Coward, L.A. (2001). The Recommendation Architecture: lessons from the design of large scale electronic systems for cognitive science. Journal of Cognitive Systems Research 2(2), 111-156.

The Recommendation Architecture: Lessons from Large-Scale Electronic Systems Applied to Cognition

L. Andrew Coward [1]
School of Information Technology, Murdoch University,
Perth, Western Australia 6150

Abstract

A fundamental approach of cognitive science is to understand cognitive systems by separating them into modules. Theoretical reasons are described which force any system which learns to perform a complex combination of real time functions into a modular architecture. Constraints on the way modules divide up functionality are also described. The architecture of such systems, including biological systems, is constrained into a form called the recommendation architecture, with a primary separation between clustering and competition. Clustering is a modular hierarchy which manages the interactions between functions on the basis of detection of functionally ambiguous repetition. Change to previously detected repetitions is limited in order to maintain a meaningful, although partially ambiguous context for all modules which make use of the previously defined repetitions. Competition interprets the repetition conditions detected by clustering as a range of alternative behavioural recommendations, and uses consequence feedback to learn to select the most appropriate recommendation. The requirements imposed by functional complexity result in very specific structures and processes which resemble those of brains. The design of an implemented electronic version of the recommendation architecture is described, and it is demonstrated that the system can heuristically define its own functionality, and learn without disrupting earlier learning. The recommendation architecture is compared with a range of alternative cognitive architectural proposals, and the conclusion reached that it has substantial potential both for understanding brains and for designing systems to perform cognitive functions.

Keywords: Cognitive architectures; Computational models; Information context; Functional modules

1 Introduction

There have been numerous attempts to understand the architecture of cognitive systems as assemblies of functional subsystems. In fact, "A fundamental view in cognitive science is that an intelligent system is not completely homogeneous. It must consist of a number of functional subsystems, or modules, that cooperate to achieve intelligent information processing and behaviour" [36].

However, theoretical grounds have not in general been offered for the existence of such modules. Rather, the existence of modules has been deduced on the basis of cognitive psychology observation, the specialized regions found in brains, and experience attempting to design intelligent machines. Architectural proposals have focused on identifying high level modules, describing how systems could organize experience, how they could internally represent external conditions, or how they could associate experiences with behaviours.

Suggested modules at the highest level have included memory, thinking, attention, learning, and motor control [36]. At a somewhat more detailed level, memory has been separated into submodules like implicit memory, explicit memory, and working memory, motor control has been separated into automatic and controlled processes.

Proposals for the organization of experience have included mental models [24] and feature maps [38]. Connectionist proposals in this domain have included Hopfield Nets [23] and multilayer perceptrons [30]. Connectionist networks which detect subfeatures and organize them into features in a hierarchical fashion have been discussed [1].

Attempts to understand the internal representation of external conditions have ranged from various symbolic systems [e.g. 35] to systems using coarse coding at the input level penetrating some distance into the system [e.g. 7].

1 E-mail address Landrewcoward@shaw.ca
http://wwwstaff.murdoch.edu.au/~acoward

Ways to associate experience with behaviour have included production systems [28] as well as some of the proposals mentioned earlier for the organization and representation of experience.

Over the last 30 years there have been major advances in the technology for design of electronic systems to perform very complex combinations of functions. This design experience makes it possible to draw conclusions about architectural constraints which apply to any functionally complex system, including brains [13]. These architectural constraints include strong theoretical reasons why functionality must be separated into modules, and there are also very severe constraints on the way modules divide up functionality and on the way modules interact.

In all commercial electronic systems, functionality is defined by design. Heuristic definition of functionality (such as self modifying code) has never been successfully implemented. It is possible to demonstrate that the von Neumann architecture which is ubiquitous in commercial systems is one architectural form which can be consistent with functional complexity, but that this form makes heuristic definition of functionality impractical on theoretical grounds. The only architectural form in which both functional complexity and heuristic definition of functionality is possible is the form called the recommendation architecture. Only certain types of modular separations and certain types of relationships between modules are possible within this recommendation architecture.

Coward [8] originally described the recommendation architecture and pointed out that many psychological and physiological phenomena could be understood as phenomena to be expected in systems with the recommendation architecture. Coward has further argued that the recommendation architecture can be the basis for understanding all cognitive phenomena in terms of physiology [10] up to and including human consciousness [11,12].

The theoretical reasons why systems which learn complex functionality are forced to adopt the recommendation architecture were first articulated in [9], and developed more fully in [13]. The current paper develops these reasons further and attempts to express them in the language of cognitive science.

Simple simulations of systems with the recommendation architecture demonstrated that memory phenomena analogous with biological memory occurred in such systems [9]. A more technically detailed recommendation architecture was described in [13]. Simulations of portions of this architecture have been described [13, 14, 20]. The current paper describes in detail an implementation of the major subsystems of the architecture described in [13] and presents some of the results of its application to two problem domains with qualitative similarity to the problem domains experienced by brains. One problem is a simple categorization problem and the other is a more complex problem involving recognizing objects, recognizing and generating speech, and associatively imagining objects. The results obtained will be described and compared with cognitive phenomena. In particular it is demonstrated that the system has the ability to learn incrementally: later learning does not disrupt earlier learning. Finally, the recommendation architecture is contrasted with various alternative architectural proposals.

2 Functional complexity and functional architectures

Some current commercial electronic systems [3, 34] have thousands of millions of hardware components like transistors. The software for such systems can exceed twenty million lines of code, and the corresponding load occupy over a thousand million bits of memory. In operation there can be tens of millions of millions of transistor changes of state per second. Some of the most complex systems are real-time systems which have thousands of interacting features and use input data to control a physical system, such as a telecommunications network or a chemical manufacturing plant, with no human intervention and with severe time constraints within which it must respond. In such a system a response will depend upon what has previously happened, and a timely response depends upon many processing tasks which must be carried out concurrently. The problem in real time system design is to partition the allowed end-to-end elapse time from external event to the deadline for system action between the many modules which may require both time to generate their individual outputs and also input information from other modules which themselves require time to generate that information.

As the complexity of electronic systems increased, it became clear in practice that an architecture that provided a multilevel descriptive hierarchy was required [34], and that in the absence of such a hierarchy it was extremely difficult to achieve integration of different functional modules [19]. A critical aspect of functional integration is provision of a context for information exchange between modules. In a commercial electronic system modules must share enough context for the information to be acted on unambiguously. A major problem in integrating modules designed for different systems is the lack of common information contexts [19].

Coward [8, 9] pointed out that there are specific system requirements which give rise to these and other architectural constraints. The need for the system be constructable from design information using some set of construction processes leads to the requirement that the system be made up of large numbers of devices which are all slight variations of one of a relatively small portfolio of qualitatively different device types. The need to repair the system and to modify some of its functionality in such a way that no other functionality is affected leads to the requirement that system functionality be organized into a simple functional architecture. In a functional architecture the functionality of the system is separated into modules, each of these modules into submodules, all the way down to the detailed operational components. In a simple functional architecture there are two further constraints on how

functionality is separated into modules. One constraint is that functionality must be separated in such a way that all the modules on one level are roughly equal in terms of the number of the device (or software assembly code) operations which occur long term in support of system functionality within each module. The second constraint is that functionality must be separated in such a way that the information exchange between modules required to coordinate their functionality is minimized.

The reason for the limits on number of devices is that there are advantages to a smaller volume of design information and a smaller number of different construction processes (which must also be specified by design).

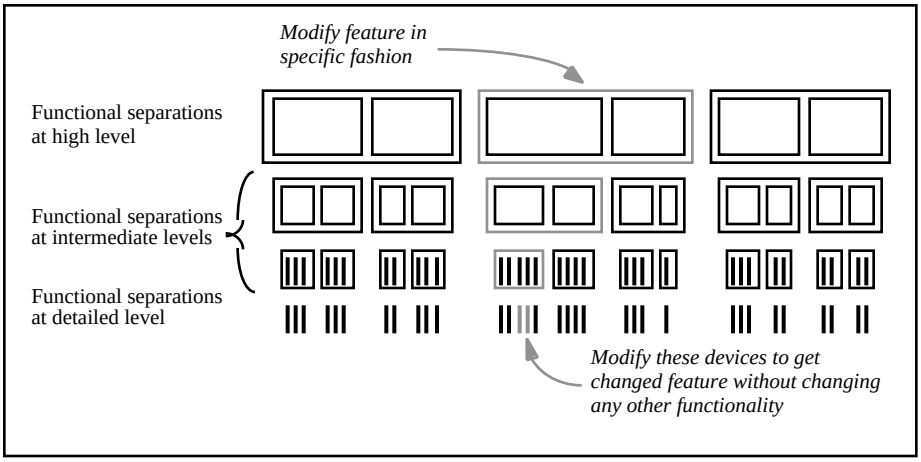

Figure 1. Logical paths through a functional architecture

The reason for the constraints on modules derives from the repairability and modifiability requirements. If there is a system failure, the initially available knowledge is at relatively high level: some aspect of the system does not operate. However, the action required is a corrective action to one or a few detailed components. Logical paths must therefore exist through the functional architecture from high level definitions of a deficit to a level of detail at which corrective action is possible, as illustrated in figure 1. If modules were very disparate in size as defined, most paths would pass through the largest module and little discrimination would result. If there was very large information exchange between two modules it would in general be very difficult to determine which was the relevant module for tracing the deficit condition. The requirement for modifiability imposes the same constraints: it is necessary to derive from a relatively high level definition of the desired functionality the detailed operational component changes which will implement the desired change without undesirable side effects.

It is important to note that although brains are not the result of an intellect driven design process, they do need to be constructable from DNA information, they do need to modify their own functionality or learn, and they need to recover from construction errors, failures and damage. These needs mean that it must be possible for a brain to use the available information on the problem condition to narrow the range of corrective actions. For example, it must be possible to use knowledge that “something is going wrong when a particular type of behaviour is used under certain conditions” to limit the physiological domains within which experimental changes to improve behaviour will be made. Hence similar constraints apply to such systems.

The ideal module separation for operational equality will not necessarily be the same as the ideal separation for minimized information exchange. The design process for electronic systems is therefore one in which functionality is initially divided into roughly equal modules, then the resultant information exchange determined and changes made to reduce the exchange. These changes may in turn force a different separation to preserve rough equality until a satisfactory compromise is found. This process of finding a compromise continues down the modular hierarchy, with the need for compromise at a detailed level sometimes forcing changes to the compromise at a higher level [12]. The result of this compromise process is a hierarchy of modules in which the functionality of a module cannot be related in any simple fashion to overall system functionality. One module may participate in many system level features in a complex fashion. For example, in a telecommunications switch providing telephone service to a group of users there are major functional categories like call processing (connecting phones together), diagnostics (detecting and indicating problem conditions), maintenance (changing the services available to different customers) and billing (generating the data which allows accurate charging for services). Even at this high level, some of the functions which would intuitively be labeled “billing” will be performed by call processing in order to minimize information exchange [12]. At a more detailed level the subset of system functionality performed by one module may be extremely complex to describe in terms of system features. Note that this is not in conflict with the requirement for

logical paths, it is only that the logical paths for different aspects of many features may pass through the same module. In the case of brains, if modules are created on the basis of rough equality and minimized information exchange, they are unlikely to correspond exactly with functions with simple cognitive labels. Two system operations which are generally similar in cognitive terms may as a result of subtle differences require the activity of different sets of modules.

3 Context for information exchange

There is an additional critical requirement that any module which receives information from another module has a context for that information. The statement that there is information exchange between two modules means that when the source module generates information of the exchanged type, that information is in a position to influence the behaviour of the target module. The module may of course choose to ignore the information based on its current state or other information received, but ignoring the information is an active behaviour of the target module, the information is always presented if generated. Any such information must have some meaning to the receiving module.

For example, if there is to be verbal information exchange between humans, unless the speaker uses a language understood by the hearer, no verbal meaning is received. Of course, in the case of a human being, some meaning is derived (for example, the speaker can speak a foreign language, the speaker believes the hearer understands that language etc.) but the meaning is much less than the speaker intended. In an electronic system the context depends on the level of the source and target modules. At a very detailed level, a receiving module may "know" that a 1.2 volt signal within a specific time interval indicates a 1 bit and that a signal less that 0.2 volts in the same time interval indicates a 0 bit. If at this level a source module began to generate 0.7 volt signals, or signals out of synchronization with the appropriate time intervals, the receiving module would have no context for the outputs and could take no appropriate action, even if the outputs were being generated in a fashion determined by the inputs to the source module. A higher level module may "know" that a signal such as 11010010 indicates a logical address. A yet higher level module may "know" that a logical address received from another specific module indicates the physical location of an available hardware resource of a particular type. A yet higher level module may "know" that a hardware resource of a particular type can be used to complete a necessary system operation. “Knowledge” in all cases is the ability to derive from the input an output which is appropriate to the functional role of the module. A module on any level may select a variety of actual outputs based on its current and past inputs.

In commercial electronic systems, the context for all information exchange is functionally unambiguous. Functionally unambiguous means that individual module outputs on any level can be interpreted as commands for the system to perform some action. For example, the software for a system is written in the form of instructions, and a single corrupted voltage representing a bit in the system will in general cause the system to crash. At a higher level, there are hierarchies of information, with global information for which a context exists in every module in the system, and local information for which a context only exists within one module and its submodules.

There are several consequences to the use of unambiguous contexts for information exchange between modules at higher levels. One is that because information may be used by many different modules, there is a need for a reference location at which any module can find the current, unambiguous value of any needed information. Hence the ubiquity of memory subsystems. A second is that if one module is potentially changing some element of information, that information is ambiguous for any other module. As a result, only one module can be operating on an element of information at any one time, and operations by different modules tend to be performed sequentially by a single processor subsystem. Parallel processing requires that system information be partitioned into orthogonal segments which can be operated on by different modules. In a functionally complex system this partitioning must be dynamic and is difficult to manage. Thus the maintenance of a functionally unambiguous context for information exchange results in the familiar memory, processing and sequential operation of the von Neumann architecture. It is in fact a very severe constraint to insist on an invariably unambiguous context but it makes it possible in electronic systems to ensure that exactly the specified functionality will be performed. Every module in such systems at every level has a portfolio of possible input conditions and a corresponding response specified by design. If a module receives an input which is outside its portfolio, the system will in general cease to operate in any useful fashion.

There is a further consequence of the use of functionally unambiguous contexts, which is extreme difficulty in heuristic definition of functionality. If the system heuristically defines its own functionality, some modules at least must change the inputs they receive and the outputs they generate in response to their inputs. However, in a functionally complex system there will be a very complex pattern of information exchange between modules. When a module changes its outputs, it would be extremely difficult for all the modules which have chosen to receive that output to establish an unambiguous context for that output in their own functional terms when the source module made the decision to generate the output without knowledge of the functionality of all the modules which might choose to use its output.

However, even if it is not unambiguous, information exchange cannot be meaningless, because the definition of information exchange is that it can influence the behaviour of the target. In other words, a receiving module ultimately receives some electrical, chemical or other input and it must be able to derive from that physical input some meaning related to its functional objectives. If the input cannot be functionally unambiguous it must possess some partial, ambiguous meaning.

To again give a human level illustration, suppose that the understanding exists that if I leave a message for my wife asking her to call me at the office, something disastrous has happened. If one day I decide I would just like to hear her voice and leave the message, she will take action as if there has been some disaster. In this scenario, the leaving of the message is functionally unambiguous, it will always result in disaster response behaviour. As long as the message is functionally unambiguous it leaves no room for learning new behaviours. However, if partially ambiguous contexts are maintained, additional meaning may be derived from the tone of my voice which results in a different behaviour, but the communication channel with my wife must make it possible to derive additional meaning. Furthermore, even if a partially ambiguous context is allowed, there are restrictions on both the degree to which messages can be changed while retaining the expectation of an appropriate response and the changes I can make to the meaning of exactly the same message. Thus, I cannot leave the message in a foreign language and expect a call, and I cannot leave the usual message to call me at the office and expect that I will called at the new number which I have just been assigned. So the critical questions in any system in which partially ambiguous information is exchanged is how much can the conditions under which the same message are generated be changed, and how much can the form of the message generated under the same conditions be changed without destroying the meaning of the message to the recipients?

Analogous context issues apply between modules within a system. As an example, consider a notional arithmetic calculator with three major modules. One interprets input from a keyboard as a pair of two digit numbers, for example 23 and 47. The second module calculates the product of the two numbers, in the example it would be 1081. The third module determines the pixels on the screen which will display that product. Within the second module there are four submodules. One submodule separates one of the numbers (say, 23) into a tens part and a units part and communicates one part to each of submodules two and three. These second and third submodules multiply the parts of one number with the other number and generate partial products (in the example, 20 x 47 = 940 and 3 x 47 = 141). The fourth submodule adds the two partial products and sends the result to module 3. In this case all the information communicated between modules or submodules is completely unambiguous. However, consider the partial products (i.e. 940 and 141) output by submodules 2 and 3. If they were communicated outside the second module they would be ambiguous but not necessarily meaningless. The presence of a given partial product occurs on average in about 0.5% of all possible products of pairs of two digit numbers. Hence the presence of a partial product within a certain range could communicate some meaning limiting the range of possible answers, provided the receiving module has a context for the information which allowed it to make use of the input. For example, if a module detected and indicated the presence of a partial product between 87 and 94, this would limit the system action to less than ~4% of all possible answers for two digit numbers. It is possible to imagine a system in which different modules selected various ranges of partial products and indicated the presence of the condition they had selected when it was present. Some other module could potentially use correct/incorrect feedback to construct high integrity answers out of combinations of these ambiguous signals.

This is not a particularly attractive architecture for a calculator, but it brings out a number of points. Firstly, modules exchanging partially ambiguous information detect the repetition of information conditions such as the presence of a partial product within a particular range. Consistent detection of the information condition by the same module is essential for meaningful system responses. Secondly, the definition of partially ambiguous is that it provides a probabilistic meaning to recipients. Thus the presence of a particular partial product means that there are a set of answers that are not possible and another set which are possible. If the system contains enough modules detecting and indicating a wide enough range of different information conditions there can be enough information to discriminate between all the alternative system behaviours. This use of information is very similar to coarse coding [5]. The third point is that many modules might become reliant on the output of any one module. For example, the presence of one partial product could be relevant to the indication of many different final products. The issue with maintaining context when modules can define and change their own functionality and partially ambiguous information is exchanged is thus that a module may select a condition (such as the presence of a partial product within a specific range) to detect, and many targets may choose to use the output of that module for different functional purposes. If it later changed the condition to which it responded, it would be hard to determine the implication of the change for all the different recipients. Such changes, for example extending the range of partial products detected, must be managed in such a way that a sufficiently meaningful context is maintained for all recipients.

In a system performing a complex combination of functions, modules are needed which detect information combinations of many different levels of complexity. The functional architecture is therefore a modular hierarchy in

which the simplest devices detect the repetition of some portfolio of conditions, modules detect information conditions which are the repetition of significant subsets of their constituent devices, yet higher modules detect conditions which are the repetition of significant subsets of their submodule conditions and so on. If information exchange were unambiguous, this would be a feature, category, supercategory hierarchy, but because information exchange is partially ambiguous the modules on every level detect information conditions which only correlate partially with functionally unambiguous conditions.

For heuristic definition of functionality, application of consequence feedback is essential but such application raises a context issue. As discussed earlier, because functionality must be separated into modules on the basis of rough equality and minimized information exchange, in a functionally complex system the relationship between module functionality and system features will in general be very complex, with one module contributing to a large number of such features. If consequence feedback is applied to such a module on the basis of performance of one feature, changes resulting from the feedback will result in unpredictable, almost certainly undesirable changes in other features. It is extremely unlikely that a change which improves the performance of all features affected by the module exists. In other words, context for information exchange cannot be maintained if consequence feedback is applied. Consequence feedback can therefore only be applied to a separate subsystem which interprets the outputs of the functional architecture into behaviour. This separate subsystem cannot itself be functionally complex because this would mean that it also had the same context management problem. It can only be given a portfolio of simple actions and use consequence feedback to associate different functional architecture outputs with different actions. However, because those functional architecture outputs could become associated with complex sequences of the simple actions, the overall system behaviour could be very complex.

There is thus a major separation in the architecture of any system which learns to perform a complex combination of functionality between a functional architecture, called clustering, which is a hierarchy of partially ambiguous repetition detection, and a subsystem called competition, which uses consequence feedback to associate the outputs of the functional architecture with a portfolio of behaviours. The outputs of the functional architecture can be interpreted as recommendations for different system behaviours, with one behaviour being selected by the competition subsystem. The system is therefore called the recommendation architecture in contrast with the instruction based von Neumann architecture.

There is still an issue of context maintenance even within the functional architecture. Any module output may be distributed widely to many other modules, but the distribution of any specific output is determined heuristically at the individual receiving module level. Once a module decides to produce an output in response to some information condition, it has limited control over what other modules may decide to use that input or how they use it. If it later changed the output it produced in response to a repetition of the same condition or produced the output in response to a different condition, it would be difficult for it to communicate the meaning of the change to all modules which might be making use of its output, since it cannot know what the output means to those modules in their own functional terms.

Because the meaning of an output cannot be specified by design, there are only three sources of information which can provide context to a receiving module. One is the identity of the module from which the input comes, in other words outputs from the same module have a similarity in some information sense. The second is any structure in a specific module output. At the device level such structure could be the rate of firing within the output. At a higher module level it could be the identities of the devices within the module which were the module output. Such structure could provide some information on where in the range of conditions to which the module responds the current condition is located. In the calculator example this kind of information could indicate whether the current partial product was in the low, middle or high part of the range. The third source of information is any global signals (i.e. signals available to all of an intercommunicating set of modules, affecting them all in the same way) which change the meaning of all outputs in some consistent fashion. For example, in the calculator example the detection of a plus sign in the input could inhibit all the modules detecting partial product ranges from generating outputs, essentially turning off multiplication if addition were indicated.

The critical context issue in a system which heuristically defines its own functionality is therefore how modules can modify their portfolio of conditions without damaging the meaning which other modules derive from their earlier outputs. The most conservative solution to this problem is to require that if a module once produces an output in response to an information combination it will always produce the same output in response to any repetition of the condition and no output in response to any other condition. This degree of restriction would be impractical for a real system, and the requirement is therefore to identify ways to allow (severely limited) changes. These ways are discussed below.

4 Device level implications of information context

Even at the device level, an exchange of signals must in general have some meaning related to the objectives of the system. It is not feasible that meaningless information (i.e. random noise) at device level be somehow added

together to generate meaning at a higher level. It is, however, feasible that partially ambiguous information at device level add to form less ambiguous information at a higher level. This is the essential point of coarse coding, such as the colour discrimination derived from retinal inputs with very broad but profiled frequency responses. The question is not whether device outputs have meaning or not, but rather how much meaning must there be for the system to be viable and how is that meaning created and preserved for the recipient. The requirement to sustain at least the minimum degree of meaning to ensure system viability then becomes a severe system constraint.

In a functionally complex system which learns, many devices may choose to use the output of any one given device output. These target devices will be using the output for many different functional purposes which are unknown to the source device. In neural networks, device algorithms involve changes to the relative weights of inputs. Such neural networks appear to suffer from catastrophic interference by later learning with earlier learning, which completely disrupts or erases that earlier learning. French [18] has reviewed this problem and the various approaches to avoiding it. One approach is sparse coding of internal representations implemented in a range of ways, but such sparse coding creates problems when a large number of patterns must be learned. A second approach is to limit learning to domains which have a high degree of internal structure, and early on present examples of all possible pattern types. However, this approach limits learning to the pattern types presented early on. A third approach is to restrict the receptive fields of inputs, which in the extreme makes the network localist. However, as receptive fields expand to allow more distributed representations, interference increases.

The contention of this author is that the problems with catastrophic interference are symptomatic of inadequate management of the (partially ambiguous) context for information exchange between devices. When the input weights to one device are changed, the meaning of the output of that device is changed for all its targets. Because each of these targets has assigned its own meaning in its own functional terms to the source output, a confusion of meaning propagates across the network. While there are ways to limit this confusion, these ways will ultimately break, as demonstrated by the examples in the previous paragraph, as the rate of change or the degree of complexity (i.e. degree of information exchange) increases.

Another approach discussed by French [18] is to have new learning always include some reiteration of old learning. One way to achieve this reiteration is by direct inclusion of a selection of past inputs. Another way is to separate two subsystems for new learning and permanent storage, with new learning occurring first in the new learning subsystem, and learning in the permanent storage subsystem resulting from a combination learning in the new learning subsystem and pseudopatterns corresponding with existing learning generated from permanent storage [17]. However, presenting a mix of old and new inputs again mitigates but does not directly address the issue that any change in the meaning of a device output in favour of some system objectives will inevitably result in loss of meaning for other system objectives.

Coward [8] therefore argued that the starting point device algorithm required to support (partially ambiguous) context in a functionally complex system is one in which information combinations are selected instantaneously and permanently as subsets of larger provisional input sets, and the presence of any repetition of a recorded combination is indicated by a consistent output. In the most conservative case, only one such combination is recorded per device. Once the performance of a system using this algorithm is understood it becomes possible to (slightly) relax this rigorous algorithm in ways which do not jeopardize maintenance of context. For example, the proportion of a recorded combination required to generate an output could be reduced, or multiple combinations which tended to occur at correlated times could be programmed on the same device. However, such relaxations must be such that any recipient modules can maintain an adequate context for their own functional purposes.

As will be discussed below, conventional neural network algorithms are appropriate in the functionally simple competitive subsystem, where they are in fact necessary for the management of learning.

5 Minimization of information exchange

In a system which heuristically defines its own functionality, individual modules make local decisions on the inputs which they will accept. However, as described earlier, for an effective functional architecture it is necessary that information exchange between modules be minimized globally. The requirement for minimized information exchange means that inputs to devices within a layer must be limited to as small a set as possible, and inputs to a higher level module must be limited to outputs from as small a set of other modules as possible. Because device and higher module inputs are locally determined, a global management process is needed to limit the possible range of the local decisions.

This information exchange minimization process must reach a compromise between the functional coordination value of additional information and the weakening of the architecture resulting from extra information exchange. A module can therefore be allowed to take information from a new source only if there is a high probability both that the module needs additional information and that the specific information will be functionally valuable to the recipient. The only available indication of the potential functional value of the outputs of one module to another module is the degree to which the two modules have often been active in a temporally related manner (e.g. at the

same time) in the past. A partial rerun of recent module activations would make it possible to determine this parameter. Provisional connectivity could then be established between modules which are often active in a temporally related fashion. For example, at the device level there is a requirement for the capability to record additional input combinations. These combinations are subsets of larger sets of provisional inputs. If these provisional inputs were randomly selected but with a bias in favour of inputs which had frequently contributed to firing of other devices in the same module in the past, the probability that the connectivity to support relevant combinations would be available would be increased, and the probability of irrelevant inputs active by accident being incorporated would be reduced. At a higher level, provisional connectivity would be established between modules which were frequently active in a temporally correlated fashion, provided there was some indication that the target module needed additional information.

The requirement for additional information can be identified at a module level as follows. Suppose that the outputs from one module are used by a competitive subsystem to determine behaviour, and that exactly the same module outputs at different times result in the same behaviour but positive consequences on some occasions, negative on others. This condition implies that the module outputs do not discriminate between conditions which are significantly different for functional purposes. This is the situation in which additional inputs to the module are required. In later experience, additional information recording is triggered which incorporates the new inputs and generates new outputs which discriminate between the apparently similar conditions. Note that this indirect use of consequence information to influence the behaviour of the repetition hierarchy minimizes the effect on context for information exchange because already existing outputs are not changed. An additional benefit of such a process is that it provides the information necessary to ensure that modules on one level responded to roughly equal proportions of system input conditions.

This process involves taking the system off line from experience, and performing a rerun of a selection of module activations, with an emphasis on recent activations. The system would experience this as a rerun of a juxtaposition of many elements from past inputs but the experience would leave no record for later retrieval. Coward [8] has pointed out the resemblance between such a process and REM sleep in mammals. It is important to emphasize the difference between this model and other REM sleep models which have been proposed [26, 27, 29]. No changes to recorded information take place in this model during REM sleep. The effect is to put in place the best available estimate of appropriate provisional connectivity which can be used if required to record information during future experiences.

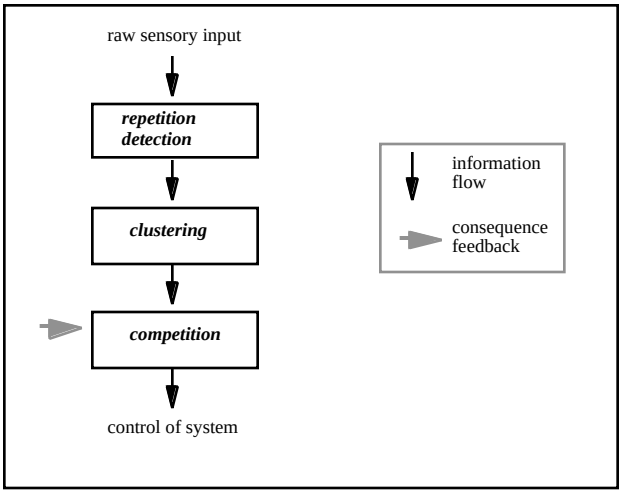

Figure 2. High level view of the recommendation architecture

6 The recommendation architecture clustering system

The primary separation of the recommendation architecture into clustering and competition is illustrated in figure 2. Clustering is a hierarchy of modules which select and permanently record combinations of their inputs and indicate the presence of any repetition of these partially ambiguous information conditions by an output. The selection process is unguided by any unambiguous system criteria and in general there will be some random element in the determination of which combinations are recorded. Architectural considerations can improve on completely random selection, but cannot eliminate the random element. Some module outputs are received by competition which uses consequence feedback to interpret them into behaviours. In order to provide competition with information to work with, every system input condition must generate some output from clustering. Such an output

may of course be interpreted by competition into a "behaviour" of doing nothing. Modules must be added or their portfolios extended dynamically in response to a sufficiently novel condition to ensure that some clustering output results. In order to meet the requirement for a simple functional architecture, information condition portfolios must be defined for the modules on one level in such a way that over the long term each module generates an output in response to roughly equal proportions of system inputs and information exchange between modules is minimized using the off line process described earlier.

Within the recommendation architecture form imposed by complexity, many different clustering implementations are possible. These implementations will differ in the effectiveness with which they could use different types of sensory input, learn to perform different types of behaviour, and utilize available device and connectivity resources. This is analogous with the many different systems designed within the constraints of the von Neumann architecture which all share certain forms but differ at a sufficiently detailed level.

In order to illustrate the factors which make a difference between one clustering implementation and another, consider first a very simplistic implementation. Suppose that clustering receives raw output from a visual sensory system and is made up of one level of devices which each receive as input a different randomly selected number of randomly selected raw outputs. All inputs to a device have the same weight. The system is able to count how many devices are generating an output in response to the current input condition and generates a signal S if that number is less than 100. Devices initially are in a virgin condition which will not respond to any combination of their active inputs unless the signal S is present. In the presence of that signal the threshold number of active inputs required to fire virgin devices gradually falls. The ones with the highest number of active inputs will then begin to fire. When a virgin device fires, it is converted to a regular device by deletion of all inactive inputs and setting of its threshold such that it will fire if all its inputs are active, independent of the signal S.

Given enough devices, such a clustering implementation would generate at least 100 device outputs in response to every input condition. A very long learning process might eventually be able to associate some of the device outputs with useful responses to the corresponding visual input. There are some serious problems with this implementation however. The first problem is the huge number of device resources required. Eventually there could be more devices that raw inputs, and the competitive subsystem would be swamped with connectivity. Many of the devices would be programmed with information combinations which had little functional value, in some cases less value than some raw inputs. There is no mechanism for identifying devices with low functional value and evolving their programmed combination in such a way that they become more useful without losing whatever value they already have. Finally, the 100 device output limit was selected arbitrarily and it may be either excessive or inadequate to discriminate between functionally significant differences.

A more effective clustering implementation must therefore address these issues. It must compress the input space into an output space in which an output has less ambiguity (i.e. more functional value) than an input. It must have a way to influence the combinations selected for recording in favour of combinations most likely to be of functional value. It must find ways to modify the conditions under which outputs are directed towards competition in such a way that the output becomes more useful without damaging its existing value. Finally it must find ways to generate the minimum number of outputs which will provide the competition with enough information to make appropriate behavioural selections.

There are therefore a number of management functions which an effective clustering implementation must perform. It must select and partition the input space in which it will search for repetition. It must select repetitions in such a way that there is an information compression between a potentially huge input space and its output space to the competitive subsystem, and the outputs contain adequate functionally useful information. It must accurately detect the presence of any repetition, and it must indicate the presence of a repetition with enough information richness to communicate adequate context despite changing repetition portfolios. Much of the structure of clustering described in section 9 is required to perform these functions effectively in such a way that proliferation of resources (such as total number of device level repetitions) is minimized.

A critical general point is that these management functions must also be based on the detection of repetition since this is all that is available to the system. The complexity of information combinations most appropriate for these management functions may not be the same as that of types of combinations appropriate for guiding behaviour, where information combination complexity is defined as the number of raw system inputs which contribute to the combination. For example, combinations of roughly the same complexity as features, objects, groups of objects, and groups of groups of objects may be the most useful for determining behaviour because the discrimination between the cognitively significant conditions by the partially ambiguous combinations will be optimized at this level of complexity. However, combinations with different complexities may be optimal for the management functions. Combinations of all the necessary types must therefore be detected. The structure to ensure that the system defines and detects combinations at the appropriate levels of complexity will in general be specified by design, and different structures may be most appropriate for different functional problems. The requirement to detect repetition on different levels of complexity will result in a layering of devices, with devices in one layer detecting the repetition of

combinations of combinations detected by the previous layer. The number of layers will be determined by the number of different repetition complexities required for behaviour and system self management.

In addition, the types of repetition appropriate for determining one type of behaviour may be somewhat different from the most appropriate for a different behaviour. There would therefore be an advantage in separate hierarchies of repetition, or separate superclusters, for different types of behaviour (for example food seeking, fearful or aggressive) with little information exchange between them. An additional advantage of such a separation is the ability to globally modulate the relative probability of different types of behaviour, based on some indication of system need (for example detection of hunger, self weakness, or threat) without jeopardizing context for information exchange. The disadvantage is the increase in resources required, and in general a compromise can be expected with only some different very general types of behaviour having separate superclusters.

7 The recommendation architecture competition system

Clustering provides competition with a range of outputs indicating the presence of information conditions in current system inputs which correlate partially with the presence of features, objects, groups of objects etc. These outputs resemble outputs in response to past conditions and carry some indication of the degree of similarity with such past conditions. Although the information is coarse coded, it has enough content to discriminate between functionally significant differences. If it did not have enough discrimination, the feedback of contradictory consequences mechanism described in section 5 would add such content, adding information exchange between clustering modules to achieve the required content. The interaction between functions is thus managed within clustering.

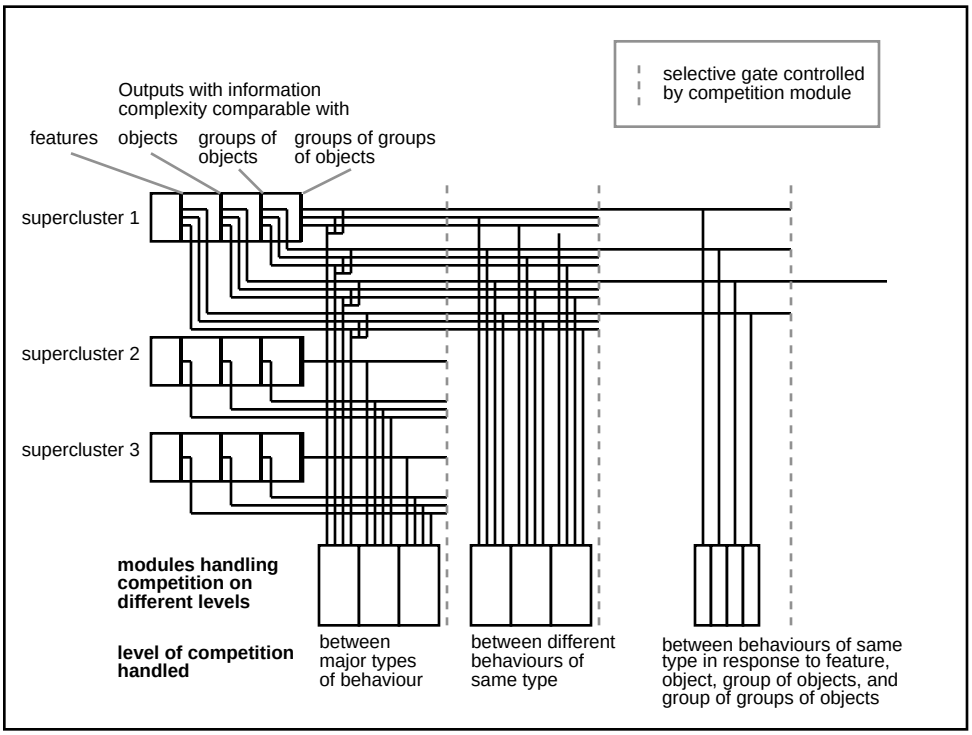

Figure 3 Selection of behaviour narrowed down by selection of smaller subsets of available clustering outputs by a series of competition subsystems

Competition interprets these clustering outputs into behaviours, using consequence feedback to improve behavioural responses. However, it cannot carry out this function in a manner that requires a complex pattern of internal information exchange, or the use of consequence information would generate the major context issue as discussed earlier.

The way to avoid the context problem is for separate modules within competition to correspond exactly with different system behaviours, with a winner-takes-all competition to determine the behaviour selected at any point in time. The output of a module if received by another module then has the meaning of a recommendation not to perform the target module behaviour, and a module output which exits competition has the meaning that the module behaviour will be performed. Consequence feedback can then be directed at the single module which generated the

behaviour. For example, if the consequence is favourable the module adjusts its input weights to increase the probability that it will generate an output in response to similar inputs from clustering in the future, and decrease the probability that it will be inhibited from producing an output by inputs similar to those generated by other competition modules.

There are, however, three major issues with this simple implementation. One is how such a system could generate complex behaviours. The second is the very high volume of information which needs to be communicated to each module. The third issue is how outputs from clustering can initially become associated with behaviour modules other than at random followed by a very long training process.

In principle the answer to how complex behaviours could be generated is that each module corresponds with a very simple action, such as a short sequence of muscle movements. Information from clustering drives long sequences of these simple actions, resulting in complex behaviours. Note that input to clustering would include information on current body position, hence output from clustering could take account of the point in the sequence which had been reached by the current behaviour.

However, this “in principle” answer emphasizes the problem of delivering all clustering outputs to the huge number of modules corresponding with each simple action. The full solution to the problem of complex behaviours and volume of information flow requires that both clustering and competition are structured in such a way that competition can make decisions on actions which are the release of subsets of outputs from one stage of clustering to a later stage of clustering. The outputs from the later stage then pass through a further subsetting by competition. Functionally this is a process of selecting a broad type of behaviour, then a more specific behaviour within the broad type and so on until the selection is between alternative detailed muscle movements.

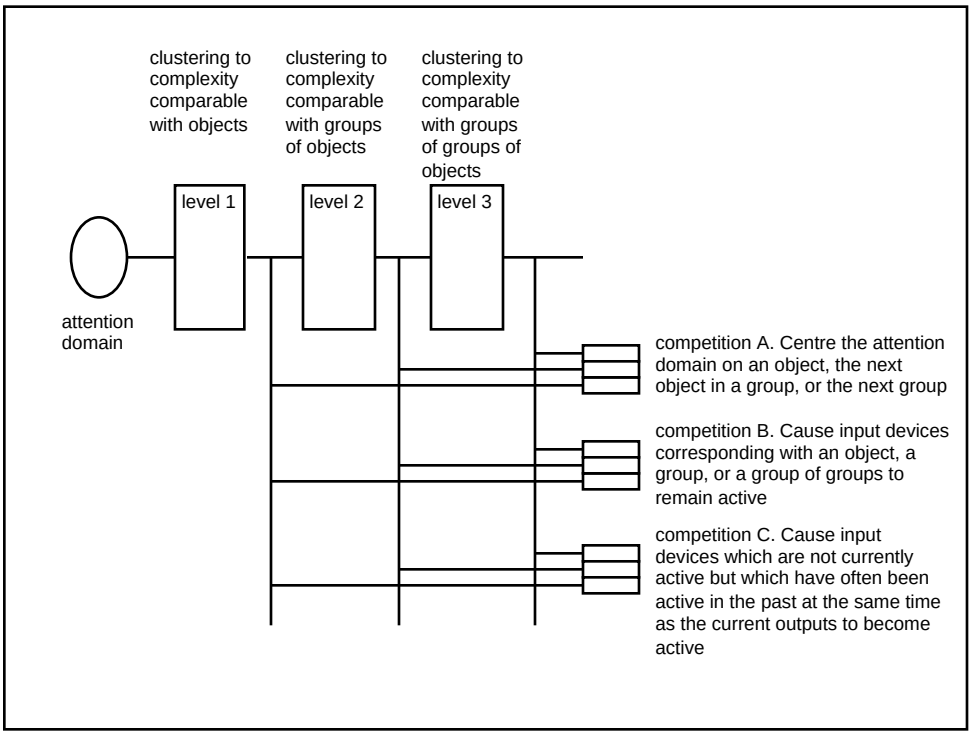

Figure 4. Management of attention to reduce the volume of connectivity

The gradual selection of behaviour can be understood in general terms by consideration of figure 3. In the figure, superclusters as described in the previous section independently find repetitions in current system inputs which could be interpreted as behavioural recommendations of different general types. Such types could be aggressive, fearful, food-seeking etc. The first competitive subsystem selects the outputs of only one of these superclusters and allows it to proceed. The next competitive stage selects between specific behaviours of the same general type. The next stage selects between behaviours of the specific type but targeted at objects, groups of objects, or groups of groups etc. Further stages could select between different physical ways of performing the selected behaviour (use of arm, leg, etc.). Information released from this stage would go to yet further stages which manage detailed movement. It is important to note that the various competitive subsystems gate but do not modify clustering outputs, because such modification would be functionally complex.

This gradual selection of behaviour reduces the total information flow at one time, but has less impact on the total connectivity required. To reduce total connectivity requires detailed management of an attention function as illustrated in figure 4. In the figure, sensory input from a controlled visual domain is received in the first clustering level and the information clustered to a complexity comparable with objects in the visual field. The results of a succession of outputs from this first level are held simultaneously active at the input layer to the second clustering level by prolonging the activity of the input layer devices. The combined information is clustered in the second level to a complexity comparable with groups of objects. A succession of outputs from the second level is clustered in the third level to outputs comparable with groups of groups. Each level compresses its input space to a smaller output space.

The selection made in competition A in figure 4 is between which level of clustering will control changes to the attention domain. If level 1 gains control, level 1 information will be passed to a more detailed level of competition which will select muscle movements to either adjust the attention domain for better fit to a single object or shift the attention domain to another object.

The selection made in competition B is again between levels 1, 2, and 3 and a successful action will cause devices which are already active at some level in clustering to remain active. One role for this mechanism is to achieve simultaneous activations in level 2 of information derived from several objects as described earlier.

A selection made in competition C activates a population of devices which are not currently active but which have often been active in the past when the clustering outputs controlling the action were also active. In other words, this mechanism activates information derived from, for example, objects which are not currently present but have often been present in the past when the current object was also present.

Note that there will be other levels of competition to determine which of the different types A, B, or C will be accepted, and the architecture could permit acceptance of more than one recommendation at the same time.

The third issue of how clustering outputs can initially become associated with appropriate behaviours, other than by a long process starting from random selection, can be addressed in several ways. One way is the provision of a priori hints, a second is by imitation of an internal model, and a third is external imitation. A priori hints would be repetitions programmed by design (or genetic programming) to indicate the type of behaviour to be tried first. For example, certain colours and shapes could encourage trying eating, others could discourage it. Subsequent behaviour would depend on learning starting from these guided first attempts. Imitation of an internal model involves initially invoking a behaviour based on sufficient information similarity with an already learned behaviour. For example, if a body of behaviour had been learned for table tennis, there might be enough information similarity with tennis that such behaviours would be a good starting point for learning the different game. External imitation is a cognitively complex concept but could potentially provide effective starting points for learning.

Note the implication that if no suitable behaviours are available in the current competition portfolio of modules at some level, a module which will eventually select a new type of behaviour can be created and given as inputs the clustering outputs which usually occur at the same time and have no appropriate behaviour. This will tend to happen much more frequently at higher behavioural levels than at very detailed levels.

To handle overlap between successive similar inputs, there must be some way to distinguish between inputs from different objects. Such distinction could be on the basis of different level 2 devices initially receiving the inputs from level 1 and other devices finding combinations across all inputs, or by tagging device activity in some way. Using different input device populations is the technique used in the actual implementation described in section 20. A more sophisticated approach would be to tag device activity with a marker such as frequency modulation of device firing rate. This marker could indicate the input population from which the device activity derived.

The management of reward in the competition system remains to be discussed. There are two aspects to this management process. One is identifying the targets of reward within competition, the other is determination of the conditions which attract a reward.

The selection of behaviour involves a series of selections on different levels of detail as described earlier. Hence the reward can be applied independently to the behaviour decisions on these different levels. For, example, a negative consequence following an inept muscle movement will not necessarily apply to the general type of behaviour under way at the time. A reward targets recent competitive selections. However, the definition of recent may cover a longer period for general types of behaviour than for detailed muscle movements.

The conditions under which reward will be assigned will vary with the level of the behaviour, and in general will be defined heuristically through a clustering, competition subsystem [8]. In general there will be specific types of conditions defined a priori as attracting rewards, but conditions which are frequently present when a reward occurs could be added to the a priori conditions. In the case of more general behaviour type decisions, a priori conditions could be similar to those which rewarded more detailed behavioural decisions, but conditions which are of the level of complexity of the clustering outputs which determined the more general behaviour could come to influence such reward decisions.

To illustrate how the different clustering levels and competitive subsystems can come together to support generation of a complex behaviour, consider the model for mental arithmetic processing illustrated in figure 5. There is a domain of visual attention which is the domain from which input is allowed to pass into clustering level 1. The size and positioning of this domain is controlled by outputs from levels 1, 2, and 3 acting through competitive subsystem A in figure 4.

Initially the attention domain covers the whole of the equation (step 1 in figure 5a). The domain is matched to the area of the equation because information derived from mismatch would be interpreted by the combination of level 1 and subsystem A as muscle movements which improve the match. Once matching is achieved, the outputs of level 1 are interpreted by subsystems A and B as actions to shift the attention domain to the left segment of the equation (step 2 in figure 5a) and to prolong the activity of some of the devices in level 1 which were activated by the full equation. Step 2 results in an output from level 1 in response to the left segment of the equation. This output is again interpreted into two actions. One is to shift the attention domain to the left object in the left segment (the number 3 in step 3), the other is to prolong the activity of some of the devices in level 1 which were activated by the left segment.

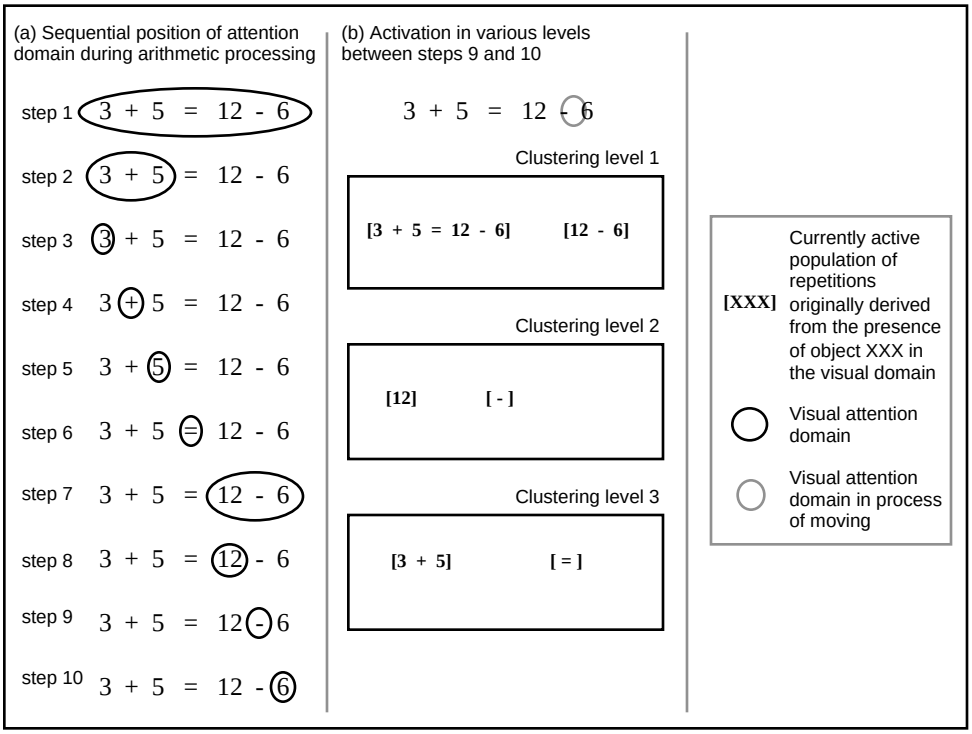

Figure 5. Arithmetic processing model as an illustration of how a complex cognitive function could be managed with a recommendation architecture

Steps 3 to 5 result in outputs from level 1 in response to the three characters in the left segment of the equation which are passed to level 2 and held active simultaneously. The holding active simultaneously in level 2 is as a result of interpretation by competitive subsystem B of the prolonged activation in level 1 of the information derived from the left segment. The output from level 2 resulting from the combined activation of the three characters is passed to level 3, where it is held active as a result of the interpretation of the continued full equation activity in level 1 by competitive subsystem B. Steps 6 through 10 repeat the segment process for the other two segments of the equation, at the end of which, information derived from all three segments is simultaneously active in level 3. Level 3 then generates an output which could be interpreted as a speech recommendation (e.g. to say “that’s not right” in the example in figure 5). A more interesting action would be that generated by competition C in response to the level 3 output. This action would activate devices in earlier levels which had often been active in the past at the same time as the current activations. Since devices recording information in response to other numbers would fall in this category, a pseudo-activation could be generated which after some experimentation could be a 4 in place of the 6. Inclusion of this pseudo-number in place of the 6 in a repeat of the ten steps would result in a level 3 output which could be interpreted as a recommendation to say “that should be 12 – 4).

7.1 Relationship of competition with reinforcement learning

The algorithms employed in competition have some similarity to those used in reinforcement learning, but there are some significant differences. A typical reinforcement learning algorithm [37] has a policy, which defines the way the system will behave at a given time; a reward function which maps each perceived state to a single number which is the reward for being in that state; and a value function which is the expected total reward over the long term starting from any given state. The reward function is not alterable by the system. The key element in reinforcement learning is predicting value.

In the recommendation architecture, policy corresponds with the current set of weights in the various competition subsystems. However, there is no value function, and in this sense the algorithm is more similar to evolutionary learning. Rewards act directly on policy, but the policy is multileveled. There are separate but interacting competitive subsystems (i.e. policies) for behaviours on different levels of detail, from general type to detailed muscle movement. In addition, the reward function can be changed in response to learning, because the definition of reward conditions are themselves specified by a clustering, competition system.

8 Learning complex, real time behaviours

The problem facing a cognitive system is learning to respond appropriately to a wide range of conditions. There are two aspects of this problem which have rarely been taken into account in cognitive models. One is the constraint imposed by the passage of time on the availability of information and the appropriateness of a response. The second is the complexity with which appropriate behaviour towards one condition depends upon subtle variations within the condition, on the presence of other conditions, and on variations within the cognitive agent. These two aspects of the cognitive problem are the essence of functional complexity.

For example, suppose that two people, Rick and Elaine, encounter each other. The simple categorization problem is for Rick to identify Elaine and speak her name. Even that is not as straightforward as it appears. There are variations in Elaine's appearance and the lighting conditions from past occasions, and variations in the position from which Rick sees her, with the result that no raw visual input will ever be an exact repetition of any past condition. Furthermore, Rick's raw visual input will be constantly changing as he and Elaine move, partly in response to their interaction, and Elaine's appearance may be changing in response to Rick.

However, the behavioural problem is not simple categorization. For example, depending on the location of the encounter, the appropriate response might be "Hi Elaine, what are you doing here?". Depending on other people also present, it might be "Hi Elaine, I didn't know you knew these people". Depending on her detailed appearance, it might be "Hi Elaine, are you feeling okay?", although this response might not be appropriate depending on the identity of the other people present, but if the appearance was sufficiently bad it would be appropriate whoever else was there. The appropriate response would depend on whether Rick wanted to be drawn into a conversation at that moment, but that internal state might be affected by changes to Elaine's appearance (e.g. expression). Finally, the appropriate response must be delivered within a couple of seconds.

Appropriate behaviour is thus not a simple categorization, but must be determined in a short available time by a complex interaction between the presence of perceived objects, detailed features of the objects, groups of other objects, and the internal state of the agent, all of which are changing in response to each other. Furthermore, appropriate behaviour must be learned without a priori knowledge of what information will be relevant to which behaviours, but that information must be organized in such a way that all needed information can be accessed within the time limit for the behaviour.

It is of interest to consider a cognitive model such as EPAM III [15] from the point of view of the issues which arise in learning complex real-time behaviours. EPAM is an attempt to model performance and learning in a recognition system. The input to EPAM is a set of features provided by an (unspecified) sensory-perceptual system. Its output is a symbol that accesses an image stored in semantic long term memory (also unspecified). An image in EPAM is defined as a feature list, and recognition is a tree search for an image which contains a large set of the features in the currently perceived object and no features that are not in the object. Learning is of two types. In familiarization either a new image is defined because no image corresponding with the current object could be found, or an image containing a subset of the object features is modified by addition of some of the missing features from the object. In the second type of learning, discrimination, the tree used in the search is updated because it generated an image which did not correspond with the object.

The first question is, how could EPAM model the management of complex behaviour? The only apparent way would be for sets of images retrieved from semantic long term memory to become associated with behaviours. Consider the problem which develops when there is a need to learn. Suppose that the system is faced with a situation in which two objects with identical feature lists require different behavioural responses. For example, a familiar person might develop a subtle expression which is not one of the existing features but which indicates that they did not regard the behaviour of the agent as appropriate in current circumstances. The sensory-processing subsystem would somehow have to be told to add new features, but the recognition system has no knowledge of the

type of features required, because all its information is derived from the set currently delivered. Hence the sensory-processing system would have to add features at random until one that could be used was identified.

In fact, EPAM offers no mechanism for such feature addition. The implication is that features must be sufficiently simple that every feature that could conceivably be relevant to the recognition system is provided a priori. The level of complexity of such features would be similar to that of shape primitives (e.g. boundary elements) of which there would be a very large number. Furthermore on this model there must be an object or small set of objects for every behaviour, again a very large number. Hence the implied EPAM model for handling functionally complex behaviour is a tree search from a very large number of primitive features to a very large number of object sets. Such a model would experience severe real-time problems.

This discussion of EPAM illustrates a number of points. The first is that the capability to map high level recognition of a behaviour problem to a detailed level at which the problem can be solved is essential. The second is that the boundary between two subsystems must be made in such a way that problems of this type can be resolved. The third is that when information is exchanged between two modules, if that information is a predefined unambiguous set there will be problems with learning. The way in which a system with the recommendation architecture handles these problems is discussed in section 12.

9 Intermediate design description of clustering implementation

A system with the recommendation architecture has been implemented. This section will describe the design at an intermediate design level. The next section will explain the functions of different parts of the design. A later section provides a detailed design description.

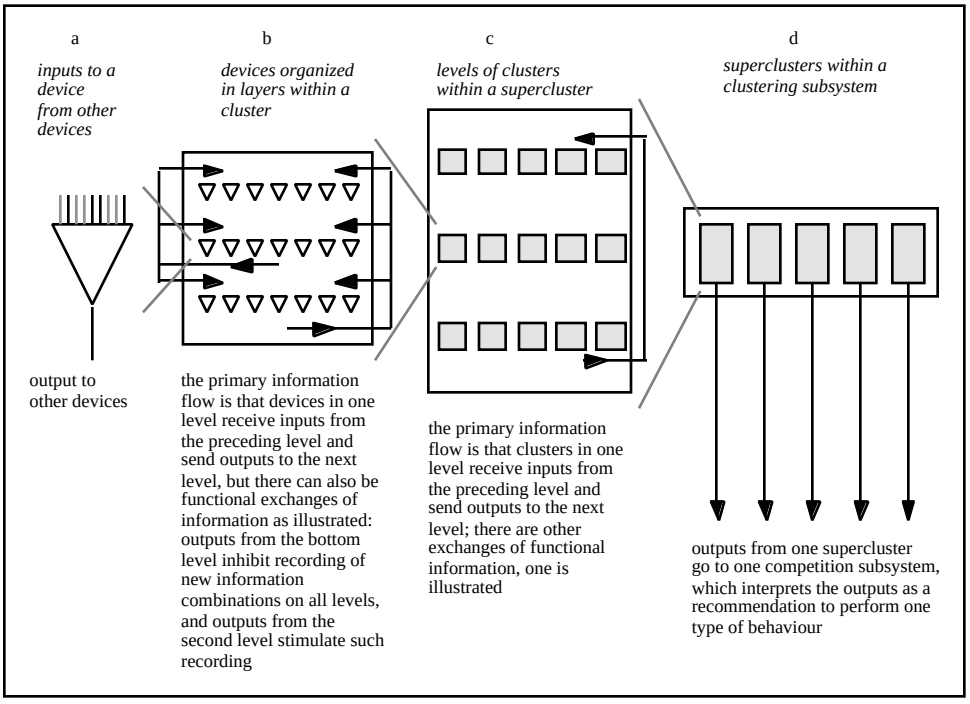

Figure 6. The architecture of a clustering subsystem at various levels of detail. A device is the basic element which records information combinations. The illustrated device had a number of provisional inputs selected by a random process. The black inputs were the inputs active when the device was triggered to select its information combination. The grey inputs were provisional inputs which were not active and are therefore deleted. The device will produce an output if a large subset of its black inputs repeats in the future. Devices are organized into layers, layers into clusters, and clusters into superclusters (additional levels are possible but not illustrated). Each level records and detects information combinations made up of sets of combinations at the more detailed level, and indicates any repetition of a large subset of its set by producing an output. Outputs are all ultimately combinations of device outputs.

The clustering subsystem design is illustrated in figure 6. At the component level the clustering subsystem uses devices which can record combinations of information and indicate any repetition of the combination by producing an output. Such a device is illustrated in figure 6a. A device initially has randomly selected inputs, all with the

same weight, but a threshold set at a notional infinity so that it will not respond to any input combination. These devices are referred to as virgin. During the sleep with dreaming process which configures the virgin devices there are various biases placed on the random selection process which will be discussed under detailed design. There is a functional signal which if present gradually lowers the effective thresholds of virgin devices, and another functional signal which overcomes the first signal and sets such thresholds back to notional infinity. The origin of these signals is described below. If a virgin device actually fires it produces an output, all its inactive inputs are disconnected, and its threshold is set at or slightly below its remaining input count, independent of the future presence or absence of the functional signals. The device has effectively recorded an information combination. I refer to this recording process as imprinting of an information combination on a virgin device. The meaning of a device output in the future is therefore that an information combination with a degree of similarity in information terms to the original combination is present.

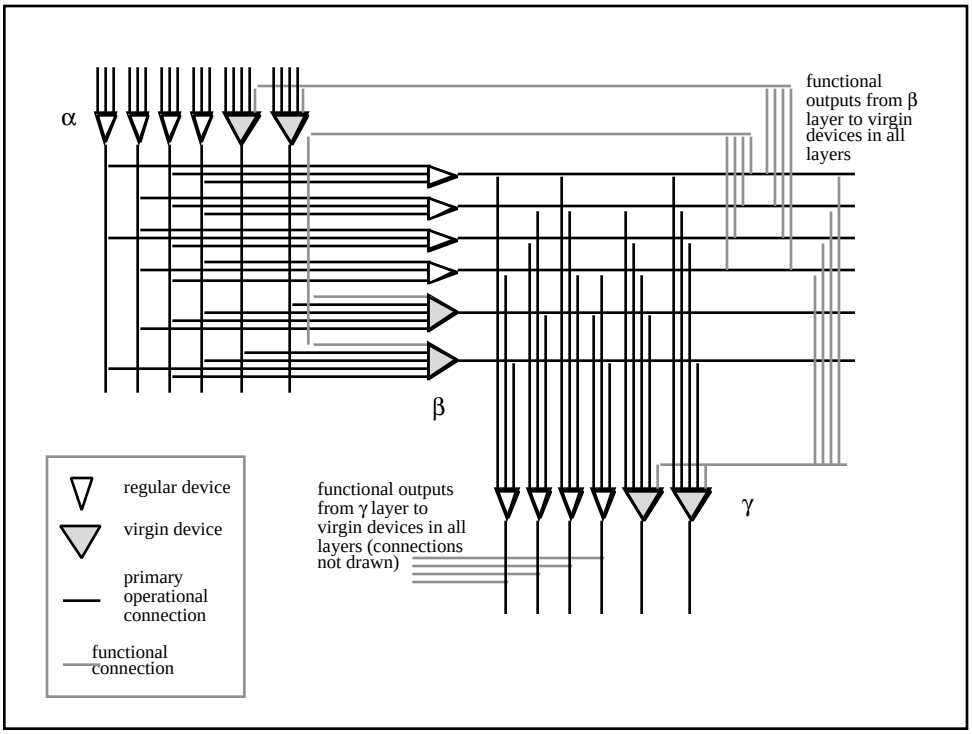

Figure 7. Connectivity within one cluster. Primary connectivity is excitatory. Functional connectivity from γ level to virgin devices inhibits virgin device imprinting, and is stronger that the functional connectivity from β level to virgin devices which excites such imprinting.

Devices can only add additional inputs after imprinting under narrowly defined conditions which will not destroy context for the recipients of their outputs. For example, in the electronic implementation devices in the β layer of the cluster described below can add inputs under the following restricted conditions. Devices are given provisional inputs from virgin devices in the preceding α layer, the number of such provisional inputs is proportional to how frequently the device has fired in the past when the cluster as a whole has produced an output. If the β device is firing when the α device is imprinted, the connection becomes permanent. Some of the inputs which cause the β device to fire can come from these newly imprinted virgin devices, but the total number of inputs from regular devices must amount to more that half the β device threshold for the device to fire. Hence the portfolio of combinations to which the device responds can be expanded, but only if the input condition has significant similarity to past conditions at both the cluster level as indicated by the presence of imprinting, and the device level as indicated by the existing inputs which are active.

Figures 6b and 7 illustrate a module called a cluster, which in the illustrations is made up of three layers of devices of the type described. Devices in the top α layer receive inputs from outside the cluster, devices in the middle β layer receive inputs from the top layer, and devices in the bottom γ layer receive inputs from the middle layer. All these inputs define information combinations which increase in complexity from top to bottom.

In addition, devices in all layers receive functional inputs from β layer devices which together indicate how much firing is present in the middle layer. These inputs are the functional signal which lowers virgin device thresholds.

Devices also receive inputs from a set of γ layer devices which together indicate if any firing is present in the bottom layer and are the functional signal which forces virgin thresholds back up to notional infinity.

There is further connectivity from devices in the α layer of all the clusters in a level. This connectivity is inhibitive and operates to prevent the imprinting of a new cluster. There is also inhibitive connectivity from devices in the β layer of all the clusters in a level to devices in all other β layers. This connectivity operates in a winner takes all manner and ensures that imprinting only occurs in one cluster in response to one input condition.

The functional connectivity is actually implemented in the electronic version by calculating average effects applied to every device in the target layer rather than by explicit connectivity.

The design decision to use three levels implies that all necessary behavioural and management functionality can be supported by information combinations at three levels of complexity. Cluster structures with more layers and with different functional connectivity may be required for higher degrees of functional complexity to reduce use of resources, help minimize information exchange, and for specific functional purposes [13].

The outputs from such clusters are the outputs of devices in the γ layer, and the particular combination of outputs provides information about where in the spectrum of cluster repetition conditions the current input condition is located. If combinations are required which have a higher information complexity than those indicated by cluster outputs, the outputs can be further clustered by successive layers of clusters as illustrated in figure 6c. This hierarchy of cluster layers make up one supercluster. The clustering subsystem is separated into superclusters which independently select and indicate repetitions as illustrated in figure 6d.

10 Functional description of the simple clustering subsystem design

The purpose of this section is to explain the functionality achieved by the design described in the previous section. An important aspect of this description is how functionality is achieved by separation into modules in such a way that higher modules provide management processes for more detailed modules.

At the most general level, clustering has two functional objectives. One is to generate outputs in response to every system input condition which are coarse coded but correlate sufficiently with functionally relevant conditions to allow the functionally simple competition system to determine behaviour. Secondly, to generate internal information which is again coarse coded but adequate to manage the evolution of clustering in such a way that both adequate information contexts and a simple functional architecture are maintained.

The five levels in the clustering functional hierarchy which will be described are device, layer of devices, cluster, layer of clusters, and supercluster. The function of a supercluster is to generate outputs of an appropriate information complexity to guide one type of behaviour. The function of a layer of clusters is to generate outputs at some functionally useful level of complexity in response to all system input conditions. The function of a cluster is to select and detect a portfolio of similar conditions which are a subset of system input conditions roughly equal in terms of occurrence frequency to those of all other clusters in the same layer. The function of a layer of devices is to detect and indicate the presence of information conditions which are of a level of complexity useful to one or more functional or management process.

The function of a device is to record information conditions under the influence of higher modules (which of course exert that influence by combinations of device outputs) in such a way that it can indicate any repetition of the condition in a manner that is functionally useful to the targets of its outputs. These functions will now be considered in more detail.

A device records an information condition, and indicates the first occurrence and any sufficiently similar repetition by producing an output. “Sufficiently similar” is a compromise between conflicting requirements. One requirement is retaining context. A second is the need for the device and the module within which the device is located to respond to roughly the same proportion of inputs as their peers. A third is the minimization of device resources which record functionally irrelevant conditions.

Some changes to the combination originally programmed on a device referred to above which could be consistent with maintaining an adequate context are reduction in threshold and addition of inputs (or even combinations) which are present when an already programmed combination is present. To take place, the recording or modification of a combination requires a higher level functional signal indicating that the benefits of change outweigh the resource and context costs. The sources of functional signals indicating appropriate or inappropriate change conditions originate in specific device layers, often within the same cluster as the device.

There are a number of approaches to ensuring that changes still allow recipients of the information to retain an adequate context. One approach is to limit the range of modules which can be targets for the outputs, any changes then being limited to those which meet the functional needs of that limited set. A second approach is to provide additional context information by modulation of device outputs or by using the identities of combinations of devices which generate simultaneous outputs. This approach essentially provides an indication of where in the range of conditions to which the device or module responds the current condition is located.

Each layer of devices in the simple three layer design has three functions. The α layer records and indicates to the β layer information conditions of a somewhat higher complexity than cluster inputs; it manages the conditions under which new clusters are created; and it manages the addition of input information to the cluster. The β layer records even more complex conditions and indicates their presence to the γ layer; it indicates conditions in which recording of new combinations by the cluster is appropriate; and indicates when such recording is inappropriate in other clusters. The three functions of the γ layer are recording of yet higher complexity conditions; indicating that conditions are inappropriate for recording new combinations; and generating cluster outputs with adequate structure to support context. It is important to note that functional signals are recommendations not commands. For example, if a virgin device does not have enough active inputs in its currently programmed provisional set to achieve a minimum threshold, it will ignore the recommendation to imprint from the β layer. As another example, if a cluster has received a signal indicating the presence of contradictory consequences that signal will partially overrule the γ layer signal inhibiting imprinting.

The functional role of a cluster is to ensure that other clusters overlap with its portfolio as little as possible, and that it responds over the long term to approximately the same proportion of input conditions to the layer of clusters as every other cluster. The function of a layer of clusters is to compress an input space into an output space which has an output information complexity and discrimination appropriate for a major (generally behavioural) role.

11 Functional description of knowledge acquisition

The functional operation of knowledge acquisition by a layer of clusters can now be described. A layer of clusters divides up experience into a set of heuristically defined repetition conditions. Successive layers of clusters divide up the same experience into sets of conditions at increasing levels of complexity. To understand this process, suppose that one layer of clusters performs the complete compression process from sensory inputs at a fairly primitive level to repetition complexity comparable with objects. Suppose further that this layer receives a sequence of sensory inputs derived from just apples, plums and blueberries. These fruits can be distinguished by size, shape, color etc. but the system has no knowledge of these categories in advance. Suppose further that the first object is an apple. No cluster produces any output because no combinations have been recorded so far. This absence of response triggers random selection of one cluster, and information combinations are recorded until a cluster output results. Any future object may produce an output from this cluster if it generates enough information repetition in the middle layer. The middle layer information combinations might happen to correspond with conditions like *some degree of green plus some degree of roundness*. Such a combination could be triggered by more green and less roundness or vice versa, and therefore although on average an apple is most likely to trigger output, some plums and even some (unripe) blueberries might trigger output as well. As the cluster is exposed to more experiences, the set of input conditions to which it will respond broadens through recording of additional information combinations. Outputs from such a cluster might eventually indicate the presence of a spectrum of repetition conditions which might respond to most apples, some plums, and a few blueberries. An object which generated no response in this cluster would trigger creation of a new cluster which would develop a response to a different spectrum of repetition conditions. The clustering subsystem is managed towards the point at which every experience results in some output from at least one cluster. In other words, the space of detected repetitions is expanded until every experience contains at least some repetition. Initially many experiences require recording of new information combinations, this recording tends to zero with time unless radically novel conditions such as a new type of fruit are experienced, in which case addition of clusters and/or expansion of existing clusters will occur.

Consider now how this process operates in more detail. External input comes in to a level of clusters and perhaps generates activity of some α devices in some clusters. If the degree of activity in one cluster exceeds a proportion specified by design, no new cluster can be initiated. The function happening here is that if the current condition is similar at some minimal level to one of the clusters, even if that cluster does not produce an output the creation of a new cluster is inhibited. This similarity is at a moderate level of complexity as defined by combinations recorded in the α layer. Long term such conditions will fit either in one of the existing clusters or in a new cluster created in response to some more novel condition which expands to include such conditions. The purpose of this function is to balance the undesirable effects of clusters expanding too rapidly against the undesirable effects of clusters created too close together.

In each cluster, any α activity may produce β activity which in turn may produce γ activity. The functional objective is that at least one cluster in the level generates some γ device output which can then provide information for the competitive function to work with. If in any cluster some γ output is present, no further action is required and imprinting of virgin devices is therefore inhibited. If there is activity in the β layer of a cluster which is above a criterion defined by a design specified algorithm involving the ratio of active β devices to total number of β devices in the cluster, imprinting of additional information is justified

To give a further example of how this process works, suppose that the system inputs are from pictures by different artists. The raw inputs might be conditions correlating with the presence of boundary elements or colour.

The repetitions in the α level of one cluster might be of the same order of complexity as shapes, colours etc. Repetitions at the β level might be of the same order of complexity as combinations of shapes and colours characteristic of a particular artist. In other words, these repetitions would be of the order of complexity of patterns indicating the style of the artist. Repetitions at the γ level would then be of the order of complexity of individual paintings by the artist. However, bear in mind that the actual information combinations in clusters only correlate ambiguously with such unambiguous patterns. If the system has already had a range of experiences of painters such as Magritte, Gaugin, and Picasso, a range of clusters will already exist. The combinations recorded in the α layer of all clusters might have some similarity, since some colours and simple shapes are general to most painters. The combinations recorded in the β layer would be much more specific to individual artists. Since there are sometimes elements of style which are similar between artists, one cluster might respond at β layer to more than one artist. However, the combinations recorded at γ level would have much less in common between artists, although there could be enough general similarity to have some overlap across artists. Suppose that the outputs at the γ level correspond with recommendations to say the name of the artist and the name of the painting.

If the system is exposed to a painting which it has seen before, there is in general a repetition path which results in the same output as before. If the system is exposed to a painting which it has not seen before but which is by a familiar artist, there will be significant activity in the β layer of one or more clusters which have produced an output in response to paintings by the same painter. There will be no gamma level output if the painting is sufficiently different from past paintings, but the β level activity will trigger imprinting until an output results. The implicit combination portfolio of the cluster is now expanded to include paintings of this latest type. Note that because combinations may be recorded at all levels, the portfolio of the cluster can be expanded to include new use of colour for example at the α layer, changes to style in the β layer, and additional paintings at the γ level.

The handling of style is worth further discussion. If a painting by a radically different painter is seen, the system would generally create a new cluster. The same would apply if the painting were by the same painter but in a radically different, unfamiliar style. However, for a painting in a somewhat different style, there might be activity in the α layer enough to inhibit a new cluster, but not enough β activity to generate imprinting. This is essentially a decision to delay on the basis that in all probability either a very different style painting will be seen later which will create a new cluster into which the current painting would fit, or an intermediate style painting will be seen which will expand one of the cluster definitions to the point at which the current painting would be accepted. This mechanism thus reduces proliferation of clusters at the cost of delay to learning. It is most important in early system learning.

The cluster outputs from the γ layer thus provide the meaning that a painting is present by one of a small set of painters, and the identities of the specific active γ devices provides information on which of the number of painters is the current artist, and also information making it possible to distinguish between individual paintings by the same artist.

To conclude this discussion of knowledge acquisition, consider what is happening at the device level. A device receives primary information inputs from the preceding level. It also receives numerous inputs from specific other layers in its own or different clusters which communicate recommendations on whether or not to change its currently programmed repetition portfolio. The design or genetic specification must ensure that the different inputs managing change achieve an adequate compromise between maintenance of contexts, minimization of resources, and architectural simplicity.

12 Mapping of behavioural issues from high level to detailed level

It is of interest to consider how the clustering system would handle the problem raised in the discussion of EPAM in section 8.0. This problem was how the system could cope with inadequate discrimination by its feature inputs between functionally significant differences in objects. In the case of EPAM, the use of a fixed set of unambiguous features implied that all of the most detailed features which could be functionally relevant would need to be linked in a tree structure to all behaviourally relevant objects.

The inputs to clustering will in general be at the same most detailed feature level. Successive layers of clusters compress this huge input space into smaller output populations which correlate with behaviourally relevant objects. Suppose now that the same problem occurs: at the behavioural object level there is inadequate discrimination because of inadequate discrimination at a more detailed level. The initial response is that contradictory consequences will cause the clusters at high complexity level which generated the outputs will add inputs and generate additional outputs. If this does not resolve the problem, the same process is applied to the clusters at a less complex level which were the sources of their inputs at the times of the problem. This process can propagate if necessary to the least complex levels, adding outputs until the problem is resolved. The architecture thus makes it possible to target the detailed functional level at which the problem can be solved. This is an example of the logical paths which are made possible by a simple functional architecture.

13 Intermediate design of competitive subsystem

The competitive system interprets the functionally ambiguous outputs of the clustering subsystem into a system behavior using learning based on consequence feedback. The algorithms for this learning can be understood by consideration of figure 8. The devices, as illustrated in figure 8a, are similar to classical neural network devices. They can have both excitatory and inhibitory inputs with different input weights. In its simplest form the competitive subsystem is made up of a series of parallel modules or pipes (figure 8c). Each pipe corresponds with one type of behavior, and the pipe with the strongest output is the behavior in response to the current input condition. A pipe has two layers of devices (figure 8b). Devices in the first layer receive combinations of inputs from devices in the corresponding supercluster. These inputs have a excitatory weight. Devices in the second layer receive stimulatory inputs from first layer devices in the same pipe, and inhibitory inputs from devices in the first layer of other pipes. The threshold of all devices in all pipes is lowered until there is an output from at least one pipe. The strongest output is taken as the behavior, if there are several outputs of equal strength, one is selected at random. If the consequences of the action are good, all recently active excitatory input weights are increased and inhibitory weights reduced, and vice versa for bad consequences. Over a period of experience such a subsystem converges on a high integrity behavioral interpretation of the clustering subsystem outputs.

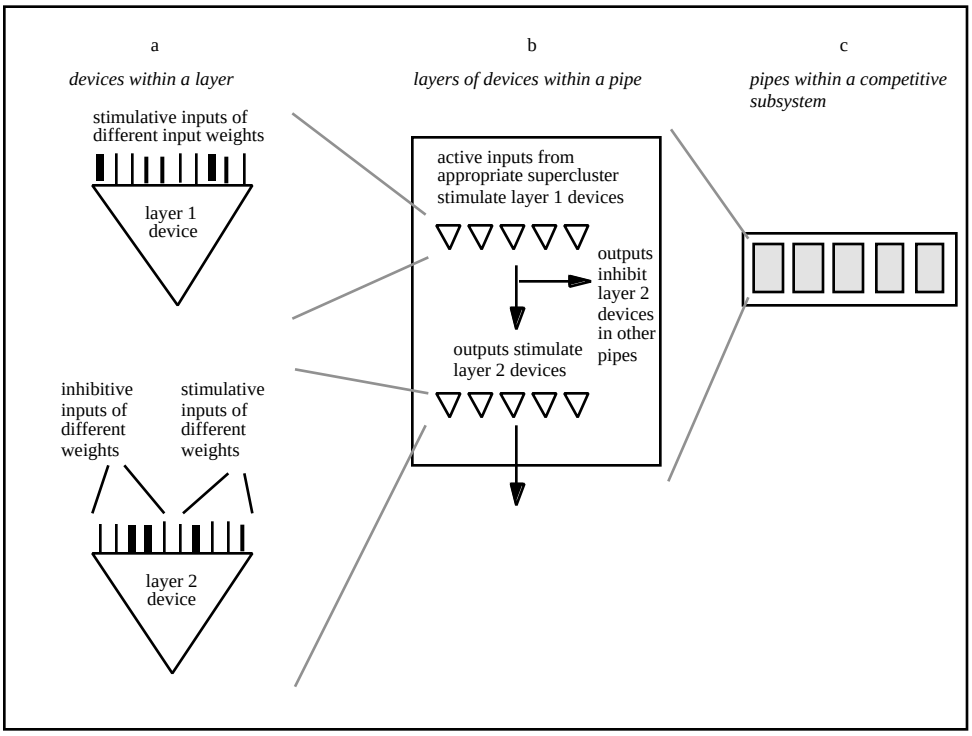

Figure 8. The competitive subsystem

14 Implemented recommendation architecture system

An electronic implementation of the recommendation architecture has been completed. Because the functionality of the system depends upon dynamic modification of connectivity (not just connection weights), devices and their connectivity are simulated in software rather than physically constructed. The system is written in Smalltalk V and runs on any Apple Macintosh platform. The system is presented with an input, and sequentially evaluates the state of each device in order to determine its response. Multiple passes through the system are required for each input because of various functional feedback paths described below. An inherent parallelism of the architecture (i.e. large groups of devices which could have their state determined in parallel) is not currently exploited. Inputs to the system are binary vectors which indicate the characteristics of the current input condition. These binary vectors can be of any size, simulations have been performed with input vectors up to 10,000 elements. Simulation time increases roughly linearly with input vector size, but if the inherent parallelism were exploited would be almost independent of vector size.

15 Architectural approach and simple categorization scenario

In information terms, a brain has a very large input space derived from sensory organs (visual, sound, propioceptic etc.). No overall input condition ever repeats exactly, and any one input condition can only be experienced once, it cannot be recalled in totality for later review. The biological problem is to find subconditions which repeat, and associate these subconditions with appropriate behaviours. The relevant repetitions may be complex combinations of individual inputs. For example, perception of a dog relies on raw retinal input on shape and colour, but also input indicating the distance of the object and therefore its absolute size.

The scenario used to emulate the biological problem was similar to that used in earlier simulations [13, 14, 20]. In this scenario, an input space of 1000 possible binary inputs or properties is used. Input conditions are 1000 element binary vectors, where a one-element indicated the presence of a property and a zero-element its absence. An input condition is generated by random assignment of ones and zeros based on a relative probability distribution. Different probability distributions defined different types of input conditions. Such different types could be interpreted as different categories of input conditions. In the simple category recognition simulations, ten categories were defined in the input space. The probability distributions for the ten categories are shown in figure 9. Any one probability distribution had a non-zero probability for about 50% of the input properties.

The problem the system is required to solve can be understood by consideration of figure 10. In that figure, the properties of 10 objects from two adjacent categories in a 100 input space are shown. The category of an input condition cannot be determined on the basis of the presence or absence of a few individual properties. For example, for objects in the 1000 input space, only about 60% of the conditions of a category include both of the two most probable properties for that category, and 5% do not contain either of the two properties. However, 5% of the objects in the categories on either side (in figure 9) of that category contain both of these properties, and 50% contain one of them. 10% of the objects in the two categories next furthest away in figure 9 also contain one of these properties.

The recommendation architecture system was presented with a sequence of different input conditions (or objects) of a number of the ten different types. Periodically the sequence was interrupted by a period of sleep to manage information distribution. A typical experience sequence was 100 input objects including 10 different objects of each category presented in the order A B C D E F G H I J A B C etc. This experience was followed by a period of sleep, and then another sequence of 100 objects . A simulation run included 760 different objects of each type being presented to the system, each object being presented only once.

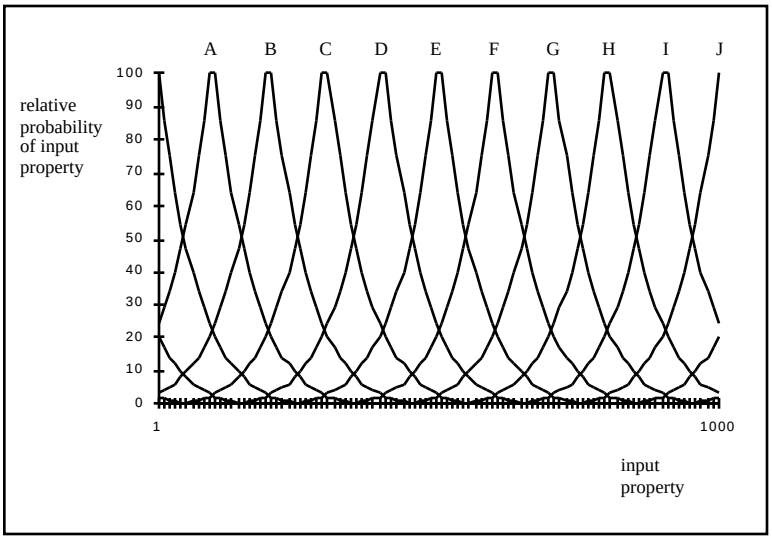

Figure 9. Probabilities of occurrence of properties in objects belonging to the ten categories

Although this scenario is much simpler that a real biological scenario it shares some key qualitative characteristics with the biological problem. Because a binary equivalent can be constructed to any form of input, sufficiently large binary conditions would be equivalent to biological inputs in information terms. The input space is fairly large, and as discussed below the results scale to much larger input spaces. The relevant conditions for determining the category of an object are complex combinations of inputs, and the nature of the relevant combinations are unknown a priori to the system. All overall input conditions are different, and any one overall condition is only experienced once. The simulated problem is thus similar to the biological problem in these information terms and any proposed cognitive architecture must be able to solve this equivalent problem.

The purpose of the simulation is to demonstrate that the algorithms and management processes described can result in a subsystem which can compress a large input space into a smaller output space with outputs which discriminate effectively between significant differences in the input space. In a real system there would be a series of subsystems of this type receiving inputs from the preceding subsystem and generating outputs for different behavioural purposes.

16 High level algorithms

The clustering module organized the input conditions into a hierarchy of repetition. This hierarchy included devices which could be imprinted with an information condition, layers of devices, and clusters made up of several layers. The outputs from the clustering function were the outputs of devices in the final layer of each cluster. These outputs were presented to a competitive function with access to some correct/incorrect feedback to generate behaviour appropriate to the different input conditions. This feedback took a number of forms. In one form of feedback, called regular, positive feedback was given if the selected category was correct, negative if it was incorrect. In a second form, no feedback was provided after feedback had been given for a certain number of presentations, to see if the behaviour continued undisturbed.

1000 individual input conditions of each category were created by randomly selecting properties from a set in which they occurred in proportion to the probability distribution for the category. 210 random selections were made to generate each input condition. Duplicates were discarded, resulting in an input condition having between 126 and 160 properties, with an average of 142. This process resulted in input conditions which were all different. One objective of the simulations is to determine whether the architecture can heuristically identify repetition conditions in its inputs which can usefully be associated with system actions under conditions in which no input condition ever repeats exactly.

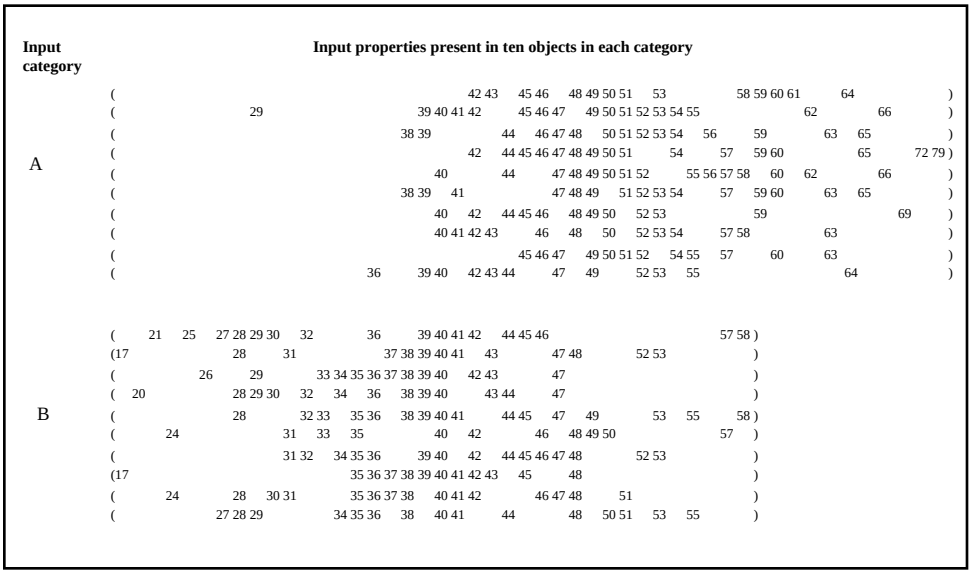

Figure 10. Examples of ten objects from each of two different categories constructed by random selection of properties from a 100 property input space. The numbers are the properties of the objects. The two categories had probability distributions of the form illustrated in figure 9.

At the highest level, a series of objects are presented to the system. After a number of presentations, a sleep with dreaming process occurs. The algorithm followed by the clustering subsystem during one presentation is illustrated in figure 11. If the series of presentations is the first series experienced by the system, there will not yet be any clusters, and information is recorded from each presentation to permit configuration of the first cluster. If any clusters already exist, each cluster is presented with the input and the device activity without recording of any additional combinations determined. If there is γ level output from at least one cluster, this output is passed to the competitive subsystem and no further process occurs. If there is no γ level output, device imprinting occurs in the cluster in which device activity in the β level exceeds a criterion by the largest amount. The γ output from this process is passed to the competitive subsystem. If there is no cluster in which β level activity exceeds the criterion, an unused cluster is imprinted if one is available and if its inputs contain a high proportion of the inputs favoured in the sleep with dreaming process which configured the cluster. If these conditions are not met, no output is generated, but

information about the input is recorded for use in configuring an unused cluster in the next sleep with dreaming process. However, if a criterion for activity in the α layer of at least one cluster is exceeded, the information will not be used to configure a new cluster, but in the next sleep with dreaming process the thresholds of β devices in that cluster will be reduced. Such presentation conditions will eventually be included in one of the existing clusters. This overlay process permits gradual expansion of the similarity conditions of clusters, encouraging a compromise between clusters which are too broad and proliferation of excessively narrow clusters.

The process followed in sleep with dreaming is illustrated in figure 12. In all existing clusters, three processes occur. Firstly, in α, β, and γ layers, unused virgin devices are deleted and new devices configured. The inputs to these new virgin devices are biased in favour of inputs which frequently contributed to firing of regular devices in the same layer of the same cluster in the recent past. This bias effectively reduces the amount of information exchange required between devices. Secondly, if a cluster has responded to less that a minimum proportion of presentations in the preceding wake period, regular α layer device thresholds are reduced. This has the effect of roughly equalizing the proportion of input conditions to which each cluster responds. Minimization of information exchange and roughly equal modules are both requirements for a simple functional architecture as discussed earlier. Thirdly, if the cluster has inhibited the creation of a new unused cluster without producing output in the preceding wake period, thresholds of its β layer devices are reduced. The sleep with dreaming process has been demonstrated to improve the correlation of clusters with categories in the input conditions and to reduce the total device and connectivity resources required [13].

In addition, a new unused cluster is configured if required. The inputs to β and γ layer devices in the new cluster are selected randomly, but for inputs to α layer devices the random selection is biased in favour of inputs which frequently occurred together in input presentations which did not result in an output in the preceding wake period. This cluster will not be imprinted unless the input condition contains a high proportion of these favoured characteristics.

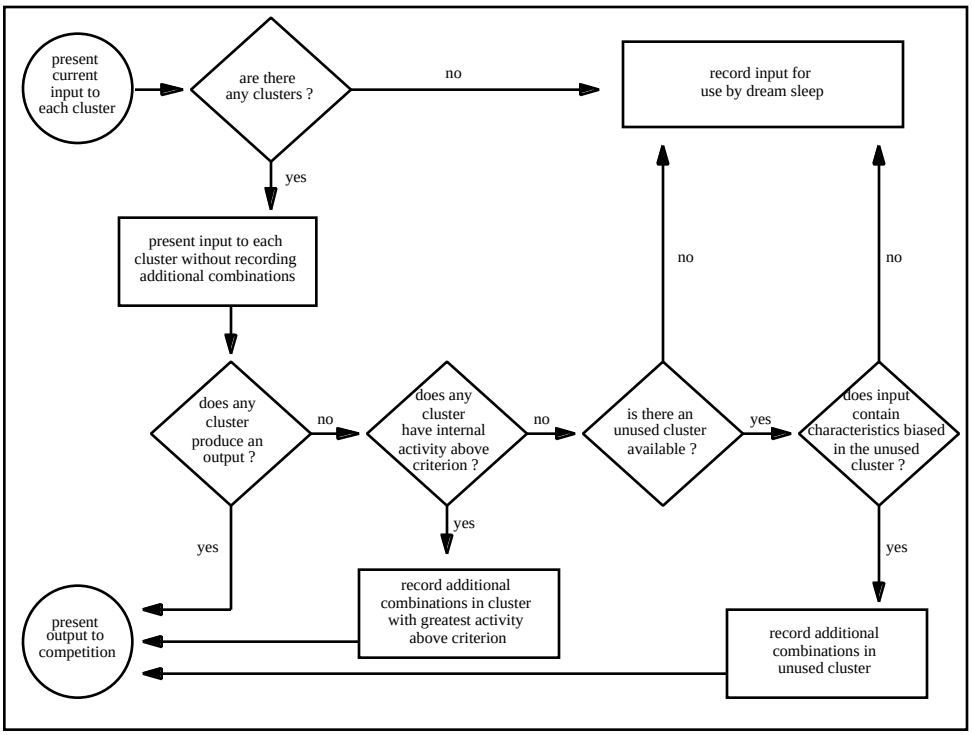

Figure 11. The process followed during the presentation of a single input condition to a simple (single layer) clustering subsystem.

The process of cluster generation continued until every level input produced an output (with or without device imprinting). Cluster outputs are used by a competition function to determine behaviour. To be effective for this purpose, although individual clusters are ambiguous they should correlate fairly strongly with different major functionally significant conditions in the input space, and outputs from those clusters should provide enough information richness to allow the functionally simple competitive function to resolve the ambiguous cluster outputs into a high integrity behaviour. In addition, there must be compression of information across a level of clusters, in

other words the number of γ outputs from a level should be much less than the size of the input space to the level. There are several problems to avoid in this generation process. One is creation of excessive numbers of clusters, and its opposite of too few clusters. Another problem is excessive use of resources, using more devices and connections than necessary.

17 Detailed algorithms

17.1 Clustering subsystem

The functional effectiveness of a set of clusters for a particular functional task depends upon a number of factors: the specification of the input space; the number of clusters and how they divide up the input space; and the information content of cluster outputs. The process of cluster creation is heuristic, and will vary with the input experience profile, and even with the same profile because of the random element in the device configuration process. The factors which determine functional effectiveness cannot therefore be specified directly, but can be influenced in a number of ways both by design and experience. Information derived from experience can be used to influence input space, number of clusters, and cluster outputs, but the better the design choices (or in a brain, the genetically specified parameters) the more effective and less complex the learning process.

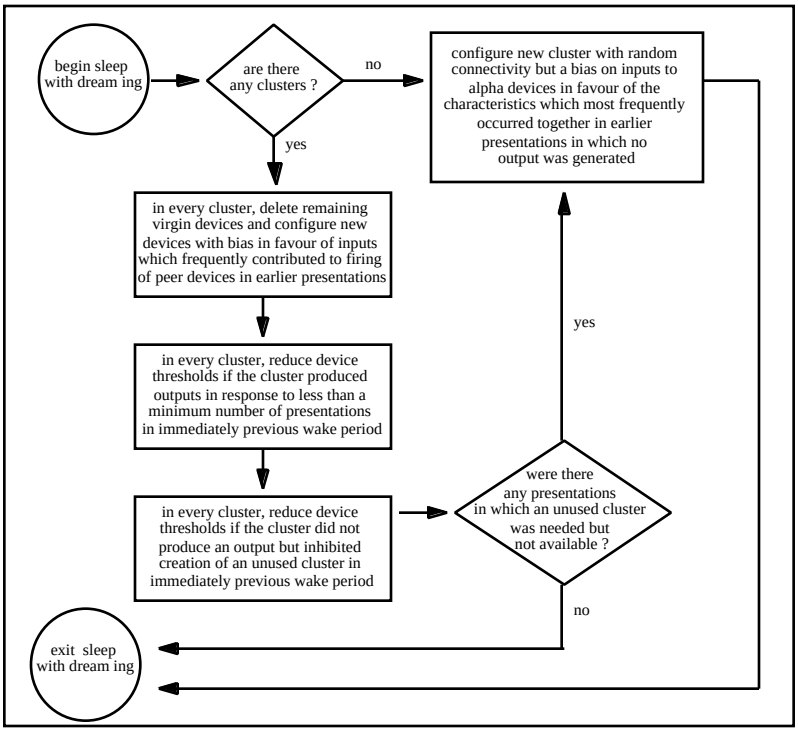

Figure 12. The process followed during sleep with dreaming in a simple (single layer) clustering subsystem.

The initial input space is strongly influenced by design, but α level neurons can in due course choose to accept inputs from sources which have often been active in the past at the same time as already imprinted neurons in the level were active. The number of clusters and the information content of cluster outputs are determined by a number of interacting factors including: 1. The degree of β activity required to generated imprinting in the absence of γ output; 2. The number of virgin neurons in each layer both initially and in subsequent sleep periods, the number of inputs assigned to each virgin neuron, and the statistical bias algorithm applied to the random input selection process; 3. The number of virgin neurons in each layer which can be imprinted in response to the initial experience and in response to any single subsequent experiences; 4. The thresholds assigned regular neurons at imprinting and how those thresholds are reduced if a cluster only generates outputs in response to very few input conditions; 5. How many presentations must separate the creation of two clusters.

The number of inputs given to a virgin device is determined by the number of random selections made, and by whether duplicate inputs were permitted. As in the simulations described earlier, no duplicate inputs were allowed, and the number of random selections made were:

Level	α	β	γ
Selections for initial cluster configuration in layer	30% of possible inputs	20	15
Selections for additional virgin devices in layer	20% of possible inputs	25	55

Elimination of duplicates meant that actual input count was somewhat less than the number of selections. For example, with an input space of 1000, the number of input selections for a virgin α device was 300, the actual number of inputs was typically in the range 240 - 280.

The number of virgin neurons configured in initial cluster configuration was typically set at 150 in the α and γ levels and 350 in the β level. During subsequent sleep periods all virgin neurons not imprinted in the previous wake period were deleted, and 40 -100 new virgin neurons were configured to be available in the next wake period depending on demand in the previous wake period. There was no limit on imprinting of α devices to generate first cluster output, and generally all 150 devices were imprinted. The limit on imprinting of β devices was 250, and that number were generally imprinted. If no β devices were imprinted in the first cluster imprinting, the cluster was deleted. The limit on imprinting of γ devices was 1. If the number of α and β devices imprinted at first cluster output was reduced, one effect was a substantial increase in the number of γ outputs in a mature cluster. The limits for device imprinting within a cluster in subsequent presentations were 30 for α and β layers and 1 for the γ layer. In general with the device numbers and input counts specified above the numbers imprinted in the α and β layers were less than these limits.

Imprinting would occur in a cluster in the absence of γ outputs if the number of β devices firing exceeded a few percent of the total number. However, it was found that if a simple percentage (e.g. 5%) were used, clusters tended to proliferate and the number of presentations required to generate a set of clusters which responded to every input also increased. A functionally effective set of clusters matured more rapidly with experience if initially imprinting occurred at a lower proportion which gradually increased with time. In other words, a cluster becomes more resistant to novelty as it matures. The algorithm used to determine the number of β layer devices which had to be active for imprinting to occur was:

$$\text{active } \beta \text{ devices} \geq (1/y) * (x^2/25 - 9x) \text{ with a minimum of one}$$

where x is the total number of regular devices in the β layer and y is the number of β devices imprinted when the cluster was first imprinted. Initially, this criterion is about 0.2% of the number of devices in the β level, increasing to about 5% of the larger number in a mature cluster. The number of active devices required could be reduced if there were many presentations in which the cluster inhibited creation of a new cluster but was not itself imprinted as discussed below.

In the imprinting process, the thresholds of virgin devices were gradually lowered until a cluster output resulted. The algorithm used attempted first to obtain an output by imprinting only γ devices, then only γ and β, then α, β and γ. The functional role of β devices in determining when imprinting would occur is enhanced by adding inputs to regular devices after initial imprinting. These inputs are provisionally added during sleep from virgin devices in the α level to regular devices in the β level in proportion to firing frequency. The connection became regular if the β level device fired when the virgin α device was imprinted. Most of the inputs which resulted in the β device firing had to derive from regular α devices for the connection to become regular. The additional inputs did not affect the threshold, so the outputs of the β devices in response to previously experienced conditions which produced an output is not changed.

The threshold of a newly imprinted regular device was set initially at one below its new total imprinted input count for α and β devices, and at 20% (rounded down to an integer) below for γ devices. The thresholds of α and β devices could be reduced in sleep if the cluster did not respond to a significant number of input conditions within a wake period. When initially imprinted, an α device had an input count which varied in practice over a range 20 - 60. In a mature cluster, thresholds had been reduced on average to about 50% of the initial level. At initial imprinting, a β device had an input count which varied over a range 8 - 20. In a mature cluster, some β device input counts could grow to up to 60, and thresholds had been reduced on average by about 15% of original value.

There are thus a large number of adjustable parameters in a clustering subsystem, such as the device numbers and connectivity, and the algorithms determining when learning occurs. Adjustment to these parameters can be by design prior to learning, or some parameters can be modified during learning in response to the system detecting that learning is not proceeding effectively. Ultimately, the algorithms by which such parametric changes occur must be specified by design. Design of a optimal system to learn a specific function would require tuning of all the adjustable parameters. The current values have been developed as a result of extensive simulation experiments with a wide range of input condition types. However, it is of interest to note that functionally effective clusters have been generated for a wide range of category types and input space sizes, with the same adjustable parameter values as described in this section except that the input counts to α devices varies in proportion to the size of the input space.

17.2 Competitive subsystem

The implemented competition subsystem was made up of a number of parallel pipes, one for each possible behaviour, as illustrated in figure 8c. Each pipe contained two layers of devices as in figure 8b. The devices are similar to classical perceptrons in having relatively fixed inputs with input weights which vary with learning. In the simulations, a fixed number of pipes were defined which corresponded with the categories present in the input. In a more sophisticated system outputs from clustering could used different combinations and permutations of the available fixed number of pipes.

Devices in the first layer had randomly selected inputs from the γ layer of clusters, and a reference threshold set at 100. The value of the device threshold under different input conditions was derived from this reference value as described below. The default value for the weight of an input was the reference threshold. However, this weight was changed by feedback. If feedback had influenced the weights of previously acquired inputs from the same cluster, the input weight given to a new input was set at the average value of the weights of the most recently imprinted outputs from the same cluster to the same pipe.

A second layer device received excitatory inputs from first layer devices in the same pipe, and inhibitory inputs from first layer devices in other pipes. Inputs to layer two devices were assigned at random. The default weights were the reference threshold value for excitatory inputs and the reference threshold value divided by the number of types of behaviour for inhibitory inputs. If feedback had influenced the weights of previously acquired inputs from the same pipe, the input weight given to a new input was set at the average value of the weights of outputs from the most recently created layer one devices from the same source pipe.

When the presentation of an input to the clustering subsystem resulted in some γ level output, the thresholds of all devices in the first layer of every pipe was varied by the same proportion until 25% of the devices (or all devices with an active input if this was less than 25%) in the first layer of at least one pipe were firing. Using these first layer outputs, the thresholds of all devices in the second layer of every pipe were varied by the same proportion until at least one pipe was generating an output. The behaviour corresponding with the pipe with the largest number of active second layer devices was selected as the behaviour in response to the input object. If more than one pipe had the same maximum output, one was selected at random.

Consequence feedback provided the information that the selected behaviour was either good or bad. If good, the weights of all active inputs to active first layer devices in the pipe corresponding with the selected behaviour were increased, and the weights of active inputs to active second layer devices in the same pipe were increased if excitatory and decreased if inhibitory. Changes in the opposite direction were made to the weights of active inputs to active devices in all other pipes. If consequence feedback were bad, the same changes but in the opposite direction were made in all pipes with active devices. The changes made in response to bad feedback were larger than the changes made in response to good feedback. Input weights to first layer could only be positive, and varied within a limited range, while input weights to the second layer were unconstrained.

In the simple category recognition simulations, there were ten pipes, one for each of the input condition categories. Each pipe had two layers with sixteen devices in each layer. Inputs to devices in the first layer were randomly selected outputs from γ devices in all clusters in a level, and inputs to layer two devices were randomly selected outputs from layer one devices in all pipes. Selection probability for any input to any device was 0.5.

The initial weight assigned to an input was +100 for inputs to layer one devices, +100 to inputs to layer two devices from layer one devices within the same pipe, and -10 for inputs to layer two devices from layer one devices in a different pipe. All device thresholds were set at a high nominal value which was the same for all pipes. When an input to the competitive subsystem was received, layer one device thresholds were lowered from this nominal value until at least 25% of the layer one devices in at least one pipe were active. The nominal threshold was then lowered in the layer two devices until there were active devices in layer two of at least one pipe. The pipe with the largest number of active layer two devices was the category selected. If more than one pipe had the same total output, one was selected at random.

Consequence feedback acted on all active devices in both layers. The weights of active inputs to active devices were increased by 10 times a consequence factor if the category selected was correct, and decreased by 20 times a consequence factor if the category selected was incorrect. Individual input weights to layer one devices could only vary between 0 and 200. In layer two, initially positive weights could not become less than zero, and initially negative weights could not become greater than zero. The consequence factor was a controllable variable which determines the proportion of the initial weight by which a weight changes in response to consequence feedback. It was set at either ± 1 or ± 5.

Two types of competitive subsystem runs were performed on the outputs from each of the nine clustering run outputs. In both types, the first 4100 object presentations developed a cluster structure before outputs or consequence feedback were provided to the competitive subsystem. After this initial period, the system configured the competitive subsystem and generated a category identification for the rest of the presentations. Consequence feedback was applied either for the rest of the presentations or was discontinued after 2000 presentations.

It is important to note that in contrast with many neural network approaches, in the recommendation architecture there are no separate training and operational phases. The recommendation architecture can begin to generate some behaviours in response to early experiences, and there is no point at which learning is discontinued.

18 Results of simple category recognition simulations

18.1 Clustering results

A series of clustering runs of three types were performed. In the first type, there were 76 wake periods separated by sleep. In each wake period ten objects from each of the ten categories were presented, so there were a total of 7600 presentations. All 7600 objects were different. In the second type of clustering run, there were 61 wake periods in which ten objects from each of eight categories were presented, followed by 25 wake periods in which ten objects from each of the ten categories were presented. There were thus a total of 7380 presentations. Again, all 7380 objects were different. The purpose of the two types of run was to investigate the effect of new learning on already learned behaviours. In the third type of clustering run the experience and sleep profiles were identical with the first type, but during the sleep following presentation 4100, functionally duplicated γ layer devices (i.e. devices which were always active at the same time) were eliminated. Configuration of the competitive function did not commence until after that point, and there were therefore no issues with maintaining information context.

	Numbers of active gammas in each cluster during final 50 object presentations				
cluster #	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
category of object presented	A F	B G	C H	D I	E J
numbers of gammas active for set of ten objects of type A through J	(0 0 0 0 30) (6 0 0 0 0)	(0 6 0 0 0) (18 0 0 0 0)	(0 22 0 0 0) (6 0 0 0 0)	(0 6 0 0 0) (0 0 14 0 0)	(0 0 0 14 0) (0 0 6 0 0)
numbers of gammas active for second set of ten objects of type A through J	(0 0 0 0 29) (6 0 0 0 0)	(0 6 0 0 0) (18 0 0 0 0)	(0 22 0 0 0) (8 0 0 0 0)	(0 7 0 0 0) (0 0 13 0 0)	(0 0 0 19 0) (0 0 6 0 0)
numbers of gammas active for third set of ten objects of type A through J	(0 0 0 0 24) (5 0 0 0 0)	(0 4 0 0 0) (14 0 0 0 0)	(0 22 0 0 0) (8 0 0 0 0)	(0 5 0 0 0) (0 0 12 0 0)	(0 0 0 15 0) (0 0 6 0 0)
numbers of gammas active for fourth set of ten objects of type A through J	(0 0 0 0 29) (4 0 0 0 0)	(0 5 0 0 0) (12 0 0 0 0)	(0 21 0 0 0) (8 0 0 0 0)	(0 7 0 0 0) (0 0 12 0 0)	(0 0 0 18 0) (0 0 6 0 0)
numbers of gammas active for fifth set of ten objects of type A through J	(0 0 0 0 21) (5 0 0 0 0)	(0 6 0 0 0) (13 0 0 0 0)	(0 21 0 0 0) (10 0 0 0 0)	(0 5 0 0 0) (0 0 13 0 0)	(0 0 0 14 0) (0 0 7 0 0)

Figure 13. For one run, the number of γ devices which were active in each cluster in response to the final fifty presentations of objects from the ten categories are shown. For example, the top left result (0 0 0 0 30) indicates that for the one A object, thirty γ devices were active in cluster five, and no γ devices in any other cluster.

At the end of the clustering runs, the number of clusters was less than the number of categories, varying from five to eight. The information compression factor averaged 6.6. Compression was somewhat higher when functionally duplicated γ devices were eliminated at an intermediate point, and somewhat lower when new categories were introduced part way through a run. Figures 13 and 14 show more detail of the final five sets of presentations of one object from each category A through J in one of the runs. The numbers of γ devices active in each of the five final clusters are shown in figure 13. It can be seen from figure 13 that cluster one generates outputs in response to objects of type F, G, and H; cluster two in response to B, C, and D objects; cluster three in response to I and J objects; cluster four in response to E objects; cluster five in response to A objects. The clusters are therefore partially ambiguous with respect to the categories.

The actual γs active in cluster two outputs in response to the final five B, C, and D objects are shown in figure 14. It can be seen from figure 14 that for categories B and D there are no γs which are only active when an object

belonging to the category is present. In other words there is ambiguity even at the device level. However, it can also be seen that there is enough difference between the combinations of active γs to distinguish between objects of the three different categories. As discussed below, differences of this type are adequate for a functionally simple competitive function to develop a high integrity category specific behaviour. The differences are clearer than those between input vectors for different categories as seen in figure 10. There is some functional duplication in the γ devices. In other words, some γ devices are always active at the same time as others. In a functionally complex system this duplication can only be eliminated before the information is used by some other function (another cluster, or a competitive function). Once the information is being used externally, duplication would require massive revision to connectivity which will in general be impractical.

	Gamma devices active in same cluster in response to one object
Gamma device number	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Gamma devices active in response to five objects in category B	16 17 18 19 20 21 16 17 18 19 20 21 17 18 19 21 16 17 18 19 21 16 17 18 19 20 21
Gamma devices active in response to five objects in category C	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22
Gamma devices active in response to five objects in category D	4 11 12 14 18 22 4 11 12 13 14 18 22 4 11 12 14 22 4 11 12 13 14 18 22 4 11 12 18 22

Figure 14. For the presentations shown in figure 13, the number of γ devices which were active in cluster two in response to the final five presentations of objects from the categories B, C, and D are shown. There were a total of 22 γ devices in the cluster.

The key conclusion from these results is that significant information compression can be achieved in such a way that the outputs provide more usable discrimination between different categories in the input conditions than the input conditions themselves. It is probable that a greater and more consistent compression could be achieved by tuning the controllable parameters for the type of input space. There is a need for further work in this area. It should be pointed out that if the size of the input space is increased, and the size of the input conditions increases at the same rate, the compression achieved increases faster than the size of the input space. When the input space is expanded to 10,000 under these conditions a compression by a factor of several hundred is achieved, with functionally adequate output information discrimination.

18.2 Competition Results

As described earlier, the competitive subsystem was operated with two different consequence factors, and in addition, consequence feedback could be discontinued after 2000 presentations. These two options resulted in four different competitive algorithms which were applied to the outputs of each clustering run. Two parameters indicating the effectiveness of learning are reported here: one is the average percentage of correct category identifications in the final 1000 presentations; the other is the rate of improvement in correct identifications when consequence feedback is first applied.

As can be seen from figure 15, for all types of clustering and competitive conditions, correct recognitions in the final 1000 presentations ranged from 79.7% to 99.7% with an average of 94.9%. None of the controllable parameters made a consistent difference to the recognition accuracy. As can be seen from figure 16, recognition accuracy reached 80% within the first 500 presentations after consequence feedback was provided. The magnitude of the consequence factor had no significant effect on the rate of early learning, but early learning was somewhat slower when functionally duplicated neurons were eliminated at an earlier point, although final accuracy achieved was the same.

The ability of the system to retain existing learning when new categories are introduced can be seen from figure 17. Prior to the introduction of the two additional categories, recognition of the eight exceeded 95%. Immediately after the introduction of the categories, there was on average a 5% drop in accuracy of recognition of the original categories. The slower learning rate for new categories reflects the need to expand existing clusters or add new clusters to generate the information for the competitive function to use.

Learning conditions	Accuracy
Consequence factor = 1	95.5%
Consequence factor = 5	94.2%
Consequence feedback to all presentations	95.5%
Consequence feedback to only 2000 presentations	94.2%
Without elimination of functional duplicates	94.7%
With elimination of functional duplicates	95.8%
Ten categories present initially	94.7%
Ten categories present initially, two added later	93.8%

Figure 15. Accuracy of category identification achieved in last 1000 presentations under different conditions. All results were obtained by averaging over a number of clustering runs.

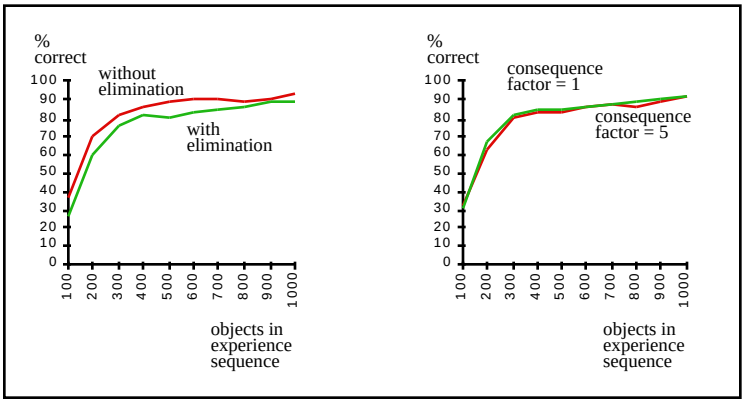

Figure 16. Increase in category identification accuracy at the start of learning. Note that clusters were well established when consequence feedback commenced.

As pointed out earlier, the mechanism in a real system for acquiring a behaviour in response to a novel type of condition is addition of a new module or pipe in one or more competitive subsystems. This module receives as primary inputs the clustering outputs which tend to occur at the same time and which generate negative consequences in response to any existing behaviour. To understand how this process works in principle, suppose that the behavioural objective is to speak the name of a previously unexperienced category, and that the appropriate sound is available for imitation. The clustering outputs can thus target currently active muscle movements, and the new module can gate the release of these clustering outputs.

If there were only probabilistic association between input conditions and rewards, the system would settle long term on the behaviour most likely to generate a reward subject to two provisions. One is that the effect on connection weights when a reward is absent (i.e. the punishment) is not disproportionate to the opposite effect when the reward is present. The other is that if punishments are not major there is an inbuilt tendency for the system to experiment with different behaviours (i.e. exhibit curiosity).

The competitive subsystem results thus demonstrate that such a subsystem can use the ambiguous outputs of a clustering subsystem to generate high integrity behaviour. Perhaps the key result is the demonstration that this type of system can learn to recognize new categories with minimal interruption to the recognition of already learned categories. This is in contrast with other neural networks in which major relearning of both old and new categories is required when new categories are introduced. The algorithms for the competition subsystem are robust with respect to variation of adjustable parameters, the variation in recognition accuracy relates more to the type of clustering achieved by the clustering subsystem. More work is needed to understand what adjustable parameters have the greatest effect on the usability of clustering outputs.

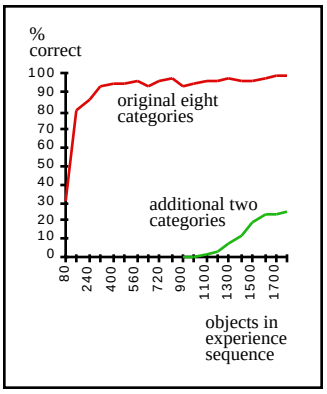

Figure 17. Recognition of categories already learned is only slightly affected when new categories are introduced.

19 Comparison with statistical clustering techniques

It is of interest to compare the approach in the recommendation architecture with clustering as defined statistically, in which a large number of sets of observational data are divided into meaningful groups (or clusters) based on some similarity criterion. In statistical clustering analysis the number of clusters and other information about their composition may be unknown [25]. Some statistical clustering methods are hierarchical and proceed by producing a sequence of partitions of the data space corresponding with different numbers of clusters. The process can be one of merges among a relatively large initial set of clusters or splits among a relatively small initial set. Other clustering methods involve relocation of data sets between a fixed number of initially specified clusters [16].

There are four major differences between statistical clustering analysis and clustering as defined in the recommendation architecture. These differences are in the areas of data availability, indication of data set membership in a cluster, changes to the indications of data set membership, and membership of multiple clusters.

In statistical clustering analysis, the same data sets are available for repeated comparison with the developing cluster set. In the recommendation architecture, each individual data set is available only once and only information recorded about that data set at the time is in any way available for later processing. In statistical clustering analysis, membership of a data set in a cluster is binary (true or false), although a numeric indication of membership probability may be available in some approaches. In the recommendation architecture the indication of membership is the particular combination of outputs generated which can contain considerable information about where within the similarity domain defined by the cluster the current input is located.

In statistical clustering analysis, the binary or numeric indication of cluster membership can change as clusters are reorganized and redefined. In the recommendation architecture, once a given subset of a data set has generated an indication of membership in a cluster, the same indication will always result if the same subset occurs in any subsequent data set. Finally, in statistical clustering analysis an individual data set is assigned to only one cluster, while in the recommendation architecture data sets can generate outputs from multiple clusters.

The significance of these differences is that while statistical clustering analysis is very effective for the problem domains at which it is targeted, it is not effective for heuristic real-time management of a complex combination of functionality. As discussed above, in a system which learns to perform a complex combination of functions, components must determine the portfolio of input information conditions to which they will respond on the basis of a sequence of input conditions which never repeats exactly. The presence of a programmed information condition must be indicated with enough information richness that other components can use the output information effectively, and because any component could choose to use an output at any time, once an output has been generated, the same output must always result from the same input condition.

20 Application of the approach to a more complex problem

To further investigate the capabilities of the recommendation architecture a somewhat more complex problem was defined. The system was presented with sequences of binary vector inputs simulating sensory inputs resulting from visual objects and speech. Six categories simulating colour (labeled red, green, brown, black, white, and clear) were constructed in a 600 bit binary space. Eleven categories simulating shape (labeled cup, coffee, tree, grass, book, pen, paper, water, chair, apple, and car) were constructed in a 1100 bit space. The visual objects simulated were shapes with a specific range of possible colours for each shape. Thus cups could be red or white; coffee could be brown or black; trees were always brown; grass was always green; books could be red or white; pens brown or black; paper always white; chairs red, white or green; apples red or green; and cars could be brown or black. Twenty-one categories simulating sounds (labeled R, G, B, W, C, T, P, H, S, L, D, E, A, I, O, U, F, K, N, CH, and M) were constructed in a 2100 bit input space. Words were simulated as sequences of three sounds, and the system was given inputs corresponding with the words for the colours and shapes.

All inputs were binary vectors constructed by the same method as described in section 15. Although the inputs are only simulated, they are similar in information terms to sensory inputs in the sense that a binary equivalent can be constructed to any form of input, the input space is fairly large, the relevant conditions for determining the category of an object are complex combinations of inputs, and all input conditions are different so that any one condition is only experienced once.

The system was given a series of inputs which simulated different combinations of visual and speech conditions and was required to find solutions to the following tasks. Firstly, given only the series of visual and speech inputs, find repetitions in those inputs. Secondly, given only binary positive/negative feedback, associate combinations of repetitions found in the visual inputs with actions which include speaking the name of a visual condition (which includes both colour and shape) and taking an appropriate action. The appropriate action was determined by the sequential presence of two objects. For example, red-cup followed by black-coffee was the appropriate condition for a drink action. The condition, action pairs which received positive feedback are listed in table 1.

Combination of input conditions	*Target action*
cup + coffee	drink
cup + water	drink
book + anything	read
pen + paper	write
apple + anything	eat
chair + anything	sit
car + anything	drive
tree + grass	walk
anything else	no action

Table 1. Conditions under which positive consequence feedback was applied

The third task was to associate combinations of repetitions found in speech inputs with appropriate visual conditions by activating the appropriate visual condition when the word was heard. The objective was to generate the appropriate action when a visual object was followed by a pair of words corresponding with another visual object.

The fourth task was to associate combinations of repetitions currently present in visual inputs with visual conditions often present in the past when repetitions similar to the current set were also active, and to activate those past repetitions. This activation would be similar to the activation actually generated by the earlier visual object. The objective was to generate the actions of speaking the words corresponding with objects often present in the past at the same time as the current object. The functional role of such pseudo-visual activations is to expand the range of information available to determine the currently appropriate behaviour [8].

The architecture implemented to achieve these tasks is illustrated in figure 18. The cluster sets are heuristically defined, with the same parameters and conditions as those described in section 17.1. All clusters were three layer (α, β, and γ layers). Competitive subsystems 1 and 2 are as described in section 17.2. They take their inputs from the γ layers of the clusters. Competitive subsystems 3 and 4 are more simple, with only excitatory connections. Competitive subsystem 3 takes its inputs from the γ layer of clustering set 5 and generates actions to activate devices in the β layers of cluster sets 1 and 2. Competitive subsystem 4 takes its inputs from the β layer of cluster set 3 and generates actions to activate devices in the γ layer of cluster sets 1 and 2. The reason for these differing sources and targets is that in the case of words, the objective is to associate a very specific combination of sounds with the typical repetitions which have been present in the past when the sounds were present. Hence the most complex repetitions corresponding with sound combinations are associated with moderately complex visual

combinations. In the case of temporal association, the objective is a less specific association, and repetitions likely to occur under a wider range of conditions are selected to activate associated repetitions. Performance is better with this connectivity.

Note that if it was required that the activation generated in response to one input condition be retained active for a period in order to be clustered with other information, a very similar mechanism to competitive subsystem 5 could meet the requirement. This mechanism would keep active any devices in the β level that were currently active. This function would be a working or short term memory.

In the implemented version of the system, the outputs from cluster sets 1 and 2 for the first and second objects were treated as independent inputs into cluster set 3, and any knowledge of similarity between the two objects derived from the first clustering was discarded. This discarding was necessary in order to distinguish between the first and second objects in cluster set 3.

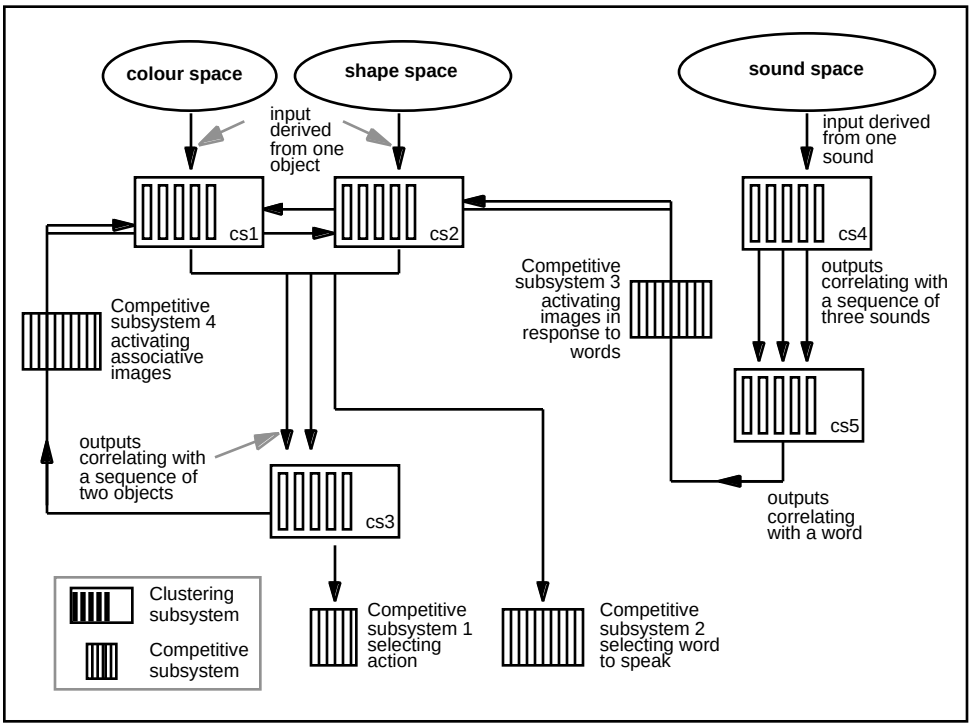

Figure 18. Architecture of a system which can learn to associate words with visual objects, and imagine objects which are not present.

21 Results with the more complex problem

The experience of the system illustrated in figure 18 consisted of the presentation of 6500 emulated visual objects and 9120 emulated words. No two objects or words were identical in information terms. At the end of these experiences, the clustering subsystems had functional modules on five levels: 5 cluster sets, 74 clusters, 222 layer modules, 40 thousand devices and 800 thousand connections. The number of cluster sets was specified by design, but the connectivity between these sets, and the number and connectivity of modules on all other levels was defined heuristically. The number of devices would be lower if individual devices supported more than one repetition, but this functionality was not implemented. The competitive subsystems were much less complex, with a total of around 500 devices and a much simpler pattern of connectivity.

The outputs from cluster set 3 are essentially a compression of the inputs derived from the colour and shape information derived from two objects, i.e. a 3400 binary input space. There were 234 devices at the output level of cluster set 3 once learning was essentially complete. Hence the information compression through the two stages was by a factor of 14.

As in the category learning system described earlier, competitive subsystem 2 used simple correct/incorrect feedback to associate the outputs of clustering sets 1 and 2 with the appropriate shape and colour conditions. The output of competitive subsystem 2 could therefore be interpreted as speaking the words corresponding with currently present conditions. The accuracy achieved by competitive subsystem 2 in a typical run is similar to that achieved in the category runs described earlier, generally well over 90%. Competitive subsystem 1 used consequence feedback as

shown in table 1 to associate combinations of cluster 3 outputs with actions. As can be seen from figure 19, correct responses climbed to about 80% within the first few hundred object pair presentations and remained at that level. The system thus demonstrates the capability of the recommendation architecture to learn behaviours appropriate to the presence of individual and sequential pairs of simulated objects.

The effectiveness of competitive function 3 in using cluster set 5 γ outputs to activate β devices in cluster sets 1 and 2 was demonstrated as follows. In the two object inputs to cluster sets 1 and 2 which generated the set of inputs to cluster set 3, one object input to cluster sets 1 and 2 was replaced by outputs from cluster set 5 generated by the two corresponding words, and it was found that the outputs frequently generated the appropriate behaviour. For example, a brown-tree object combined with green and grass words generated a walk output from competition set 2 about 30% of the time. For most of the other tests with this combination the system decided to take no action. In other words, the word outputs from cluster set 5 often generated actions similar to those generated by the actual (simulated) visual object.

As an example of the effectiveness of the links from series of inputs were provided to cluster sets 1 and 2 which included presentations of red-cup followed by black-coffee. The association between cluster set 3 outputs and activation of cluster sets 1 and 2 γ outputs was established in this period. Inputs corresponding with red cup were then presented to regions 1 and 2, and the outputs from cluster set 3 activated γ devices in cluster sets 1 and 2. This process generated activations corresponding with black-coffee in all such presentations. In other words, red-cup "reminded" the system of black-coffee on the basis of recent experience.

This system thus demonstrates the ability to heuristically organize inputs simulating visual and sound sensory inputs into a hierarchy of repetition of considerable functional complexity, in such a way that it can determine appropriate responses to its inputs, including actions in response to objects and sequences of objects, generation of appropriate pseudo-visual activations in response to words, and generation of pseudo-visual activations corresponding with objects which have often been present in the past at the same time as the currently perceived object. Work is under way to compare the performance of this system in detail with the phenomenology of implicit and explicit memory.

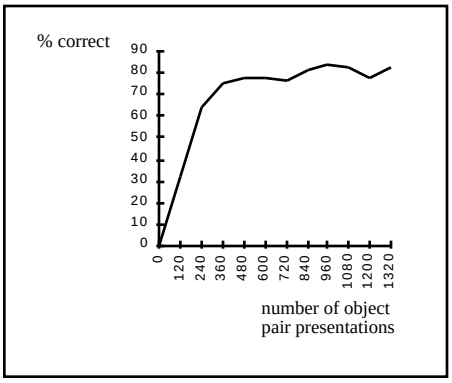

Figure 19. Learning of correct responses to pairs of objects

Given the architectural constraints on context for information exchange, one interesting question is how the performance of the system could be further improved. There are several directions in which improvement could be achieved within the constraints. A priori knowledge of the problem domain could be used to define initial connectivity and to improve the algorithms which heuristically evolve the conditions under which clusters are created and under which clusters learn. For example, in preliminary simulations it was found that if the colour and shape information are combined and repetition organized in a single set of clusters, the effectiveness with which the system determines behaviour is considerably reduced. A second possibility uses consequence information in ways which do not affect information context. One such option would be if clusters were created and tested on a behaviour in an isolated domain, then discarded and new clusters created using the knowledge gained. The critical requirement would be that the distribution of the outputs of the original outputs would have to be severely limited. Another option is less constrained. It is possible to add outputs in response to a condition provided that the original outputs continue. It would therefore be possible to detect that the same outputs from a cluster were being interpreted in the same way by a competitive subsystem but resulting in contradictory consequences. Detection of such a condition could be used to cause the cluster to accept additional input information and generate additional outputs even when an output was being generated. The additional outputs could be used by the competitive subsystem to discriminate between the similar but functionally different conditions.

22 Comparison with other cognitive architectures

There are a number of major differences between the recommendation architecture approach and other cognitive modeling approaches, which can be classified as architectural differences, algorithmic differences, and capability differences. However, the most fundamental difference is in the functional architecture perspective. In a typical neural network approach, “architecture” is defined in terms of device algorithms, number of layers of devices, and the pattern of connectivity. Architectures of this type are created, and experiments performed with a range of problem sets. Adjustable parameters are tuned to optimize performance for a particular function such as feature or category recognition. The typical approach to more complex problems is to develop networks of networks, describing the organization of the network in terms of feature combinations [e.g. 1]. Even when modules are organized into hierarchies, different levels in the hierarchy are interpreted as detection of features of different complexity [e.g. 32].

In the functional architecture approach, the starting point is to specify the complete functionality of the system at high level. This functionality is then separated into components, these components into subcomponents and so on, always within the constraints of approximate operational equality of all the functional modules on one level, minimization of information exchange between modules, and a maintained context for all information exchanged. In the functional architecture approach, these module and information requirements largely determine the device algorithms, number of layers of devices, and the pattern of connectivity.

The coordination of large numbers of interacting functions depends upon providing and maintaining a context for partially ambiguous information exchange. The typical neural network approach targets module outputs on specific vectors and does not provide an inadequate information richness for partially ambiguous communication and in fact drives such outputs towards unambiguous category identifications. For example, a typical problem attempted by Kohonen Nets is self organization of natural language information in which map outputs indicate individual words and relative position on the map indicates the syntax of the word [e.g. 22]. The use of device algorithms in which relative input weights are changed makes it difficult to maintain information context as discussed earlier. There is therefore a reliance on exchange of unambiguous feature identifications to coordinate different modules, an approach which the earlier discussion has demonstrated leads inevitably to some form of the memory, processing architecture once functionality becomes sufficiently complex.

The functional partitioning in the recommendation architecture leads to a hierarchy of functional signals which determine when learning will occur. The only comparable other example is in Adaptive Resonance [6] which makes use of an indication of input similarity as a precondition for learning. This is a single higher level functional signal, but multifunctional, multilevel functional signals comparable with those used in the recommendation architecture have not been developed.

There are significant algorithmic differences from neural network algorithms used in other approaches. Within a competition subsystem the device algorithms are similar to the widely used perceptron algorithm, although the higher level structure of parallel pipes corresponding with different behaviours, with learned inhibitory signals between the pipes is not used elsewhere. The instantaneous, permanent recording of a currently present information condition used in the clustering subsystem is unique to the recommendation architecture, although in general the processes used in the clustering subsystem could be regarded as a form of competitive learning [21].

In terms of capability, the recommendation architecture derives all its functional knowledge from its experience, unlike approaches like ACT [2] in which such functional knowledge is used explicitly in the design process. The recommendation architecture can handle extremely large and noisy input spaces under the condition that no input ever repeats exactly. New behaviours can be learned without significant effect on previously learned behaviours, an extensive relearning is not required. This is in contrast with other neural networks in which a learning phase is separated from an operational phase with very limited learning possible in the operational phase.

As a result, other approaches can only be applied either to stable, functionally simple problems or under conditions in which significant explicit functional knowledge is supplied by a designer. An example of a stable, functionally simple problem is the application of neural networks to recognition of features in MRI brain scans [31]. Such problems are algorithmically complex but functionally simple in that system outputs do not dynamically change system inputs in real time. They are stable in that any required change to the problem such as recognition of new types of feature would require an extensive relearning process including previously learned features.

It is of interest to compare the recommendation architecture to the work of Marr on cognitive architectures [26]. In his model of the neocortex, Marr supposed that there is a potentially infinite number of input patterns which can be grouped into a number of different classes. He used a class of type "poodle" as an example. As discussed in work to simulate Marr's model [39]:

"Each single event that represents a different occurrence of a poodle contains certain, but not all, of the features of the poodle class.....Two events that represent different poodles have many features in common - many more than the number of features shared by an event representing a poodle with one that does not. Marr referred to such clusters of events over the set of input fibres as *mountains*. The simplest problem that can be considered is where a network has a single output cell, and it has to learn to distinguish between the inputs from two mountains, occurring

equiprobably.....The problem is to find values for the weights and the threshold such that the outputs cell responds to events from one mountain only."

In Marr's approach the implicit objective is to link cell outputs with functionally unambiguous "mountains" of similarity in a sequence of input conditions, and indicate the presence of such mountains by an unambiguous output (in the example, the output from one cell). In the recommendation architecture, the system defines clusters of repetition analogous with Marr's mountains, but these clusters do not correlate unambiguously with functionally relevant categories of input conditions. Instead, the outputs indicating the presence of the ambiguous "mountains" have enough information richness that they can be combined with outputs from other mountains to generate high integrity behaviours (simple examples would be category identifications). Once the objective becomes "to find values for the weights and the threshold such that the outputs cell responds to events from one mountain only" the system is being implicitly designed using unambiguous information contexts, and for a sufficiently complex functionality will either be unrepairable and impossible to modify or will be forced into the von Neumann architecture.

23 The theory of knowledge implicit in the recommendation architecture

Barsalou [4] has emphasized the differences between what he labels amodal and perceptual theories of knowledge. In perceptual theories, perceptual states arise in sensory-motor systems. In Barsalou's description "Perceptual symbols are modal and analogical. They are modal because they are represented in the same systems as the perceptual states that produced them. The neural systems that represent color in perception, for example, also represent the colors of objects in perceptual symbols, at least to a significant extent. On this view, a common representational system underlies perception and cognition, not independent systems. Because perceptual symbols are modal, they are also analogical. The structure of a perceptual symbol corresponds, at least somewhat, to the perceptual state that produced it". Amodal theories, on the other hand, model higher cognitive states with representational schemes which are inherently nonperceptual. Perceptual states must be transduced into a completely different representational scheme that describes these states amodally.

The recommendation architecture is a perceptual theory [8]. However, it differs significantly from the perceptual theory developed by Barsalou [4]. In Barsalou's theory, "a perceptual symbol is the record of the neural activation that arises during perception". In the recommendation architecture, cognitive processes are the manipulation of different records of neural activation within a clustering subsystem, but during the manipulation process, no activated record can be unambiguously identified with a cognitive symbol. All cognitive processing is with ambiguous "representations" and ambiguity is only removed once an output is generated via a (functionally simple) competitive subsystem. If such an output acts back on the clustering subsystem, the resulting activations are ambiguous within that subsystem.

The closer the correlation between a neural activation within the clustering subsystem and a cognitive feature or category, the easier it is for the competitive subsystem to determine behaviour. If there is a category of objects with (perceptually) very little in common with other categories, the presence of the neural activation may correlate very closely with the cognitive feature or category. However, the critical point is that the system cannot internally distinguish these close correlation conditions from weaker correlation conditions. The system must therefore be able to cope with the presence of ambiguity in all circumstances, even though the less the ambiguity the more easily the competitive subsystem can make effective use of clustering subsystem outputs.

Barsalou emphasizes the distinction between a recording system and a conceptual system, and points out that perceptually based theories of knowledge "are typically construed as recording systems". The key feature of the clustering subsystem in the recommendation architecture is that it makes records of experience in a form which can be manipulated to generate high integrity behaviour. However, as the discussion in section 3 demonstrated, it is impossible in practice to heuristically create records which are both adequate to support a complex functionality and functionally unambiguous. The implementation of the recommendation architecture described earlier demonstrates the creation of ambiguous but functionally usable records. Barsalou's model implies the creation of functionally unambiguous records, but as he states, "The properties of [the] theory [are] not characterized formally, nor [are] they grounded in specific neural mechanisms". Any attempt to characterize formally for a functionally complex system would encounter the problem of maintaining an adequate context for information exchange.

Barsalou [4] uses the example of a car to illustrate the process of storing perceptual symbols; "As one looks at the car from the side, selective attention focuses on various aspects of its body, such as wheels, doors, and windows". The problem with this approach is that somehow the selective attention function has to determine how to limit attention to a domain defined by a cognitive feature. This is the same problem as discussed earlier with Marr's architecture, finding groups of repetitions of information which correlate unambiguously with cognitive features. The approach in the recommendation architecture is that the repetition conditions will not correlate exactly with wheels, doors etc. but every currently present repetition condition will be indicated with enough information

richness that the combination can be interpreted unambiguously by a competitive subsystem whenever a behaviour (such as identifying a feature verbally) is required.

During cognitive processing of a combination of perceived conditions, for example to determine the make of the car, the various conditions separately perceived in sequence are represented by the ambiguous indications, these ambiguous indications are combined by the cognitive processing and the output from the combination is interpreted by the competitive subsystem as an action such as speaking the make of the car. In other words, no communication, interaction or activation within the clustering subsystem can be viewed as unambiguously indicating a cognitive category. Such clustering activities only become behaviourally unambiguous when interpreted through a competitive subsystem. However, because such interpretation always occurs before clustering information can be expressed externally, the ambiguous information within the clustering subsystem is inaccessible directly.

Barsalou [4] comments that within his proposed cognitive system, "simulations of cognitive conditions are always partial and sketchy, never complete". In the recommendation architecture the direct perception of an object and the reminiscence of a previously perceived object activate populations of information combinations, some recorded previously, some recorded as part of the process of activating the population. The information combinations recorded do not necessarily include all the available input information so in that sense in common with Barsalou the activations in response to perceptions are always partial. However, it is important to emphasize that this is not the same as partially ambiguous as defined in the recommendation architecture. For example, in the recommendation architecture the activation generated in response to a specific perception of a car is made up of the activation of a population of information combinations recorded at the device level. These devices are located in a number of clusters. The presence of the active population is indicated to the competitive subsystem by the activity of a set of devices at the output level of the active clusters. Many of the active repetitions would be those recorded during past perceptions of cars, but in general none of the repetitions or clusters contributing to the active population will always be active for every perception of a car, and none of these repetitions or clusters will always be inactive in response to the perception of non-car objects. However, the outputs which communicate the activity of clusters are sufficiently rich in information that the total information available from all the active clusters is adequate for a functionally simple competitive system to determine a high integrity behaviour.

Solomon and Barsalou [33] contrast theories of knowledge in which the representation of properties is global with those in which it is local, and present experimental evidence in favour of local representations. In theories which adopt the global form assumption, a single global form represents a property that occurs in multiple contexts. In theories which adopt the local form assumption, properties are represented by different forms in different contexts. Solomon and Barsalou give the example of wings as a property of robins, butterflies, and jets. If wings is represented by a single global form, the same representation for wings will be activated when, for example, the questions are asked whether robins, butterflies or jets have wings. If wings is represented by a local form, different representations for wings will be activated for the three different types of object. In the recommendation architecture, the clusters which are activated in response to wings are those which contain information combinations recorded in the past when wings have been perceived. There will be a number of such clusters, and in each such cluster information is recorded on the basis of similarity to already recorded information. The situation with wings is very similar to the discussion of organizing experience of apples, blueberries, and plums in section 11. The number of clusters which could respond to wings may exceed the number of objects which contain wings. Although one or more of the clusters and many individual devices may respond to several different types of wing, the types of activation across all the clusters in response to robin-wing will be sufficiently different from the types of activation in response to other types that the combined cluster outputs contain enough information to discriminate between wings of the different types. The more similar the wings, the larger the probable overlap of device activation populations. Activations in response to wings of the same type will also differ, but the difference will be smaller than between wings of different types. Thus the "representations" of a feature in the recommendation architecture are more local than global in the sense defined by Solomon and Barsalou, but will have a "global" element depending on the degree of similarity between the different instances of the feature which have been detected and recorded in past experiences. Because of the ambiguity of information recording, some of this similarity could derive from factors such as seeing different features in similar locations or at similar times (for example, finding apples and plums on trees, but blueberries on bushes; or seeing robins and butterflies in gardens, but jets at airports), and such factors could increase the commonality in device activation populations.

The recommendation architecture is thus a perceptual theory in the sense defined by Barsalou, but differs from Barsalou's theory in that all information within cognitive processing is ambiguous with a rigorously managed partial context. "Representations" in the recommendation architecture are ambiguous, but tend to be local form.

24 Conclusions

Theoretical considerations demonstrate that any system which learns to perform a complex combination of functions is forced to adopt the form of the recommendation architecture. In that form, functionality is separated into a hierarchy of modules in such a way that all modules on one level are approximately equal and the information

exchange required to coordinate functionality is minimized to the degree possible consistent with approximate module equality. There is a further separation between the functional hierarchy within which a partially ambiguous context is maintained across an extremely complex pattern of information exchange, and a functionally simple system which uses the outputs of the functional hierarchy and the feedback of consequences to determine behaviour.

An electronic version of the key subsystems of this architecture has demonstrated that such a system can use inputs emulating visual and verbal inputs to generate appropriate behavioural responses. The system also demonstrates simple associative imagination: the generation of activations similar to those generated by visual objects often present in the past at the same time as the current visual object. The system confirms the functional value for minimization of information exchange between modules of a process resembling REM sleep in brains: a partial rerun of experience. The system demonstrates the capability to learn without disrupting prior learning.

The recommendation architecture has substantial potential both for understanding biological systems and for the design of systems to perform cognitive tasks.

25 Acknowledgements

The author appreciates the great efforts of the referees in assisting him to better express his ideas in the language of cognitive science. Some of the examples used to illustrate points and many of the cognitive models with which the recommendation architecture is compared were originally suggested by referees. The manuscript is much improved as a result of their efforts. The author of course remains fully responsible for the ideas in the paper.

References

[1] Anderson, J.A. and Sutton, J.P. (1997). If we compute faster, do we understand better? *Behavior Research Methods, Instruments & Computers Methods, Instruments & Computers* 29 (1), pages 67-77, 1997

[2] Anderson, J.R. and Lebiere, C. (1998). *The Atomic Components of Thought*. NJ: Erlbaum.

[3] Avrunin, G.S., Corbett, J.C., & Dillon, L.K. (1998). Analysing Partially-Implemented Real-Time Systems. *IEEE Transactions on Software Engineering*, 24, 8, 602-614.

[4] Barsalou, L.W. (1999). Perceptual Symbol Systems. *Behavioral and Brain Sciences*, 22, 577-660.

[5] Biederman, I. (1985). Human Image Understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing* 32, 29-73.

[6] Carpenter, G.A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, *IEEE Computer*, 3, pages 77-88, 1988.

[7] Churchland, P.S., and Sejnowski, T.J. (1992). *The Computational Brain*. Cambridge, Mass: MIT Press.

[8] Coward, L. A. (1990). *Pattern Thinking*, New York: Praeger.

[9] Coward, L. A. (1997). The Pattern Extraction Hierarchy Architecture: a Connectionist Alternative to the von Neumann Architecture, in Mira., J., Morenzo-Diaz, R., and Cabestanz, J. (eds.) *Biological and Artificial Computation: from Neuroscience to Technology*, 634-43, Berlin: Springer.

[10] Coward, L.A. (1999). The Recommendation Architecture: Relating Cognition to Physiology, in Riegler, A. and Peschl, M. (eds.) *Understanding Representation in the Cognitive Sciences*, 91-105, Plenum Press.

[11] Coward, L.A. (1999). A physiologically based approach to consciousness, *New Ideas in Psychology*, 17, 3, pages 271-290, 1999.

[12] Coward, L.A. (1999). A physiologically based theory of consciousness, in Jordan, S. (ed.), *Modeling Consciousness Across the Disciplines*, pages 113-178, Maryland: UPA.

[13] Coward, L.A. (2000). A Functional Architecture Approach to Neural Systems. *International Journal of Systems Research and Information Systems*, 9, pages 69-120, 2000.

[14] Coward, L.A. and Gedeon, T. (2000). Optimization of Architectural Parameters in a Simulated Recommendation Architecture, in press.

[15] Feigenbaum, E.A. and Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science* 8, 305-336.

[16] Fraley, C. and Raftery, A.E. (1998). How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. *Technical Report No. 329, Department of Statistics*, University of Washington.

[17] French, R.M. (1997). Pseudo-recurrent connectionist networks: An approach to the "sensitivity-stability" dilemma. *Connection Sciences* 9(4), 353-379.

[18] French, R.M. (1999). Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Science* 3(4), 128-135.

[19] Garlan, D., Allen, R. and Ockerbloom, J. (1995). Architectural Mismatch or Why it's hard to build systems out of existing parts. *IEEE Computer Society 17th International Conference on Software Engineering*. New York: ACM.

[20] Gedeon, T., Coward, L. A., and Bailing, Z. (1999). Results of Simulations of a System with the Recommendation Architecture, *Proceedings of the 6th International Conference on Neural Information Processing*, Volume I, pages 78-84.

[21] Grossberg, S. (1987). Competitive Learning from interaction to adaptive resonance. *Cognitive Science* 11, pages 23-63, 1987.

[22] Honkela, T, Pulkki, V. and Kohonen, T. (1995). Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map, in Fogelman-Soulie, F. and Gallinari, P. (eds.), *ICANN-95 Proceedings of International Conference on Artificial Neural Networks*, 2, pages 3-7. Paris: EC2 et Cie.

[23] Hopfield, J.J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences* 79, 2554-8.

[24] Johnson-Laird, P.N. (1983). *Mental Models.* Cambridge, Mass.: Harvard University Press.

[25] Kaufman, L. and Rousseeuw, P.J. (1990). *Finding Groups in Data*. New York: Wiley.

[26] Marr, D. (1991). *From the Retina to the Cortex: Selected Papers of David Marr*. Edited by L. M. Vaina. Boston: Birkhauser.

[27] McClelland, J., McNaughton, B., and O'Reilly, R. (1995). Why are there complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*. 102, 419 – 457.

[28] Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ.: Prentice-Hall.

[29] Robins, A.(1995). Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science* 7, 123 – 146.

[39] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning Representations by Back-propagating Errors. *Nature* 323, 533-6.

[31] Sabisch, T., Ferguson, A. and Bolouri, H. (1997). Automatic registration of complex images using a self organizing neural system. *IEEE International Joint Conference on Neural Networks*, Anchorage, Alaska.

[32] Sabisch, T., Ferguson, A. and Bolouri, H. (1998). Rotation, translation and scaling tolerant recognition of complex shapes using a hierarchical self organizing neural network. *Proceedings of International Conference on Neural Information Processing* 2, pages 1174-78. Berlin: Springer.

[33] Solomon, K.O. amd Barsalou, L.W. (2000). Representing Properties Locally. *Cognitive Psychology*, in press.

[34] Soni, D., Nord, R.L. and Hofmeister, C. (1995). Software Architecture in Industrial Applications. *Proceedings of the 17th International Conference in Software Engineering*, pages 196-207. New York: ACM.

[35] Sterelny, K. (1990). *The Representational Theory of Mind*. Oxford: Basil Blackwell.

[36] Stillings, N.A., Weisler, S.E., Chase, C.H., Feinstein, M.H., Garfield, J.L. and Rissland, E.L. (1995). *Cognitive Science: An Introduction.* Cambridge, Mass.: MIT Press.

[37] Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA.: MIT Press.

[38] Treisman, A.M. and Gelade, G. (1980). A Feature Integration Theory of Attention. *Cognitive Psychology* 12, 97-136.

[39] Willshaw, D., Hallam, J., Gingell. S., and Lau, Soo Leng (1997). Marr's Theory of the Neocortex as a Self-Organizing Neural Network. *Neural Computation* 9, pages 911-936, 1997.