Modeling Cognitive Processes with the Recommendation Architecture

L. Andrew Coward
School of Information Technology, Murdoch University,
Perth, Western Australia
landrewcoward@home.com

Abstract

The design of an electronic system with the recommendation functional architecture is described and the results of system learning with inputs simulating visual and verbal sensory inputs are presented. The differences from other cognitive architectural approaches are discussed, and it is emphasized that the recommendation architecture makes it possible for a system performing a complex combination of functions to be constructed, to recover from failures and damage, and to learn without affecting earlier learning. The properties of the implemented system supporting this conclusion are described. Given the understanding of the capabilities of the recommendation architecture derived from experience with the implemented system, complex cognitive processes can be realistically modeled.

Keywords: Cognitive Model, Connectionist Architecture

1 INTRODUCTION

Over the past 30 years there have been major advances in the technology for design of systems to perform very complex combinations of functionality. Design experience with such systems has emphasized that the needs to construct, repair and modify the functionality of such systems force the adoption of simple functional architectures. Coward [5] therefore proposed that biological brains have also been constrained to adopt simple functional architectures, and that because of the need for some biological brains to heuristically define their own functionality, the functional architecture adopted by such brains is the recommendation architecture. Coward [5] 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 [7] up to and including human consciousness [8], [9].

Simple simulations of systems with the recommendation architecture demonstrated that memory phenomena analogous with biological memory occurred in such systems [6]. A more detailed recommendation architecture which included management of the context for partially ambiguous information exchange was described in [11]. Simulations of portions of this architecture have been described [10], [11], [12].

The purpose of the current paper is to describe the implementation of a full version of the architecture described in [11], and demonstrate its application to a simple categorization problem and a more complex problem involving recognizing objects, recognizing and generating speech, and imagining objects. The architecture is then compared with other proposed cognitive architectures.

2 FUNCTIONAL COMPLEXITY AND FUNCTIONAL ARCHITECTURES

Some current commercial electronic systems have billions of hardware components and tens of millions of lines of software code [19]. Some of the most complex systems are real time systems which use input data to directly control a physical system, such as a telecommunications network or a chemical processing plant, with no human intervention and with severe time constraints within which it must respond [3]. 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 times 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 that an architecture which provided a multilevel descriptive hierarchy was required [19], and that in the absence of such an architecture it is extremely difficult to achieve integration of different functional modules [13]. A critical aspect of functional integration is provision of a context for information exchange between modules. In a commercial electronic system modules that exchange information 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 [13].

Coward [5], [6] pointed out that there are severe constraints on any system which performs complex combinations of functions. These constraints derive from the needs for processes by which such systems can be constructed, recover from failures and damage, and modify some functionality without affecting other functionality. All these processes require the existence of some means to relate functionality described at high level to functionality described at a detailed level. For example, any repair action requires that it be possible to derive from a description of a problem at high level where it is experienced (e.g. this system feature is not working) a description at a detailed level where it is possible to take corrective action (e.g. this device is not working). These requirements result in any such system being constrained into a simple functional architecture. In a functional architecture the functionality of the system is separated into modules, these components into smaller submodules and so on down to the smallest elements of functionality. This hierarchy of modules is the multilevel descriptive hierarchy as described by [19]. In a simple functional architecture all the modules on one level perform roughly equal proportions of system functions, and information exchange between modules, although essential for coordination of system functions, is minimized. These constraints permit the existence of usable logical paths from high level problem descriptions to detailed level descriptions [11]. It would be extremely difficult to construct, repair or modify a system which performed a complex combination of functions if the system did not have a simple functional architecture.

In commercial electronic systems the use of completely unambiguous contexts for information exchange between functional modules results in the memory/processing separation and instruction based operation of the von Neumann architecture [11]. If a system is to heuristically modify its own functionality (i.e. learn from experience), individual modules must determine the inputs they will receive from other modules. Under these conditions it is impractical to maintain an unambiguous context for all information exchange, but the requirement to maintain a partial, ambiguous context results in such a system being forced to adopt the recommendation architecture [11].

3 THE RECOMMENDATION ARCHITECTURE

A system with the recommendation architecture defines a portfolio of partially ambiguous information conditions out of its experience. The portfolio is defined in such a way that every experience is a repetition of one or more of these conditions. For any novel experience, condition definitions are dynamically extended so that the experience includes some of the extended conditions. Different combinations of repetition conditions are associated with different behaviours.

In the recommendation architecture there is a primary separation into clustering and competition. The clustering subsystem handles functional complexity and is therefore forced to adopt a simple functional architecture. The competitive subsystem is functionally simple and does not require such an architecture. The clustering subsystem selects and permanently records combinations of inputs from the environment and indicates by an output that they are occurring, and also indicates if a sufficiently similar repetition occurs in the future by generating a similar output. The clustering subsystem compresses a potentially huge input space into a much smaller output space. The competitive subsystem uses reinforcement learning to interpret the outputs of the clustering subsystem as alternative behavioral recommendations and selects one behavior.

The hierarchy of modules in the clustering subsystem is a hierarchy of repetition complexity. At the device level, combinations of device inputs are selected and detected if repeated in the future. Modules are made up of sets of devices, and a module condition repeats if a significant subset of its devices indicate a repetition. Supermodules are made up of modules and so on. The repetitions detected by modules on different levels can range in complexity from relatively simple direct combinations of system inputs at the device level to very complex combinations of such combinations occurring in a specified time sequence. To achieve this complexity, modules use the outputs of other modules as their inputs. Information combinations are recorded instantaneously and permanently at the device level. This recording algorithm is essential to maintain context for information exchange but is radically different from conventional neural network algorithms. The reason that outputs must be stable is that any 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. In other words, when a module begins to produce an output, any one of the potentially huge population of other modules might decide to use that output as an input. If the conditions under

which that output was produced were later changed, the meaning of that output would have changed, and it would in general be impractical to communicate the changed meaning of the output to all its recipients.

The high level recommendation architecture is illustrated in figure 1. Information combinations are heuristically selected by the clustering subsystem. The selection process is unguided 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. All information communicated (i.e. all outputs indicating the presence of repetitions on different levels) within the clustering subsystem or from the clustering subsystem to the competitive subsystem is therefore partially ambiguous: for example, no outputs at any level correspond consistently with patterns or categories in the input space which are clear conditions for system behavior. All outputs must therefore contain enough information to provide context for the recipient function to use the output. An output may be available to any of a large number of potential recipients, each of which makes its own decision whether to use the output. The requirement to maintain context means that once an information combination has been recorded and indicated by an output, any exact repetition of the combination must always result in an output which includes exactly the same output.

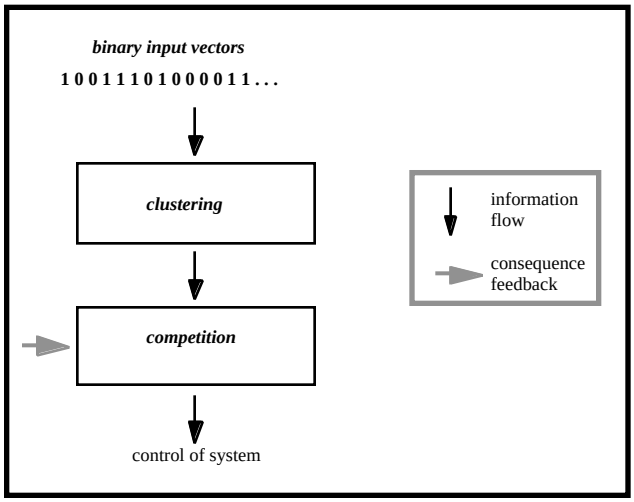

Figure 1. High level view of the recommendation architecture

There are four functions which an effective clustering subsystem must perform. The subsystem must select 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. It must accurately detect the presence of any repetition, and it must indicate the presence of a repetition with enough information richness to allow the competitive subsystem to generate a high integrity behaviour. Much of the structure of the clustering subsystem 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. The requirement for a simple functional architecture must also be met: modules on one level must detect roughly equal numbers of repetitions in ongoing experience; and information exchange between modules, although essential, must be minimized. In the recommendation architecture there is a primary separation into clustering and competition. The clustering subsystem handles functional complexity and is therefore forced to adopt a simple functional architecture. The competitive subsystem is functionally simple and does not require such an architecture. The clustering subsystem selects and permanently records combinations of inputs from the environment and indicates by an output that they are occurring, and also indicates if a sufficiently similar repetition occurs in the future by generating a similar output. The clustering subsystem compresses a potentially huge input space into a much smaller output space. The competitive subsystem uses reinforcement learning to interpret the outputs of the clustering subsystem as alternative behavioral recommendations and selects one behavior.

Use of consequence information is essential to enable the system to learn to perform appropriate functionality. However, consequence information cannot be used to modify the outputs generated by clustering, because this would not maintain information context. For example, suppose that a set of repetition conditions detected by clustering were interpreted by competition as a recommendation to perform a particular behaviour, but the consequences of the behaviour were negative. The modules in clustering which were active in generating the recommendation will also be active in generating different behaviours in different circumstances. Hence these modules cannot be modified to

give better performance for the current behaviour without introducing unknown changes to other behaviours. Therefore, as shown in figure 1, consequence feedback can only be applied to the interpretation of clustering outputs in the competition subsystem. The competitive function uses consequence feedback and reinforcement learning algorithms to converge on appropriate behavioral interpretations of clustering outputs.

4 MORE DETAILED VIEW OF THE RECOMMENDATION ARCHITECTURE

Inputs to the system as illustrated in figure 1 are vectors in which the elements indicate the presence or absence of some characteristic in an input space. These elements could, for example, be the presence or absence of pixels at different points in a visual field. As described below, in the implemented system these vectors are binary (i.e. either present or absent). Continuously variable elements are possible with greater architectural complexity.

4.1 THE CLUSTERING SUBSYSTEM

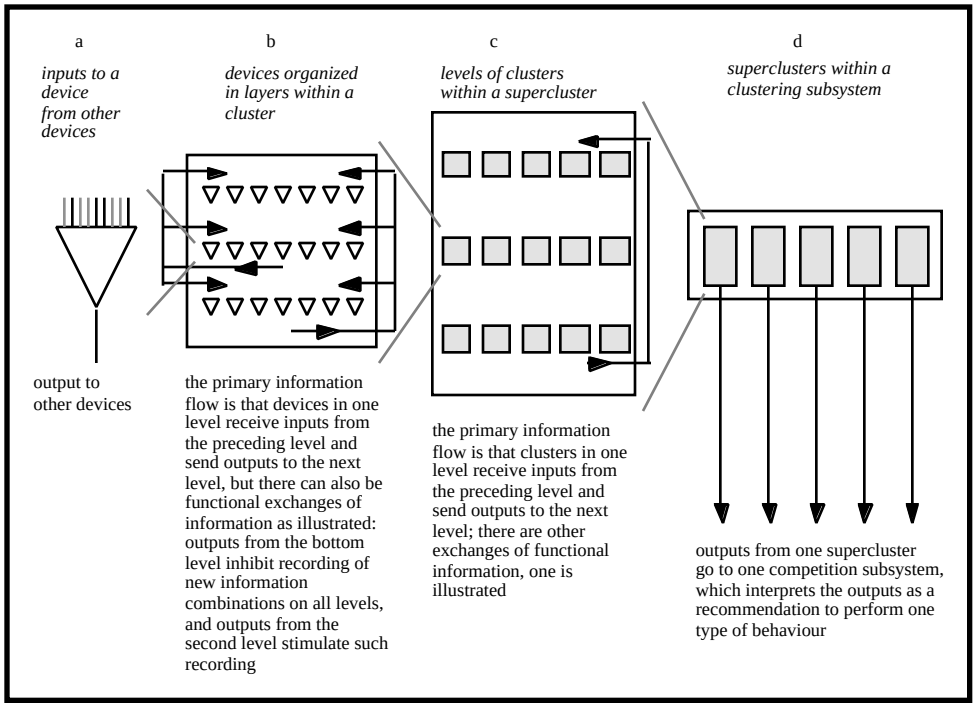

Figure 2. The architecture of a clustering subsystem at various levels of detail. A device is the basic element which records information combinations. The illustrated device had a number of provisional inputs selected by a random process. The black inputs were the inputs active when the device was triggered 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 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 2a. A device initially has randomly selected inputs, all with the same weight, but a threshold set so high that it will not respond to any input combination. Given a significant number of active inputs plus a functional signal supplied as described shortly, the device produces an output, and also disconnects all its inactive inputs and sets its threshold so that any repetition of a large subset of its current inputs will cause it to produce an output. The device has effectively recorded an

information combination. Devices which record multiple combinations are possible, but require somewhat more architectural complexity as described in [11].

The functional signal which triggers a device to record a combination can be understood from figure 2b, and from the more detailed view shown in figure 3. This figure illustrates a functional 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 a set of β layer devices which together indicate how much firing is present in the middle layer. These inputs stimulate any recipient devices which are able to record an information combination to do so. Devices also receive functional inputs from a set of γ layer devices which together indicate if any firing is present in the bottom layer and inhibit recording of new information combinations.

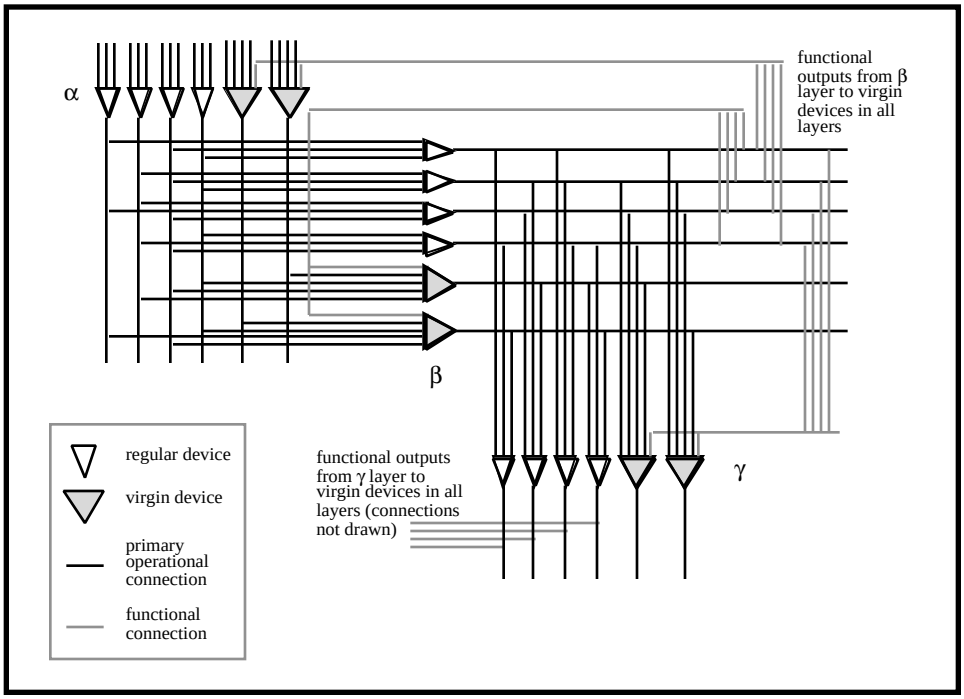

Figure 3. Connectivity within one cluster. Primary connectivity is excitatory. Functional connectivity from γ level to virgin devices inhibits virgin device imprinting, and is stronger that the functional connectivity from β level to virgin devices which stimulates such imprinting.

A layer of such clusters such as one of the layers shown in figure 2c can divide up experience into a set of heuristically defined repetition conditions. To understand this process, suppose that a system 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 to 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 are experienced.

The outputs from such clusters are the outputs of devices in the bottom layer, and the particular combination of outputs provides information about where in the spectrum of cluster repetition conditions the current input condition is located. Simulations have demonstrated that such a system can divide its experience up into a limited number of clusters, each indicating different (although sometimes partially overlapping) repetition conditions, and that although the outputs from one cluster do not correlate unambiguously with one object category, the outputs in combination contain information in a form which can readily identify the input category [11]. A separate competitive function using simple correct/incorrect feedback for early experiences will rapidly converge on high integrity identification of the correct object category. A complex input experience space may require subdivision of input clusters into repetition conditions which are different combinations of the first level clusters. This subdivision can be achieved by successive layers of clusters as illustrated in figure 2c. This hierarchy of clusters make up one supercluster. The clustering subsystem is separated into superclusters which independently select and indicate repetitions as illustrated in figure 2d. Outputs from different superclusters are directed to the competitive subsystems for different major types of behavior such as aggressive, fearful, food seeking, speech etc. The reason for separate superclusters is that the type of information combinations most useful for determining one type of behavior may be different from those useful for another type. Discrimination of this type could be provided by a genetically programmed selection of different input spaces for different superclusters. It can also be learned to some degree despite the restriction on changing outputs in response to consequences. If the same cluster output, when interpreted as an accepted behavior, sometimes results in good consequences, sometimes bad, the cluster is no producing a sufficiently rich output for functional purposes. If the cluster is then forced to accept additional inputs and record new combinations even when producing outputs, new outputs will provide additional discrimination. Previously recorded combinations are not affected, so the context for information is retained.

At the highest level, the clustering subsystem can thus be understood as arbitrarily dividing up all its experience into sets of repetition conditions. The presence of any of these repetition conditions is indicated by an output from the functional module detecting the condition. The competitive function then has the task of interpreting these combinations of repetition conditions into a system behavior. If the repetition conditions were generated by a supervised process, device level repetitions would be patterns, clusters would correspond with categories, superclusters with supercategories. In a clustering subsystem the correlation with such cognitive, functionally useful conditions is partially ambiguous.

4.2 THE COMPETITIVE SUBSYSTEM

The competitive system interprets the functionally ambiguous outputs of the clustering subsystem into a system behavior using a form of reinforcement learning. The algorithms for this reinforcement learning can be understood by consideration of figure 4. The devices, as illustrated in figure 4a, 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 pipes (figure 4c). 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 4b). 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.

In a complex system with multiple superclusters, a more complex competitive architecture is required. A set of pipes is needed to interpret the outputs of each supercluster into one of a number of different behaviors of the same general type, then a later set of pipes to select behavior between supercluster type. The different types of behavior corresponding with different superclusters and corresponding pipe sets include actions on the system itself as well as external behaviors. One important behaviors of this type is to keep the current set of active repetitions active for longer. This behavior corresponds with, for example, keeping a recently read telephone number active for long enough to dial it. Another important type is to activate a new set of repetitions which are not currently active but have often been active in the past when the current clustering subsystem outputs have been present.

Although the clustering subsystem cannot be directly affected by feedback of consequences, there is an indirect mechanism which is functionally important. If similar outputs from the clustering subsystem at different time are

interpreted as similar behaviors but result in opposite consequences, this condition is effectively an indication that the discrimination between repetition conditions is too coarse. A response which would not affect the context for existing outputs is to add input information to the relevant clusters and to generate additional outputs in experience conditions where there are already outputs. This mechanism would not change existing outputs but the additional outputs would allow the competitive subsystem to distinguish between conditions which were similar but different in a functionally important way.

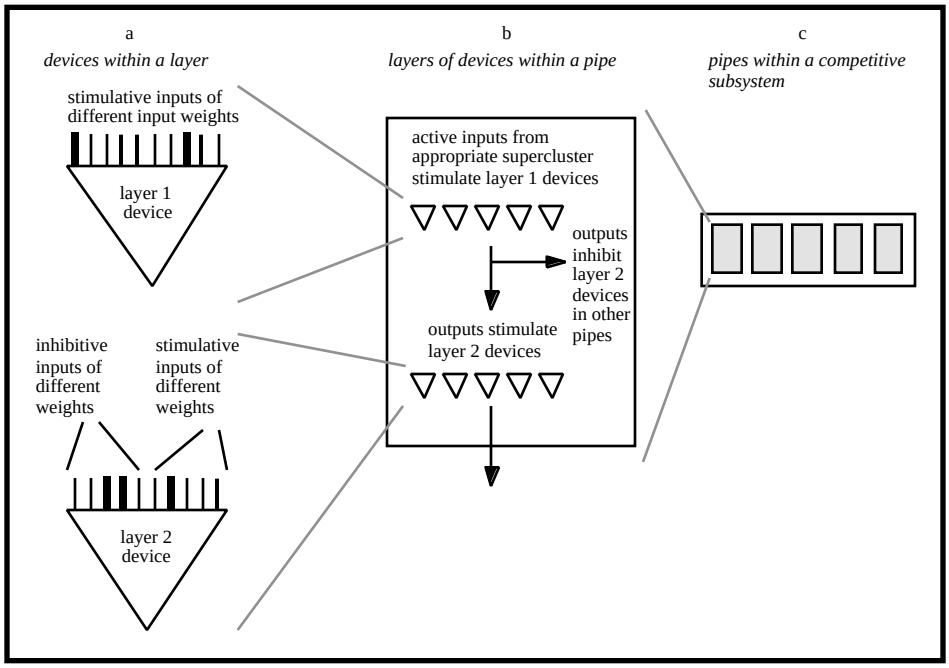

Figure 4. The competitive subsystem

Adding additional inputs increases the information distribution within the functional architecture. Such information distribution, although essential for the coordination of functionality, must be minimized. Minimization of information distribution also applies at the device level: the number of inputs must be minimized subject to the need to record functionally useful information combinations. Because modules at all levels heuristically determine information exchange, there must be a process to limit such exchanges to those most likely to be functionally relevant. The only accessible criterion to assess probable functional relevance is correlation of information in time: either simultaneous or with a constant separation interval. Hence a mechanism is required to limit provisional connectivity (at device, cluster, or higher levels) to inputs active when functionally relevant information has been active in the past. A rapid rerun of past experience would generate the environment in which such a mechanism could operate, and [5] therefore proposed that information distribution minimization is the primary function of dream sleep.

5 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.

5.1 SIMPLE CATEGORIZATION SCENARIO

To determine the ability of a recommendation architecture to heuristically organize its inputs into a hierarchy of repetition and use the outputs to determine behaviour, an artificial scenario was defined similar to those used in earlier simulations [10], [11], [12]. In this scenario, an input space of 1000 possible binary inputs or properties was used. Input conditions were 1000 element binary vectors, where a one-element indicated the presence of a property and a zero-element its absence. An input condition was 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 5. 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 6. 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 5) 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 5 also contain one of these properties.

The recommendation architecture system was presented with a sequence of different input conditions (or objects) 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.

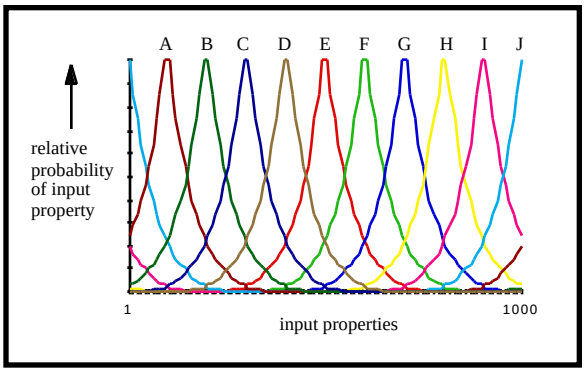

Figure 5. Probabilities of occurrence of properties in objects belonging to the ten categories

5.2 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.

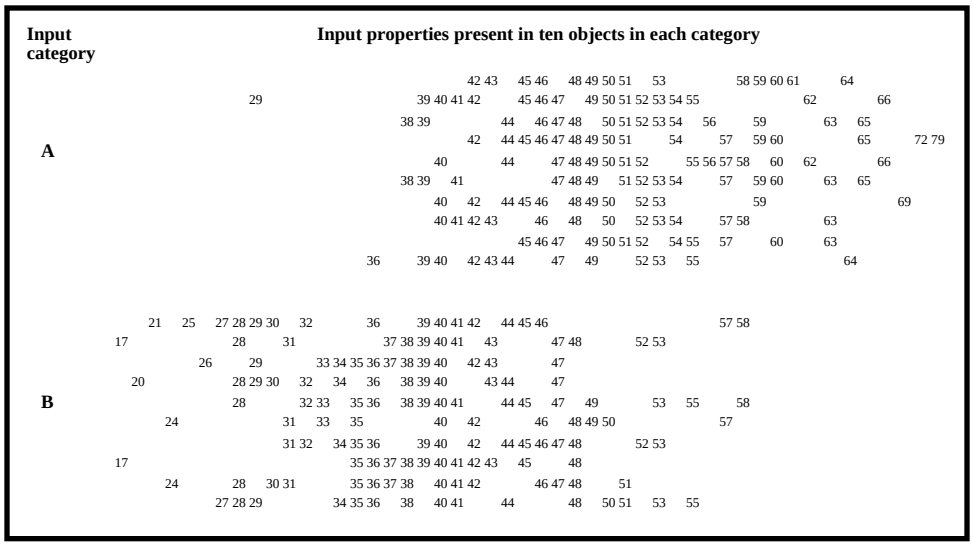

Figure 6. Examples of ten objects from each of two different categories constructed by random selection of properties from a 100 property input space. The numbers are the properties of the objects. The two categories had probability distributions of the form illustrated in figure 5. The actual objects used in the categorization experiments described in this paper were constructed in a 1000 property input space.

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 7. 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 8. 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.

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.

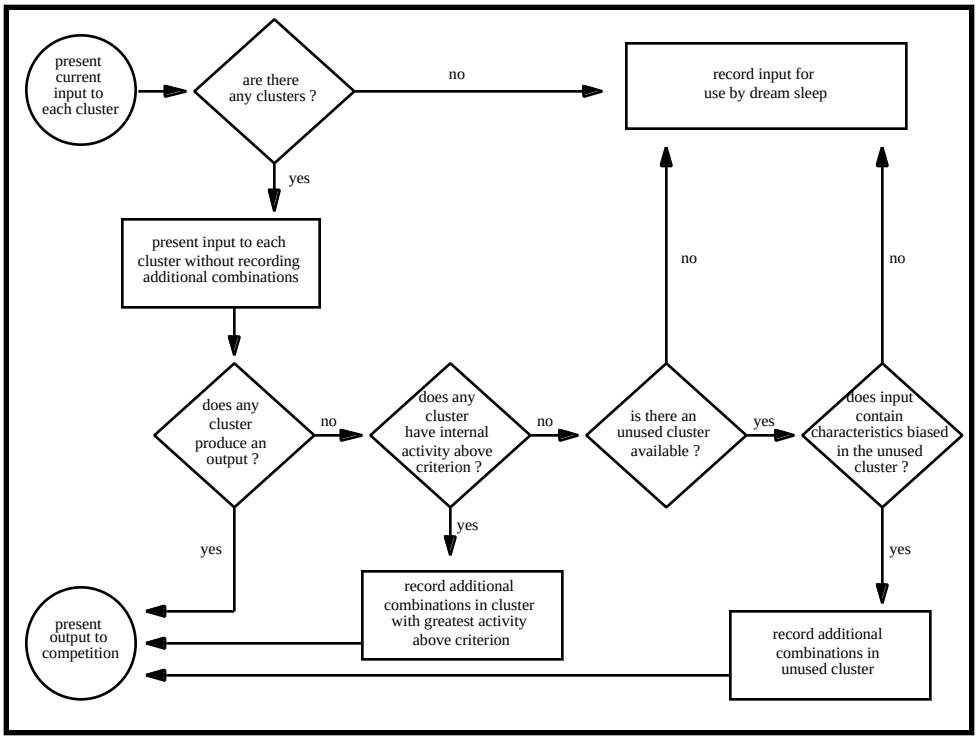

Figure 7. The process followed during the presentation of a single input condition to a simple (single layer) clustering subsystem.

This 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 [11]. It is called sleep with dreaming because it is an off-line process which uses a rerun of recent experience for functional purposes, as proposed for the function of sleep with dreaming in mammals by Coward [5]. In the simulations the rerun is implicit for implementation reasons.

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.

5.2 DETAILED ALGORITHMS

5.3.1 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 biological brain, the genetically specified parameters) the more effective and less complex the learning process.

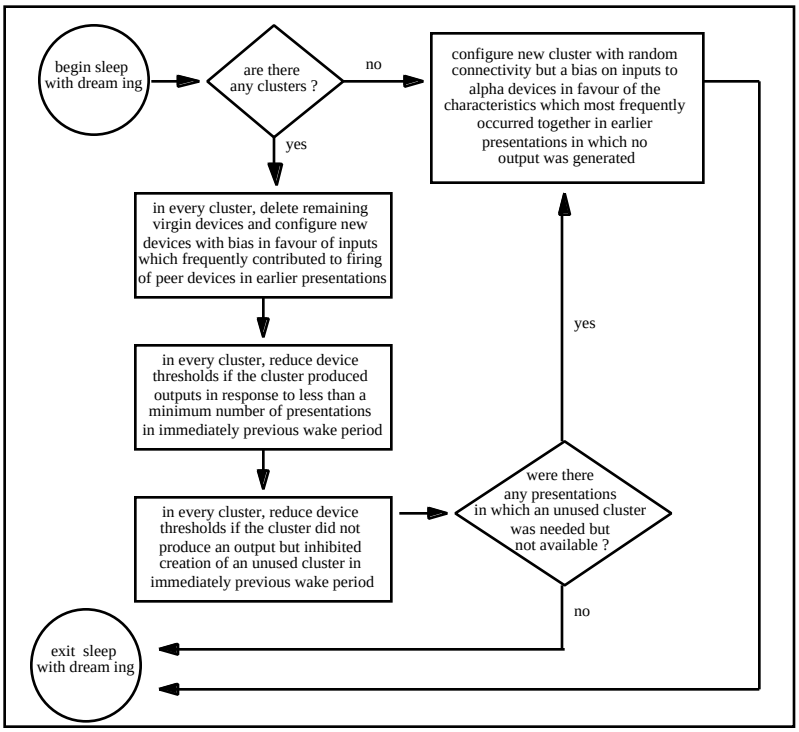

Figure 8. The process followed during sleep with dreaming in a simple (single layer) clustering subsystem.

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 in [10], [11] and [12], 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:

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

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.

Competitive subsystem

The implemented competition subsystem was made up of a number of parallel pipes, one for each possible behaviour, as illustrated in figure 4c. Each pipe contained two layers of devices as in figure 4b. 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.

There were two types of competitive subsystem runs performed on the outputs from each of the nine clustering run outputs. In both types, the first 4100 object presentations only developed a cluster structure: no outputs or consequence feedback were provided to the competitive subsystem and no action was determined. 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.

6 RESULTS OF SIMPLE CATEGORY RECOGNITION SIMULATIONS

6.1 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 9. 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 9 and 10 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 9. It can be seen from figure 9 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 10. It can be seen from figure 10 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 6. 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 numbers	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 10. For the presentations shown in figure 9, 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 [12].

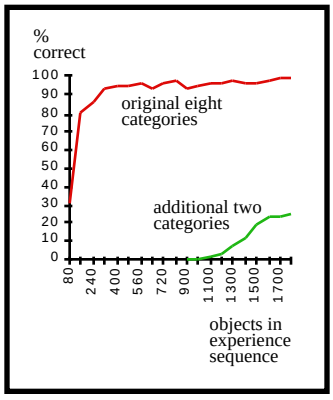

Figure 11. Recognition of categories already learned is only slightly affected when new categories are introduced.

6.2 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.

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. 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 when functionally duplicated neurons had been eliminated at an earlier point, early learning was somewhat slower, although the 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 11. 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.

7 APPLICATION OF THE APPROACH TO A MORE COMPLEX PROBLEM

The application of the architecture to a somewhat more complex problem can be understood by consideration of figure 12. The system is presented with a sequence of visual inputs, which are initially received by cluster set 1 in figure 12. These visual inputs model objects of different types using an input space of 1700 binary bits. 600 of these bits provide information on colour, the remaining 1100 on shape. There are six colour categories (red, green, brown, black, white, and clear) and eleven shape categories (cup, coffee, tree, grass, book, pen, paper, water, chair, apple, and car) which exist in the input space, the categories are modeled as probability distributions in the input space as in figure 5, and inputs corresponding with individual objects are constructed by random selection with the appropriate bias. A complete object input is constructed by merging a colour and a shape input. For example, a red cup is the merging of one red input with one cup input. In addition, there are four location categories (home, work, street and laboratory) which are modeled as probability distributions across the full 1700 bit space.

The system is also presented with a sequence of sound inputs, which are initially received by cluster set 3 in figure 12. These sound inputs model twenty-one different sounds (R, G, B, W, C, T, P, H, S, L, D, E, A, I, O, U, F, K, N, CH, and M) as probability distributions in a 2100 bit input space. Initially, clusters are created in cluster sets 1 and 3 in response to sequences of typical inputs are described earlier. The objects presented to cluster set 1 are shapes with a specific range of possible colours for each shape. Thus cups can be red, white or clear; coffee can be brown or black; trees can be green or brown; grass is always green; books can be red, brown or black; pens brown or black; paper always white; chairs green or brown; apples red or green; and cars can be red, black or green. As before, the system never sees two identical input vectors.

Clusters in cluster set 2 are then created by receiving a sequence of inputs each made up of a set of three outputs from cluster set 1. These three outputs are those generated by two objects plus one location. Clusters in clusters set 4 are similarly generated by presentation of a sequence of presentations of sets of three outputs from cluster set 3. The sets of inputs to cluster set 3 are limited to sound combinations corresponding with 28 words (red, green, brown, black, white, clear, cup, coffee, tree, grass, book, pen, paper, water, chair, apple, car, home, work, street, laboratory, eat, drink, walk, sit, drive, read, write). Some words may have only two sounds, in which case the third component in the input vector is empty. Different outputs from cluster set 4 are thus associated with the presence of different words.

Connectivity is then established between γ layer devices in all clusters in cluster set 4 and α layer devices in all clusters in cluster set 1. A series of presentations were made of a word to cluster sets 3 and 4, and simultaneously an object containing the concept referred to by the word to cluster set 1. For example, a word "red" presented at the same time as an object red cup. The input strengths of the connections to α layer devices from γ layer is set proportional to how frequently the corresponding α and γ devices were active at the same time in this word training period.

The two competitive subsystems were then trained. Competitive subsystem 1 was trained by reinforcing association between γ outputs from cluster set 1 and behaviours corresponding with speaking the 21 words (red, green, brown, black, white, clear, cup, coffee, tree, grass, book, pen, paper, water, chair, apple, car, home, work,

street, laboratory). Competitive subsystem 2 was trained by reinforcing association between γ outputs from cluster set 2 and behaviours corresponding with the eight actions (eat, drink, walk, sit, drive, read, write and think). These behaviours could also be interpreted as speaking the corresponding words.

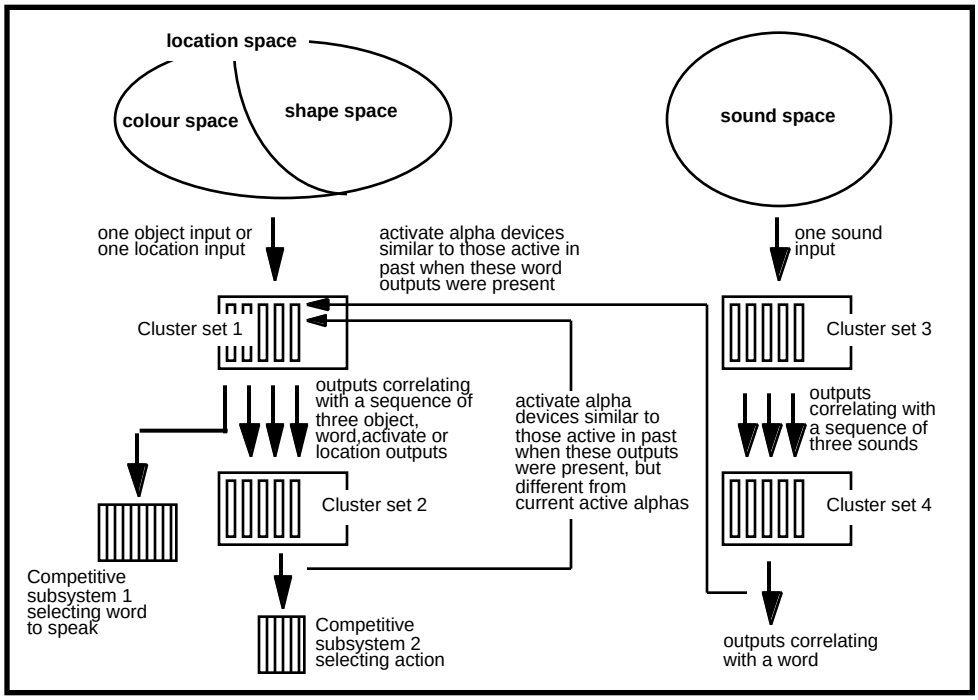

Figure 12. Architecture of a system which can take actions, learn to associate words with visual objects, and imagine objects which are not present. The system takes inputs from a colour input space, a shape input space, a location input space which is the combination of shape and colour spaces, and a sound space. The system organizes its experiences of inputs from these spaces into a hierarchy of repetition and interprets combinations of repetitions as system behaviours.

In the final training stage, connectivity is established between γ layer devices in all clusters in cluster set 2 and α layer devices in all clusters in cluster set 1. The connectivity is similar to that established from cluster set 4 to cluster set 1, but the weights are updated by every experience. These connections activate cluster set 1 any time there is neither a visual object nor a word presented to cluster set 1. However, the α devices activated are those which are not currently firing but have often fired in the past when similar outputs have been present. It can be interpreted as generating an activation corresponding with an object often present in the past at the same time as the object just presented.

The consequence feedback applied to competitive subsystem 2 rewarded the object combination, action combinations shown in table 1.

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

Table 1. Conditions under which positive consequence feedback was applied

8 RESULTS WITH THE MORE COMPLEX PROBLEM

Results with the implementation of the architecture illustrated in figure 12 demonstrate that it has the capability to learn to identify colour and shape conditions in its inputs from objects, and to generate the appropriate action in response to groups of objects.

In addition, the outputs generated by words from cluster set 4 can replace one of the object inputs to cluster set 1 and generate the appropriate action. For example, a red-cup object combined with "water" word and home location generated an output from cluster set 2 which was interpreted by the competitive subsystem selecting action as drink. In other words, the word outputs from cluster set 4 generated actions similar to those generated by the actual visual object.

Finally, if cup and home objects were presented without a third object, and the combination generated an activation in cluster set 1 via the connection path from cluster set 2 outputs to cluster set 1 alpha level, this activation generated outputs from cluster set 1 correlating with those generated directly with the presentation of object water. In other words, cup plus home reminded the system of water.

9 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.

The coordination of large numbers of interacting functions in the recommendation architecture depends upon providing and maintaining a context for partially ambiguous information exchange. Other investigations of concepts for more complex networks describe 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. [18]). The typical approach of targeting module outputs on specific vectors provides 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 [15] in which map outputs indicate individual words and relative position on the map indicates the syntax of the word. 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 [11] has demonstrated leads inevitably to some form of the memory, processing architecture once functionality becomes sufficiently complex.

The recommendation architecture has a hierarchy of functional signals which determine when learning will occur. The only comparable other example is in Adaptive Resonance [4] 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.

Other approaches do not discuss issues like rough operational equality of functional modules and minimization of information exchange between them, which are explicit in the recommendation architecture and essential for any system which manages a complex functionality.

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 [14].

In terms of capability, the recommendation architecture derives all its functional knowledge from its experience, unlike approaches like ACT [1] 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 [17]. Such problems are algorithmically complex but functionally simple in that system outputs do not dynamically change system inputs in real time [7]. 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 [16]. 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 [20]:

"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.

10 CONCLUSION

Electronic implementation of a system with the recommendation architecture demonstrates that such a system can heuristically organize noisy inputs from a very large input space into a hierarchy of repetition. A functionally simple competitive structure can use the indications of repetition to generate appropriate behaviour with high integrity. Such a system can learn how to behave appropriately, using only the inputs themselves and simple correct/incorrect feedback. Once the system has learned a behaviour, later learning of different behaviours has limited effect on existing learning.

A system with the recommendation architecture has been shown capable of simple verbal behaviour: generating words in response to visual inputs; and generating activations in response to words which are similar to the activations generated by the corresponding visual object. The system has also demonstrated simple associative imagination: generation of activations corresponding with visual objects which have often been present in the past at the same time as the current visual objects.

The construction from clustering and competition components of a system capable of simple verbal and imaginative behaviours indicates that the construction process could be extended to systems capable of more realistic cognitive processes. The recommendation architecture thus has substantial potential both for the design of systems to perform cognitive tasks and for understanding of biological systems.

REFERENCES

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

[2] Andersen, 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

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

[4] 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.

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

[6] 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.

[7] 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.

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

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

[10] 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.

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

[12] Coward, L.A. and Gedeon, T. (2000). Optimization of Architectural Parameters in a Simulated Recommendation Architecture. *Journal of Advanced Computational Intelligence*, in press.

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.