal neural "name" for a particular person Mary. Let us refer to this basic symbol as MARY. Similarly, suppose there is a basic symbol JOHN for a particular person John, and a basic symbol LOVE that is the internal name for the class of possible loving situations. Then the statement that John loves Mary can be encoded by means of the CM sub-configuration depicted in Figure 1.

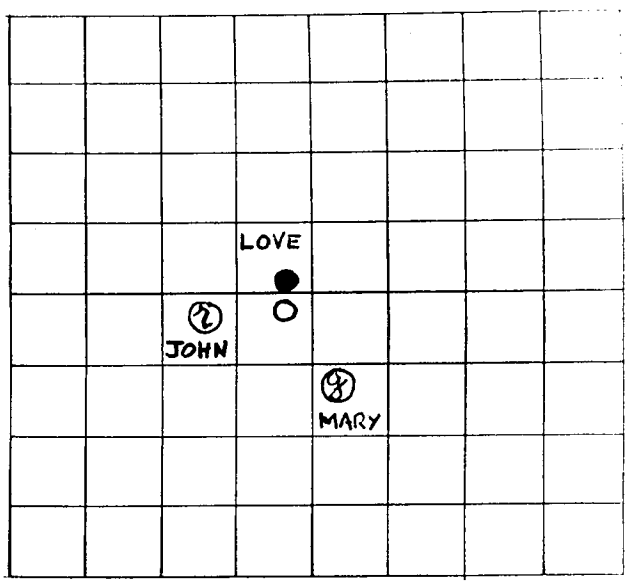

Figure 1: John loves Mary.

The figure is to be understood as follows. It illustrates an 8 by 8 region within a CM. Squares illustrate CM cells. (The illustration is schematic: CM cells are not meant to be geometrically square regions of cortex.) Capitalized words inside the squares illustrate basic symbols, the black disc illustrates highlighting in black, the empty circle illustrates highlighting in

white, and 'r' and 'g' in circles illustrate red and green highlighting respectively. Squares with no basic symbol shown illustrate cells containing a special NULL basic symbol that does not represent anything. Basic symbols JOHN and MARY represent the people John and Mary, and basic symbol LOVE represents the "love" situation-class. The highlighting serves to distinguish the roles of adjacent CM cells with respect to each other. White and black highlighting are confined to the special use illustrated here: a white CM cell represents a member of the class represented by the adjacent black CM cell. Therefore the white cell in the Figure represents an individual love situation, and is the "head" cell of the statement. In a love statement the agent and object cells (those containing JOHN and MARY here) must be highlighted in red and green respectively to indicate their particular roles.

The most important thing to notice about the example is the use of *adjacency* between CM elements as the "glue" sticking components of a data structure together. The particular relative positioning of CM states is unimportant except in so far as the appropriate adjacency relationships are achieved. Note that adjacency includes diagonal adjacency.

More complex data structures are built up in much the same way, as shown elsewhere [Ref. 2]. However, the representation method as so far presented suffers from an overcrowding problem. Most obviously, the method as it stands cannot handle an n-ary predication for n > 7, rather than a binary predication (using "loves", say). The overcrowding problem is avoided by the use of "unassigned symbols", a special type of basic symbol. An unassigned symbol does not permanently name anything. Rather, a CM element containing such a symbol can temporarily name something by virtue of that element's current role in a data structure. The crucial point is that if any two CM elements contain the same symbol, then they are in a sense identified with each other. Roughly this means that the two elements both currently represent the same thing. Therefore, data structures can be split up into sub-configurations.

The technique is illustrated in Figure 2, showing how the information that *John loves Mary or Mary loves John* could be encoded. In this figure the letters 'k' and 'l' indicate unassigned symbols, temporarily naming the hypothetical love situations. The OR subconfiguration uses those unassigned symbols, thereby referring to the same love situations. The 'p' in circles indicates purple highlighting.

The sharing of symbols by CM elements constitutes the model's instantiation of Principle 3 above. Two elements are in a "similar" state just when they contain the same basic symbol, and the "association" involving them consists in their being identified with each other. It is only fair to mention that similarity association, although perhaps conceptually elegant as a representational technique, introduces some extra complexity and inefficiency into the processing mechanisms.

Logical quantification can be handled straightforwardly, by means of activity configurations that are analogous to quantified formulae in first-order logic. However, I am not happy about proposing brain data structures that are similar in this way to logic formulae. I am interested in the possibility of including more ad hoc or naive quantificational machinery. I make one suggestion below.

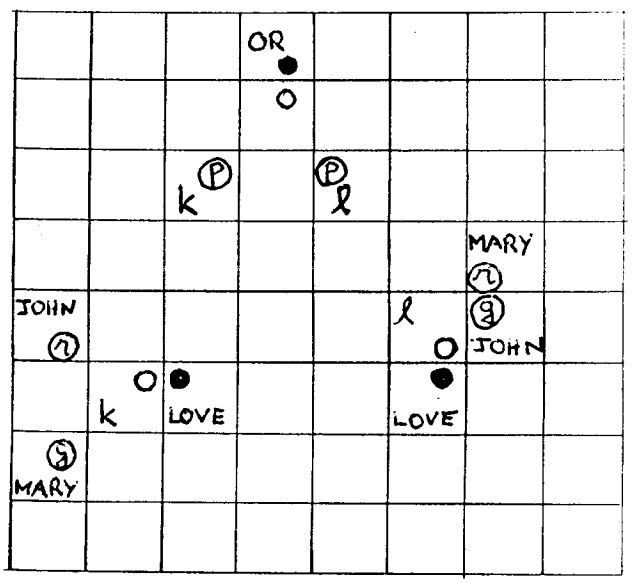

Figure 2: Structure splitting.

An Important Caveat

My interest is more on how the representational Principles enunciated above can be used to effect data structuring than in what the content of the data structures is. Thus, data structures and rules of the type illustrated in this section are merely being used for intuitive convenience. I am *not* committed to the

data structures and rules being at just this conceptual level in the brain. For instance, it may be that statements would be broken down into lower-level conceptual primitives. Nor am I committed to the (abstract-symbolic) data structures being closely analogous to logical formulae, as they are in the present model.

Some Parameters

The project takes CMs to be of size 32 by 32. This size fits well with my conjecture that each CM element is implemented in the brain as a "minicolumn" of cortex [Ref. 9]. Minicolumns are claimed by some neurophysiologists to be basic functional units of cortex. There are about 1000 in each of the larger columns of cortex (about 1mm wide) that also seem to be important functional components. Thus, a CM of size 32 by 32 fits nicely into a large column. A minicolumn contains about a hundred neurons, which is enough for my purposes, especially since a single neuron is itself capable of complex switching behavior. (Real neurons are much more complex than single logic gates.)

If there are basic symbols of the sort used in examples above, then there must be several thousand of them (and an accordingly large number of location matrices — see below — for each principal CM). The simulations in the project will use an artificially restricted set of basic symbols, numbering no more than a hundred or two.

I conjecture that there are only a handful of principal CMs in the brain, and the simulations will certainly not use more than a dozen or so. No a priori limit to the number of storage CMs will be imposed.

Spatial and Diagrammatic Imagery

Line-of-Sight Tasks — In accordance with the spatial-analogue Principle (number 4), I allow relative position of CM elements to be used temporarily to represent relative positions in geometric spaces or in abstract spaces. In particular, if both "spatial" axes in a CM are taken as analogues of geometric axes, then a CM configuration acts as a sort of "spatial image".

Figure 3 depicts an example of a spatial image. Such a data structure might be used in dealing with a task mentioned by Sloman [Ref. 12] — determining whether some adversary A can see desired goal object G, there being an obstacle that might block A's line of vision. In the Figure, the 'G' and 'A' indicate CM cells representing the goal object and the adversary. The cells could contain basic symbols representing those entities, or could be suitably qualified by abstract-symbolic structures in which the cells are embedded (though none are shown in the Figure). The large blob illustrates an irregularly-shaped group of orange-highlighted CM cells depicting the rough shape of the obstacle. The crosses indicate a group of gold-highlighted cells that are approximately in a straight line. This group depicts the line of sight. Notice the use of highlighting to differentiate the identities of various entities. All the cells illustrated are highlighted in blue as well. Certain processing rules take the relative positions of blue cells (and only blue cells) to indicate the relative spatial positions of the represented objects. The obstructing of the line of sight can be detected by a rule that responds to CM elements that are highlighted in both gold and orange.

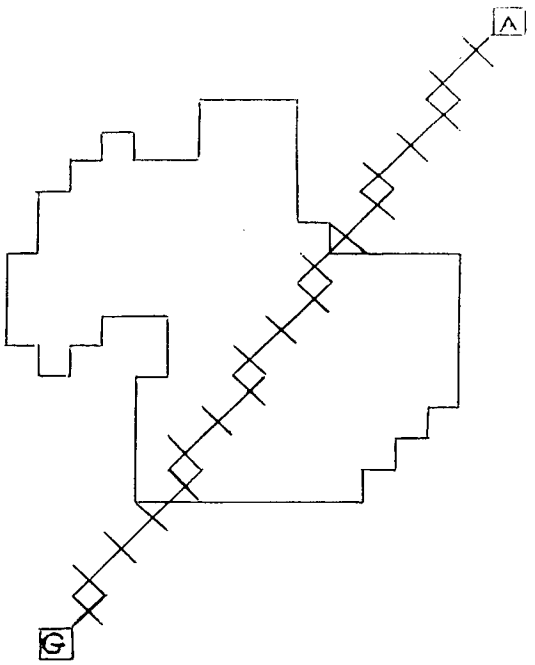

Figure 3: Line of sight.

The spatial problem-solving achievable with such images is obviously of an approximate nature, but is nevertheless heuristically useful. Elsewhere [Ref. 2] I give the details of the processing of such an image. Notice that the gold line represents a non-physical object, but itself has the same spatial quality as the blob representing the physical obstacle. This phenomenon is part of the "diagrammatic" quality of the system's imagery capability. Observe also the very important point that the image could include abstract statements about the entities involved. E.g. there

brain model than the standard logical forms of quantification are.

RULES

We now turn to the way the rule mechanisms work. Consider an inference rule that could be stated in English as:

if a man loves a woman then the man is hungry

and assume that this rule is realized as a body R of neural circuitry (mostly external to the CMs). Suppose there is a principal CM holding a data structure stating that man M loves a woman W. (Recall that a principal CM is a CM whose contained data structures can be manipulated by the rule mechanisms.) To put it briefly, R detects the data structure by means of suitable input taken direct from the CM and indirectly from it via various "location matrices" (LMs). The detection is achieved by circuitry that operates in parallel uniformly over the whole of each LM or CM concerned. R then sends signals to LMs and the CM that cause a piece of data structure stating the man is hungry to be inserted into the CM. Once again, the signalling operates in parallel over whole arrays. For instance, when signalling to the CM, R sends the same signal simultaneously to all CM elements. These then respond differently according largely to their state of highlighting and the highlighting states of their immediate neighbors.

In the following I discuss only the detection phase of rule firing. The crucial feature of this phase is the use of LMs. Consider a single principal CM. For each ordinary basic symbol B there is a location matrix LM_B associated with the CM. This is a 2D neural array isomorphic to the CM, although the LM elements themselves are not isomorphic to CM elements. For now we can assume that the state of an LM element is just a single flag value. We say that the LM element is on when the value of this flag is ON. The purpose of LM_B is to indicate the positions of those CM elements that contain the symbol B. Specifically, a element in LM_B is ON when and only when the corresponding CM element contains B.

If B is a symbol naming a *class* of entities, then there is a *member location matrix* MLM_B as well as the matrix LM_B. The task of MLM_B is to indicate the positions of *members* of the class. For instance, the basic symbol LOVE used earlier names the *class* of conceivable loving situations. Elements that are on in LM_{LOVE} indicate the positions in the CM of occurrences of the LOVE symbol itself. On the other hand, elements that are on in MLM_{LOVE} indicate the positions of elements in the CM which explicitly represent *individual* loving situations currently. The most important case of such a element is a "love-proposition head element" — that is, a white-highlighted element which is next to a black-highlighted element containing LOVE (cf. previous Figures). Another case is when a CM element contains an unassigned symbol that is also sitting in a love-proposition head element, for then the two elements represent the same loving situation.

Similarly, the task of MLM_{MEN} is to indicate the positions of CM elements that represent individual men. This works analogously to the LOVE case, since a white element next to a black element containing MEN represents some man. Of course, a CM element might contain the basic symbol JOHN, where John is a man. Thus, whenever an element in LM_{JOHN} is turned ON the corresponding element in MLM_{MEN} is also turned on (if it is not already on). Similarly, whenever a element in MLM_{MEN} is turned on the corresponding element in $MLM_{PERSONS}$ is also turned on.

The example rule mentioned above requires a further location matrix, to indicate the positions of CM elements representing man-loves-woman situations. This matrix is described elsewhere [Ref. 2]. Notice that it is needed *only* because some rule needs to detect man-loves-woman situations. *No location matrix is necessary unless it forms part of the machinery for some rule.*

Similarity-association or symbol-sharing is given computational flesh by LMs. The basic idea is that whenever element (x,y) of a given LM is ON and the corresponding element of the CM contains a non-null symbol B, we want the system to turn ON every other element (x′,y′) (in that LM) such that element (x′,y′) in the CM also contains B. This effect is just what is required for symbol-sharing to be taken as CM-element identification. In fact, to facilitate the process I take each LM element to contain the basic symbol that is in the corresponding CM element. The required spread of ONness in an LM is achieved by each ON element in the LM broadcasting its symbol to all other elements in the LM, and every element then turning ON if its own symbol is the same as the one broadcast.

Without special assumptions this broadcasting must be done separately for every different symbol represented in the LM, but if in fact the symbols are nearly-orthogonal vectors of neural activity then it

might be a statement to the effect that the adversary is afraid of "us" (the agent doing the problem solving). This ability to mix spatial and non-spatial data structures is another aspect of the diagrammatic quality of the system.

The rules that operate on the spatial image in the line-of-sight task rely predominantly on primitive signalling mechanisms that are much the same as those required in any case by non-spatial-analogue, abstract-symbolic processing. This point is an important aspect of the fact that the theory puts spatial imagery and abstract-symbolic processing into a unified framework.

Johnson-Laird's Mental Models — Johnson-Laird has devoted much attention to human reasoning about physical arrangements of objects [Ref. 5]. In a simple case, a subject is told that the fork is to the left of the spoon and that the knife is to the left of the fork, and the subject must infer or verify that the knife is to the left of the spoon. Johnson-Laird maintains that the subject constructs not only some internal propositional representations of the given pieces of information (these representations being of a form close to the linguistic statements themselves) but also a "spatial mental model" of the situation. This model consists of some sort of internal map-like representation of space: tokens standing for the various objects in the task are placed in particular positions in the map, the positions being consistent with what the propositional representations say. The desired relationships can then be read off the map directly.

The present project will provide a rather direct neural implementation of Johnson-Laird's models. That is, in a CM being used as a space map, elements can be made to represent particular objects by means of the data structuring techniques alluded to in the previous section. These elements would be in relative positions in the CM that correspond to the relative positions of the object relationships as specified in the task statement. See Figure 4. The three white cells represent a knife, a fork and a spoon. These cells are also blue, to indicate that their relative positions have spatial significance. For illustrative convenience, leftness/rightness in the CM illustration in the Figure corresponds to leftness/rightness in physical space.

In fact, an important feature of Johnson-Laird's total theory is that mental models are constructed from propositional (abstract-symbolic) representations that state the object relationships in the case at hand. His propositional representations would have an analogue in my system, in the form of abstract-symbolic data structures in various CMs. "A is to the left of B" can of course be represented in the system in a way similar to the way "John loves Mary" is represented. These data structures will be used as a basis for constructing the diagrammatic spatial images that display the relative positions. The construction will itself be done by rules in the rules component of the model.

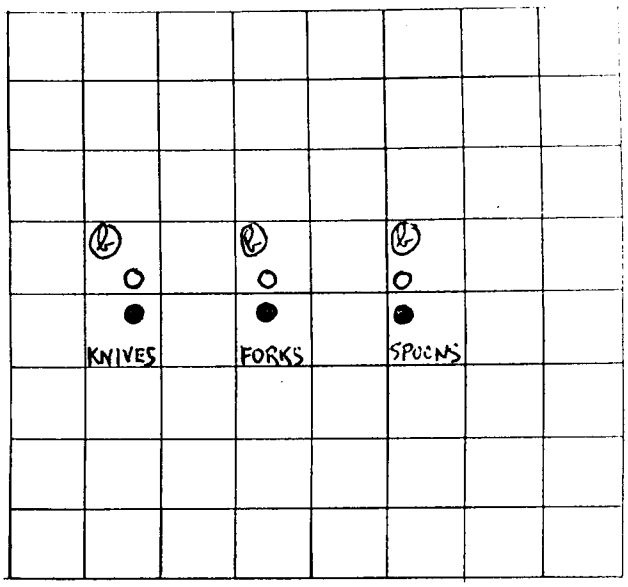

Figure 4: An implemented mental model.

For various reasons, Johnson-Laird suggests that some spatial mental models may contain simple symbolic devices of a propositional nature as well as the devices already illustrated. The CM theory is well set up to handle this sort of mixture, since it is already geared to hybrid, diagrammatic imagery as well as simple spatial imagery and purely propositional data structures.

The project will give precise computational substance to another major aspect of Johnson-Laird's ideas. This is the issue of syllogistic reasoning. Johnson-Laird proposes the use of another class of mental models to cope with such reasoning. This class is easily encompassed in my model. The syllogistic mental models involve certain restricted forms of quantification that are arguably more plausible for a

turns out that the treatment of all the different symbols can proceed in parallel. I am investigating means for achieving this near-orthogonality.

Interpreted Rules

There is a major disadvantage to the hardware implementation of rules of the sort assumed above. The rules themselves are not data structures that can be inspected and manipulated (in the service, say, of rule learning and modification). A major task of the project will be to investigate the possibility of rules in the form of data structures. That is, to continue the example used earlier, the rule "if a man loves a woman then the man is hungry" would be a data structure of some form in a CM. The hardware rule mechanism would then consist partially or wholly of rules that act together as an *interpreter* of the data structure rules.

The ideas on interpreted rules are less fully developed than are other aspects of the model. A detailed syntax of data structure rules remains to be fully defined, and the nature of the pattern matching process required for rule enabling is not yet completely worked out. I anticipate that this process will operate largely by the flow of simple signals among CM elements.

MPP SIMULATION

An MPP Pascal simulation program for a small pilot version of the model is running. In this version there is just one CM, 20 basic symbols (the location-unspecific neural activity patterns that can be instantiated in CM elements), 5 highlighting colours, 2 very simple rules, and 22 LMs (the location matrices subserving the rules). Later simulations of the model will involve a hundred or two basic symbols, a dozen or so highlighting colours, a few principal CMs, dozens of storage CMs, dozens of rules per principal CM, and several hundred LMs per principal CM.

The current level of simulation is above that of detailed neural circuitry, so that for instance a basic symbol is simply simulated at present by a integer value. The highlighting flags at each CM element are simulated by boolean values. The ON/OFF state of an LM element is also simulated by a boolean value.

Accordingly, the CM is simulated by (1) a parallel array of integers (in a subrange) and (2) for each highlighting colour, a parallel array of booleans. Each LM is simulated by a parallel array of booleans. All these arrays are simultaneously present in the ARU. Since in the model each CM and LM is of size only 32 by 32, only parallel subscripts in the range 0..31 have significance currently.

After the initial state of the CM has been set up from input data, the simulation proceeds cyclically. Each cycle consists of (a) redefining the state of every LM, (b) firing one rule (if any can be fired at all), thus usually altering the state of the CM. The cycling stops when either no rule can fire or an input-specified number of cycles has been performed.

The neural model itself relies mostly on uniform, local, parallel operations over any given matrix (i.e. a given operation involves all elements of a matrix simultaneously, the processing at the elements varying only according to their own current states and their neighbors' current states). Therefore, most of the programming of the LM-state updates and rule-firing is straightforward, and makes good use of the ARU's SIMD nature and of MPP Pascal's parallel-array features. A certain amount of array shifting is needed in those cases when the processing at a matrix element brings in the neighboring elements.

In later model versions, there will be several CMs, and more than one will be "principal" (i.e. associated with rule machinery). Since there will be significantly many periods during the model's processing when various different principal CMs (and attendant LMs) are being similarly processed, and since it is unlikely that any model version will contain more than 16 principal CMs, it will be convenient to simulate the principal CMs by using different 32 by 32 portions of the ARU (with the LMs attached to a given CM using the same ARU portion as that CM). Thus various principal CMs will be able to be operated on simultaneously. However, the arrangement will complicate the programming of the frequently needed copying of the contents of one principal CM into another principal CM.

Storage CMs, on the other hand, can be conveniently stacked vertically beneath principal CMs, as there would be little benefit in arranging them horizontally across the CM, and they do not have any associated LMs.

LM Paging

In later versions of the model, each LM element will have a few flags in addition to the single on/off state mentioned earlier. These flags and the state will be simulated by booleans. Now, in the model, each LM element will also contain a copy of the basic symbol in the corresponding element of the CM to which

that LM is associated. However, it turns out that my current hypotheses about the use of these symbol copies are such that there will be no need to carry the duplication over into the simulation itself. Thus, it will be adequate to go on implementing each LM as a small set of boolean parallel arrays. Nevertheless, since large versions of the model will contain several hundred LMs for each principal CM, the memory capacity of the PEs in the ARU will be too small to hold all the LMs. A scheme will be needed for paging LMs to and from the staging memory.

A Snag

To oversimplify a little, I would have liked to use the following piece of program to do some basic LM updating. (Type **symbols** is a subrange of **LMids**, which is itself a subrange of integer.)

```
(* CM_symbols:  parallel array[0..127,0..127]
                          of symbols;
   LM: parallel array[LMids,0..127,0..127]
                 of boolean;
*)

where   CM_symbols <> null_symbol
   do   LM[CM_symbols]:= true
```

That is, the desired action was:

for each i,j where `CM_symbols[i,j] <> null_symbol`,

set `LM[CM_symbols[i,j], i,j]` to `true`.

Unfortunately, albeit understandably, neither the hardware nor MPP Pascal supports this "indirect addressing" at PEs. Therefore, I have had to resort to a loop that cycles through the (non-null) symbols, using a control variable s, doing

```
LM[s]:= (CM_symbols = s)
```

at each stage. This loss of parallelism will not have a drastic effect on my own simulation work, but the phenomenon is of some general interest. Goodyear Aerospace's Dr. Ken Batcher, designer of the MPP, suggested indirect addressing at PEs as an important addition in future versions of the machine, in his invited speech during the Symposium. It is also noteworthy that he appealed specifically to artificial intelligence as an application domain that might benefit from indirect addressing.

REFERENCES

1. Barnden, J.A. On short-term information-processing in connectionist theories. *Cognition and Brain Theory, 7* (1), 1984.
2. Barnden, J.A. Diagrammatic short-term information processing by neural mechanisms. *Cognition and Brain Theory, 7* (3&4), 285–328, 1984.
3. Block, N. (Ed). *Imagery.* Cambridge, MA: MIT Press, 1981.
4. Hinton, G.E. & Anderson, J.A. (eds) *Parallel models of associative memory.* Hillsdale, NJ: Lawrence Erlbaum, 1981.
5. Johnson-Laird, P.N. *Mental models.* Harvard University Press: Cambridge, Mass. 1983.
6. Kohonen, T. *Self-organization and associative memory.* Springer-Verlag: Berlin, 1984.
7. Kolers, P.A. Perception and representation. *Ann. Revw. Psychol. 34,* 129–66, 1983.
8. McClelland, J.L., Rumelhart, D.E. and the PDP Research Group. *Parallel distributed processing, Vol. 1: Foundations.* MIT Press: Cambridge, Mass. 1986.
9. Mountcastle, V.B. An organizing principle for cerebral function: the unit module and the distributed system. In G.M. Edelman & V.B. Mountcastle, *The mindful brain.* MIT Press: Cambridge, Mass. 1978.
10. Richardson, J.T.E. *Mental imagery and human memory.* St. Martin's Press: NY. 1980.
11. Rumelhart, D.E., McClelland, J.L. and the PDP Research Group. *Parallel distributed processing, Vol. 2: Psychological and biological models.* MIT Press: Cambridge, Mass. 1986.
12. Sloman, A. Image interpretation: the way ahead? In O.J. Braddick & A.C. Sleigh (eds), *Physical and biological processing of images.* Springer-Verlag: Berlin, 1983.
13. Touretzky, D.S. & Hinton, G.E. Symbols among the neurons: details of a connectionist inference architecture. *Procs. 9th Int. Joint Conf. on Artificial Intelligence,* 1985.
14. Papers in *Cognitive Science, 9* (1), 1985: special issue on connectionism.