Optimization of Architectural Parameters in a Simulated Recommendation Architecture

L. Andrew Coward
Tamas Gedeon

Abstract

The recommendation architecture is a functional architecture with the capability to heuristically define a complex combination of functions which has been proposed as the architecture adopted by biological brains. This paper reports simulations of a system with that architecture which can heuristically define repetition conditions in a constantly varying input space which never repeats exactly the same state. The simulations demonstrate that the outputs from the system indicating the presence of the heuristically defined repetition conditions are a compression of the input space which carry enough information to allow rapid convergence on appropriate system behaviour when provided with simple correct/incorrect feedback. The simulated architecture has the capability for massive parallel operation on its inputs to derive system behaviour.

Introduction

The recommendation architecture has been proposed as the basis for understanding the functionality of biological brains (Coward 1990, 2000a) and as the only architecture which can support heuristic definition of a complex functionality (Coward 2000a). Simulations of the architecture have demonstrated the ability of a system to heuristically organize its experience into a hierarchy of repetition similarity, and to use the resultant hierarchy, with the assistance of simple correct/incorrect feedback, to rapidly converge on high integrity system behaviour (Coward 2000a). Further simulations have demonstrated that the outputs of a repetition hierarchy heuristically derived from a wide range of different input types contain enough information to guide appropriate system behaviour (Gedeon, Coward and Bailing, 1999). The objectives of the present paper are to extend the earlier simulations to address much larger input spaces than in previously reported work. Functional problems involving extremely large input spaces are expected to be the domain in which the recommendation architecture will demonstrate advantages over more conventional approaches (Coward 2000a). A key requirement is to demonstrate compression of input information into functionally useful outputs and to determine how to achieve optimal compression. A second objective is to demonstrate that categories of input conditions can be added in later experience without affecting responses to already learned categories, and that an effective repetition hierarchy can be generated when the order and relative frequency of input conditions from different categories is varied. A third objective is to demonstrate that the recommendation architecture is able to generate behaviours appropriate to sequences of objects, not just single objects as demonstrated in earlier work.

The recommendation architecture

Current systems design technology is based on the partitioning of functionality into functional components. The functionality of the system as a whole is separated into a set of such components, each component is subdivided into smaller components, the smaller components into yet smaller components, and so on until the smallest operations of the system (generally transistors and assembly code instructions) is reached. The resulting

hierarchy of functional components is known as the functional architecture. This functional architecture is vitally important because it makes it possible for functional components to be designed relatively independently, for multiple copies of the system to be constructed from summary information, for the system to be repaired, and for the functionality of the system to be changed without the introduction of undesirable functional side effects. To achieve these properties there are three major constraints on the form of the functional architecture: on each level of the hierarchy the number of system operations performed by each component must be roughly the same; the information exchanged between components, although vital for functional coordination, must be minimized as far as possible; and the information exchanged between components must be meaningful to the receiving component.

In current commercial electronic systems, all functional components have a completely unambiguous context for all information received from other components. This approach results in the familiar memory, processing separation and sequential execution of the von Neumann architecture. Heuristic modification of functionality in such systems (e.g. self modifying code) has proved impractical because of the difficulty of maintaining the unambiguous context. If ambiguity is allowed in information exchange, a partial context is still required This requirement for partial context results in the clustering and competition separation of the recommendation architecture. In the clustering separation, system experience is organized into a hierarchy of repetition similarity. The competitive separation uses the outputs of clustering to determine behaviour. The less constrictive requirement for a partial context allows heuristic definition of functionality, which means that the hierarchy of repetition similarity is heuristically organized. A system in which the hierarchy is defined by design is also possible, and heuristic organization can be guided in various ways by design, genetic or other a priori constraints.

In a system which heuristically defines its own functionality, the functionality of the system, including interaction between functions, is managed within the clustering separation. Within that separation, experience of an input space is heuristically organized into a hierarchy of repetition similarity which is useful for determining behaviour. The component hierarchy consists of repetitions, clusters, superclusters etc. A repetition is an information condition, a cluster is a set of repetition conditions, a supercluster is a set of cluster repetition conditions. If all information were unambiguous this would be a pattern, category, supercategory hierarchy, but the components in the clustering separation only correlate partially with such unambiguous conditions. The presence of a repetition condition on any level is indicated by outputs from the corresponding component. These outputs must have sufficient information richness to communicate a partial context to any component which receives them. A critical requirement in the clustering separation is a management process to select the information conditions which will be recorded, and therefore indicated if they ever repeat. This management process is described in detail in Coward 2000 and summarized in the next section. A key aspect of the process is that in order to maintain context for the information exchanged between components, once a component has generated an output in response to an information condition, it must always generate the same output in the future if the same condition recurs.

In a functional architecture, the information distributed between components must be minimized. If the components are heuristically defined, the information distribution between them must be heuristically minimized. This minimization implies that components only accept information from other components if that information is likely to be relevant. The only readily available indication of probable relevance is the frequency with which two components produce outputs which correlate in time. In a recommendation architecture an operation is therefore required in which the system is taken functionally off-line, and provisional connectivity is established between components which have frequently been active in the past at the same time or with a consistent time interval between their activity. A fast rerun of past experience is one way to determine past activity correlations. Coward (1990) proposed that a primary function of sleep in mammals is management of information distribution using a fast rerun type mechanism.

In the competition separation of the recommendation architecture, the outputs of a clustering separation are interpreted as a set of alternative behavioural recommendations and the alternatives compete with each other to generate a system behaviour. The consequences of the behaviour affect both the relative stimulative weights assigned to the repetition conditions interpreted as recommending the accepted behaviour, and the relative inhibitive weights of the alternative recommendations which tried to inhibit the accepted behaviour. The overall effect is to modulated the probability of a similar behaviour occurring under similar input information conditions in the future. To maintain the context for information generated by clustering components, the competition function does not affect the nature of the outputs indicating the detection of repetitions, only the functional interpretation of those outputs.

Viewed from a high level, the system requirement is to determine appropriate behaviour using information from some input space, and drive that behaviour in an implementation subsystem using outputs generated in some output space. Such outputs must be understandable by a relatively simple implementation subsystem. A clustering function takes information from an input space and generates outputs in an output space which indicate the repetition of information conditions in the input space. In general the output space must be functionally useful but smaller than the input space, i.e. there must be compression of information. In practical systems there could be a series of clustering functions, with competitive functions selecting subsets of the outputs of a clustering function to pass on to the next clustering function. The result would be compression of the information required to generate behaviour in a series of stages. This compression is the process which generates signals comprehensible to the competitive subsystem from the potentially huge input space. In principle the signal need not be compressed, if its information content were easily usable by the competitive subsystem, this condition could arise if there was substantial duplication of outputs which behaved in the same way.

The functional problem

The key questions about systems with the recommendation architecture is whether such systems can heuristically organize a functionally useful repetition hierarchy, and if such a

hierarchy can achieve enough compression of information to generate outputs which can be used by a relatively simple competition function to determine behaviour. A further question is the nature of the algorithms and parameters which define the starting point for and place constraints on the direction of the heuristic definition process. These algorithms and parameters will strongly influence the effectiveness of the repetition hierarchy for a specific functional domain, and will need to be defined in advance by a design process. The question of whether natural selection could lead to appropriate algorithms and parameters being coded in genetic information for biological systems of this kind is also of interest.

A further question is the nature of the a priori parameters which must be placed on the heuristic definition process for the process to be effective in a specific problem domain, and how such constraints can be defined in advance by a design process could plausibly derive from genetic information.

To address these questions, an artificial scenario was defined. This scenario had an input space of N possible binary inputs or properties. Input conditions were N element binary vectors. N was set at 100, 1000, or 10,000. 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. For example, as illustrated in figure 1 for N = 100, five different relative probability distributions define five input types A, B, C, D, and E. The three different combinations of characteristic probability distributions were those used in Gedeon, Coward and Zhang 1999. For example, in an input condition of type E in B0, each element of the vector is assigned a value of zero or one based on the curve labeled E in figure 1. In this case, only in the range 25 to 75 (i.e. 50% of the input space) will there be any ones, and the probability of ones is highest around input (or property) 50.

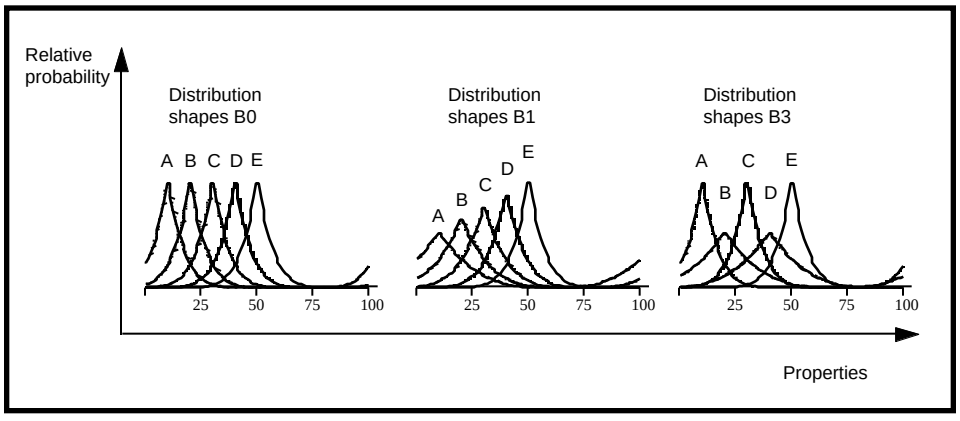

Figure 1. The definition of five input condition types A, B, C, D, and E by the relative probability of occurrence of the possible 100 binary properties in an input condition. The diagram shows the three different types of probability distribution mix used in Gedeon et alii 1999.

For the 1000 characteristic input space, the shape of the probability distribution was the same as for B0 in figure 1, but scaled to the 1000 input space. For example, input conditions had non-zero probabilities for about 50% of the input space. While the offset between the probability distributions in B0 was 10 properties (i.e. 10% of the input space) the offset between distributions in the 1000 property input space was varied between 50 and 125 (i.e. between 5% and 12.5% of the input space). The two extremes are illustrated in figure 2. For the 10,000 property input space, the shape of the probability distribution was again the same as for B0 in figure 1, but scaled to the 10,000 input space. The spacing between probability distributions was set at 0.5% of the input space.

Input conditions of a given type were created by randomly selecting properties from a set in which they occurred in proportion to the probability distribution for the type. For the 100 input space, 21 random selections were made to generate each input condition. Duplicates were discarded, resulting in an input condition having empirically as low as 9 properties, with an average of 14.6. For the 1000 input space, 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. For the 10,000 input space, 2100 random selections were made to generate each input condition. Duplicates were discarded, resulting in an input condition having between 1264 and 1357 properties, with an average of 1307. This process resulted in practice in all input conditions being different. This is appropriate to the objective, which 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.

Individual input conditions do not always appear obviously characteristic of their particular type, rather as a group their statistical properties will be characteristic of their type. For example, in the input conditions created within the 1000 input space with the minimum offset, about 80% of the conditions of type E included the most probable E property, but 50% included the most probable type D property, 25% the most probable type C property, 10% the most probable B property, and 0.4% the most probable type A property. In the input conditions created within the 1000 input space with the maximum offset, again about 80% of the conditions of type E included the most probable E property, and 20% included the most probable type D condition, although the most probable A, B and C conditions did not occur.

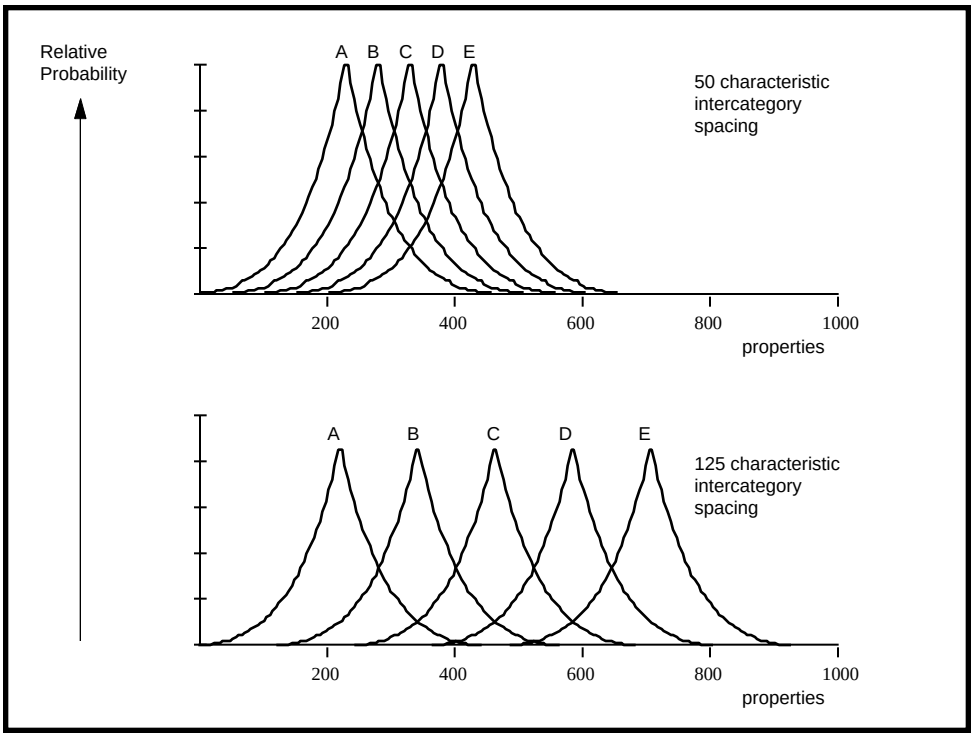

Figure 2. The definition of five input condition types A, B, C, D, and E by the relative probability of occurrence of the possible inputs in an input condition. Within one simulation, the relative probabilities for the different types were offset from each other by a constant spacing which varied from 50 to 125 properties. The two extremes are illustrated in the diagram.

The recommendation architecture system was presented with a sequence of different input conditions of different types. Periodically the sequence was interrupted by a period of sleep to manage information distribution. A typical experience sequence was 125 input conditions including 25 different conditions of each type A, B, C, D and E presented in the order A B C D E A B C D E etc. This experience was followed by a period of sleep, and then another sequence of 125 conditions. A full simulation run included 1000 different conditions of each type being presented to the system.

The clustering module organized the system experience of a sequence of 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 analyzed to determine whether they contained enough

information to be used by a competitive function with access to simple correct/incorrect feedback to generate behaviour appropriate to the different input conditions.

Pairs of outputs from the set of clusters generated by presentation of the individual input conditions then became the inputs to another clustering phase. These outputs corresponded with sequential presentation of pairs of input conditions such as AB, AC, CD etc. A typical experience sequence was 125 input conditions including 25 different conditions of types AB, AC, CD, BE and DE. This experience was followed by a period of sleep, and then another sequence of 125 conditions. A full simulation run included 400 conditions of each type being presented to the system.

Architecture of the simulated system

For the purpose of the simulation, an architecture was defined on several functional levels: device, layer of devices, cluster and layer of clusters as shown in figure 2. The inputs to the first layer of clusters were the input conditions, and the outputs are ambiguous repetition conditions of an information complexity roughly comparable with the complexity of input conditions. The inputs to the second layer of clusters were sequential pairs of outputs from the first layer, and the outputs are aimed to correlate ambiguously with pairs of input conditions. This achitecture is a simplified version of the Coward 2000b proposal for a human cognitive architecture. Clusters identify repetition similarity conditions, and interact to organize the sequence of system inputs (or experiences) into a limited set of such conditions. Each cluster has three layers of devices, α, β and γ. The α layer has two functions, it receives system inputs and also inhibits the creation of new clusters. The β layer detects the presence of an input similar to the heuristically defined similarity condition for the cluster and triggers modification of that condition to include the new input. The γ layer also has two functions. One is to provide a cluster output that indicates the presence of the similarity condition within the system input and also discriminate between different inputs meeting the cluster similarity condition, the other is to prevent any further modification to the cluster similarity condition once an output has been generated. In a functionally extremely complex system the layers with multiple functions would be split into one layer per function to allow independent functional optimization. Each layer in a cluster is made up of devices which can be imprinted with a currently present information combination to detect any future repetition of a large subset of that combination. This imprinting occurs under the control of the layer functions described above. The presence of such a subset is indicated by a binary output from the device.

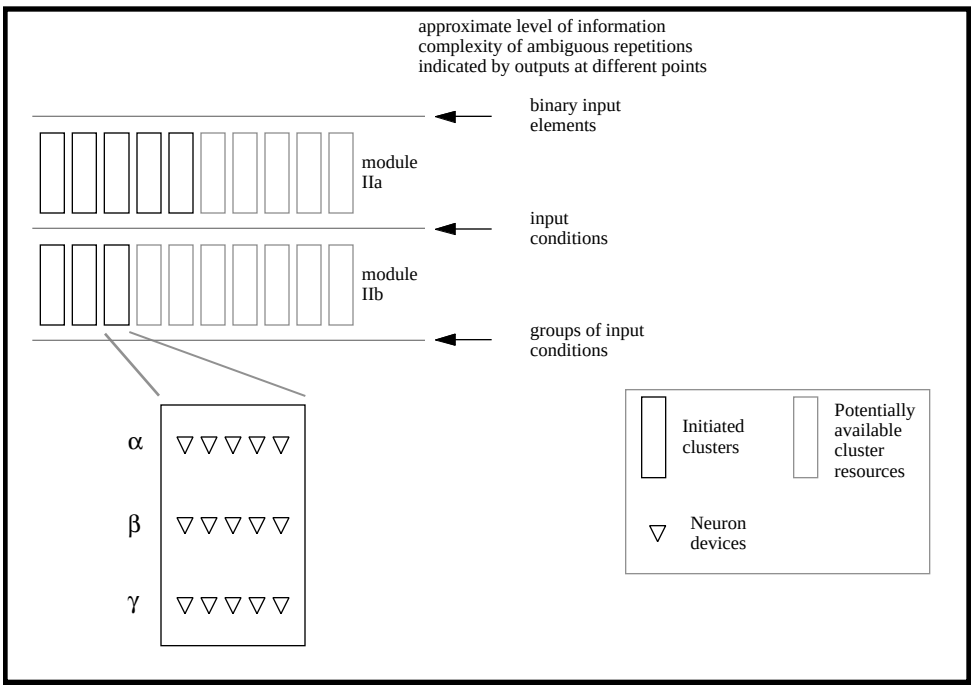

Figure 3. The recommendation architecture used in the simulations

The devices are initially provided with a randomly selected set of inputs from the preceding layer, or from the possible binary input vector elements in the case of the α layer. The random selection is biased in favor of certain inputs which play a significant functional role in other devices in the same layer as described below. This bias is an important element in the minimization of information exchange between functional components mentioned earlier. Devices when initially configured are called virgin devices, and will not produce an output even if all their randomly assigned inputs from the preceding layer are active. However, if in response to a system input a cluster has significant device firing in its β layer but no firing in its γ layer, the number of inputs required to fire virgin devices throughout that cluster is gradually lowered, resulting in virgin devices being imprinted until either the additional firing results in an output from the cluster or a minimum virgin device threshold is reached. For the first imprinting of a cluster the presence of enough significant α inputs as described below is sufficient to trigger device imprinting. Once a device is imprinted it becomes a regular device which will produce an output if a large proportion of the inputs with which it is imprinted are present, independent of the degree of firing in different cluster layers.

The number of γ outputs generated in response to a system input is an indication of the degree to which the cluster similarity condition is present in that system input. A system constructed with these functional components heuristically defines a set of repetition similarity conditions from a sequence of inputs such that eventually every input includes a repetition of one or more of the defined conditions. The presence of a repetition condition of a particular type is indicated by a gamma output from the corresponding cluster, while the identity of the actual γs producing the output provides some discrimination between different conditions of the same type.

Interpretation of outputs
Outputs from the clustering function are the indications of γ layer device firing in all active clusters. These outputs contain information of different types. One type is the simple presence or absence of outputs from each cluster. A second type is the strength of the output from each cluster (i.e. total number of devices generating the output). A third type is the particular combination of device identities generating the output. In general a competitive function will most easily distinguish between categories on the basis of type one information, less easily on the basis of type two, and least easily on the basis of type three.

Architectural parameters
A system is given an environment on which it must act, an input information space from that environment, and a sequence of experiences which are different combinations of active information from the information space. The requirement is to create a hierarchy of repetition similarity in which information conditions in the experience sequence at various levels of information complexity are heuristically selected, and outputs from the hierarchy indicate the presence of any selected repetition conditions. These outputs must be a compression of the input space, but have enough information to effectively manage system behaviour in the experienced environment.

In order to create the hierarchy of repetition, the system has information sources on which it can draw, and a major constraint on how it can use that information. The primary source of information is the actual experience sequence. A second source of information is a priori knowledge of the types of repetition hierarchies which have been effective for similar systems in similar environments in the past. This second type of knowledge will in general be expressed in the form of initial architectural parameters such as those illustrated in table 1. These parameters could be in the form of algorithms using variables derived from the actual experience sequence. Such knowledge can be defined from the experience of an external designer or from genetic programming based on natural selection.

The major constraint is the need to maintain the context of output information for any system component using the information. The requirement to retain context is severe, and as discussed in Coward (2000) means in general that the output generated in response to any selected information condition must be fully repeated if the information condition repeats. It could, however, be possible to generate a preliminary repetition hierarchy and use the form of that hierarchy to define or modify the initial architectural parameters. However, in order to ensure the context for the outputs from the final operational repetition hierarchy, the preliminary hierarchy must be fully discarded and a new hierarchy employing the new parameters created.

Some of the parameters which can be set on the basis of a priori knowledge are illustrated in table 1. It is important to note that a useful repetition hierarchy for a given information space could be derived using a very wide range of actual values for these parameters. However, the amount of experience required to generate a repetition hierarchy and the

functional effectiveness of the hierarchy are strongly affected by the values of parameters of this type.

To give an example, one parameter is the maximum number of γ neurons imprinted when an input condition does not result in any immediate γ output, but generates enough firing in the β level to trigger imprinting. In different simulations this number has been varied from 1 to 10. As discussed in detail later, a low setting of the maximum number results in greater compression of the output space relative to the input space, but the interpretation of the outputs may become more difficult, such as becoming dependent on the identity of individual γs rather than on the relative weight of cluster outputs.

The approximate number of clusters into which the system will tend to divide up its input experience is not directly specified, but is influenced by a number of parameters in table 1. A strong influence is exerted by the rate of reduction of neuron thresholds. When very few input conditions after the first generate cluster output, neuron thresholds are reduced within the cluster by a proportion until outputs in response to a significant proportion of input conditions results. The minimum number of α neurons to indicate cluster familiarity also influences the number of clusters. When a number of input conditions generate no output from any existing cluster, in general an additional cluster is configured. However, if the existing clusters have no output, but activity in the α layer above the level specified by this parameter, configuration of a new cluster is inhibited. This inhibition action also results in the reduction of the minimum number of β neurons firing to indicate similarity (table 1) in clusters generating the inhibition.

Initial configuration of a module	**Similarity criteria**	**Learning conditions**	**Resource addition to existing modules**	**Other**
Number of initial inputs to α, β, and γ neurons	Minimum number of γ neurons firing to indicate recognition	Limits on numbers of α, β, and γ neurons imprinted at initial learning	Number of virgin neurons added to α, β, and γ levels during sleep	Adjustment to regular neuron thresholds with changes in number of inputs
Statistical bias on selection of sensory inputs to α neurons	Minimum number of β neurons firing to indicate similarity	Limits on numbers of α, β, and γ neurons imprinted at regular learning	Number of inputs to virgin neurons added to α, β, and γ levels during sleep	Limits on frequency of creation of new modules
Numbers of α, β, and γ neurons	Criterion for selection of module for learning	Minimum number of inputs to a virgin neuron which must be firing to trigger imprinting under learning conditions	Statistical bias on selection of inputs to α, β, and γ virgin neurons added during sleep	Elimination of neurons which rarely or never fire after initial imprinting and neurons which fire at module output level too often (e.g. for every category object)
Allowed level of input duplication	Minimum number of α neurons to indicate cluster familiarity	Regular neuron threshold after imprinting	Number of inputs added to regular neurons during sleep	
	Minimum number of statistically favoured inputs present in an object to initiate new cluster	Changes to regular neuron thresholds when cluster is generally unresponsive, and timing of such changes	Statistical bias on selection of inputs to α, β, and γ regular neurons added during sleep	
	Correlation criterion to indicate cluster duplication		Allowed level of input duplication	

Table 1. Parameters which can be adjusted to modulate learning effectiveness

Architectural parameters could be specified exactly in advance, but this will limit the generality of the architecture. Alternatively they can be adjusted in response to indications of general learning effectiveness. In practice some combination will be appropriate for a real system. For example, it could be determined a priori that less than 10 clusters would be appropriate to organize any particular type of experience, and parameters could be adjusted if clusters were not forming or tending to form in too large numbers.

There are a number of requirements on these algorithms. The first requirement is that they cannot operate in a way that changes the context of system information. For example, the lowering of neuron thresholds resulting from rate of reduction of neuron thresholds will result in additional outputs in response to e repeated condition, but not the absence of any previously present outputs. Parameters can be influenced by any information which can be derived from system experience, but only in ways which do not affect context for existing outputs. The second requirement is that the algorithms must be adaptable enough to accommodate all possible system experiences. The purpose of the experiments with artificial data is to identify learning effectiveness indicators which could be used to adjust architectural parameters and algorithms for the architectural parameters which allow the system to adapt to different types of input conditions while maintaining information context.

The only changes to the architectural parameters made in going from the 100 property input space to the 1000 property input space were to increase the number of random selections of inputs to α neurons by a factor of 10 in both initial and dream sleep configuration processes. All other parameters remained the same. The change to the number of inputs is one which the system could make heuristically by detecting how many different properties were present in its input space.

Results of simulations

In the primary simulation runs the system was presented with 1000 different input conditions of each of five different types A, B, C, D, and E. Three simulation runs were performed with the probability distributions of each type offset by 50 properties, three runs with an offset of 75 properties, three runs with an offset of 100 properties, and three runs with an offset of 125 properties as illustrated in figure 2. The probability distribution had the same shape in each run, only the offsets were varied. In all of these runs the number of γ neurons which could be imprinted in response to one presentation (table 1) was limited to a maximum of four. In four additional runs, one for each offset, the number of γ neurons which could be imprinted was limited to a maximum of one.

Two additional runs were performed with variations in the way in which input conditions were presented. In one run conditions of the five types with 100 offset were presented in randomly selected order, and in addition the probability of selection was different for each type in the proportion

1.0 : 1.4 : 2.0 : 2.8 : 4.1

(i.e. roughly power law). In the other run, input conditions of the five types with 100 offset were presented until a response resulted from every condition. A series of presentations were then made in which type C was absent and a new type F offset further from E was added. Finally, a series of presentations with all six types were made.

Ordered presentations with maximum four γs imprinted in one presentation

	G1 50 offset			G2 75 offset			G4 100 offset			G3 125 offset		
	run 1	run 2	run 3	run 1	run 2	run 3	run 1	run 2	run 3	run 1	run 2	run 3
number of clusters	2	2	2	3	3	3	5	3	4	5	5	5
total number of output gammas	595	586	560	560	604	572	608	387	672	349	390	403

Ordered presentations with additional input condition type added later, maximum four γs

	H1 100 offset
	run 1
number of clusters	6
total number of output gammas	640

Random order presentations with different input condition type probability, maximum four γs

	H2 100 offset
	run 1
number of clusters	5
total number of output gammas	436

Ordered presentations with maximum one γ imprinted in one presentation

	M1 50 offset	M2 75 offset	M4 100 offset	M3 125 offset
	run 1	run 1	run 1	run 1
number of clusters	2	3	3	5
total number of output gammas	219	229	172	134

Table 2 The number of clusters and number of γ outputs for the systems at the end of the 1000 input space simulation runs. In simulation run H1 there were six input condition types, in all the other runs there were five types.

One run was performed in which input conditions were generated in a 10,000 input space with probability distributions offset by 500.

The major parameter of the systems at the end of the runs in the 1000 input space is given in table 2. One obvious conclusion is that the output space was much smaller when γ imprinting is limited to a maximum of one per presentation. The degree to which the outputs contain enough information to easily discriminate between the input condition types can be understood by consideration of table 3.

Input condition type	A	B	C	D	E
Numbers of gammas firing in each cluster in final 25 presentations of run 1 with 125 offset between input condition types	(0 0 0 68 0) (0 0 0 59 0) (0 0 0 42 0) (0 0 0 42 0) (0 0 0 31 0)	(28 0 0 0 0) (40 0 0 0 0) (31 0 0 0 0) (36 0 0 0 0) (28 0 0 0 0)	(0 0 0 0 26) (0 0 0 0 54) (0 0 0 0 108) (0 0 0 0 66) (0 0 0 0 63)	(0 48 0 0 0) (0 48 0 0 0) (0 30 0 0 0) (0 43 0 0 0) (0 49 0 0 0)	(0 0 25 0 0) (0 0 45 0 0) (0 0 28 0 0) (0 0 31 0 0) (0 0 22 0 0)
Numbers of gammas firing in each cluster in final 25 presentations of run 1 with 100 offset between input condition types	(4 0 0 125 0) (0 0 0 117 0) (0 0 0 100 0) (0 0 0 111 0) (0 0 0 96 0)	(64 0 0 27 0) (112 0 0 51 0) (74 0 0 7 0) (85 0 0 58 0) (79 0 0 44 0)	(15 0 0 0 59) (24 0 0 0 54) (24 0 0 0 80) (22 0 0 0 70) (15 0 0 0 77)	(0 86 0 0 0) (0 107 0 0 0) (0 91 0 0 0) (0 90 0 0 0) (0 97 0 0 0)	(0 0 70 0 0) (0 0 71 0 0) (0 0 66 0 0) (0 0 62 0 0) (0 0 66 0 0)
Numbers of gammas firing in each cluster in final 25 presentations of run 1 with 75 offset between input condition types	(10 118 0) (8 94 0) (7 76 0) (6 101 0) (7 97 0)	(83 67 0) (89 91 0) (49 62 0) (90 85 0) (85 74 0)	(216 10 0) (197 28 0) (208 31 0) (193 36 0) (207 25 0)	(169 0 40) (198 0 32) (152 0 32) (188 0 32) (177 0 52)	(83 0 87) (79 0 82) (76 0 87) (127 0 71) (78 0 80)
Numbers of gammas firing in each cluster in final 25 presentations of run 1 with 50 offset between input condition types	(223 57) (207 55) (200 37) (210 31) (160 24)	(184 202) (207 195) (184 194) (186 202) (196 186)	(148 271) (169 229) (156 280) (152 219) (154 294)	(143 265) (139 278) (126 257) (143 276) (141 259)	(127 252) (128 162) (129 188) (129 212) (133 213)

(x1 x2 x3, ...) actual number of γs producing output in clusters 1, 2, 3, ... during the presentation of the condition

Table 3 The numbers of γ neurons firing in response to the final 25 presentations. These simulations were with input conditions presented in an ordered sequence and with a maximum of four γs imprinted per presentation.

As can be seen from the table, in the simulations which allowed a maximum of four γs to be imprinted per presentation, with the largest separation between input condition types, a separate cluster was created corresponding with each type, and discrimination between types can be on the basis of different clusters producing outputs. A simple algorithm with access to correct/incorrect feedback can use this information to develop appropriate responses to the different conditions. With a smaller separation between types, the relative strength of cluster outputs contains enough information to discriminate between types. For example, as shown in table 4 for the 75 offset run, if ranges of output numbers are defined as weak, moderate and strong cluster outputs, input conditions can be distinguished on the basis of the difference between their cluster output strengths. Given somewhat more algorithmic complexity, correct/incorrect feedback can still be used to develop appropriate responses to the different conditions However, for the more similar types with 50 offset, input conditions C and D cannot be unambiguously distinguished on the basis of input strength alone. However, when the actual neurons which fired for the two different input types are examined, a basis for distinguishing can be found. For example, there were 33 γ neurons in cluster 1 which fired an average of 12 times in the last 25 C conditions but did not fire at all when D conditions were present. There were another 8 γ neurons which fired an average of 14 times when D conditions were present but not at all when C conditions were present. Other neurons also had firing frequencies which differed for C and D. Given rather more algorithmic complexity, even this individual neuron specific information can be used to develop appropriate responses (Coward 2000a).

	75 offset run			50 offset run	
Cluster / Type	1	2	3	1	2
A	w	m	-	s	w
B	m	m	-	s	s
C	s	w	-	m/s	vs
D	s	-	w	m	vs
E	m	-	m	w	m/s/vs

w	weak
m	moderate
s	strong
vs	very strong

Table 4 Distinction between input condition types based on the relative strength of outputs from different clusters

Similar output results for simulation runs in which only a maximum of one γ per presentation could be imprinted are shown in table 5. Cluster output information is adequate to distinguish between different input conditions, although there is more need to rely on individual neuron identity information than in the cases when the imprinting limit is higher. To illustrate the availability of this information, consider the results of measuring the actual number of times each individual γ neuron in cluster 1 produced an output in response to each type of input condition as illustrated in table 6. From table 5 for the smallest 50 property offset it can be seen that on simple numbers of γs firing in cluster 1 there is difficulty in distinguishing between input condition types C and D. However, from table 6 it is apparent that there are differences in the detailed pattern of firing for the two types. Similar differences in firing probabilities of individual neurons in response to different input types can be found between other input condition types. Hence there is a tradeoff between information compression between input and output and the complexity of the competitive algorithms needed to interpret the outputs. Simulations with input conditions with the offset reduced to 25 demonstrate that the outputs from the clusters generated with the same parameters as used for the other simulations have difficulty distinguishing between the input conditions. If there is inadequate information discrimination in the outputs, one option would be to use the indication that the outputs result in contradictory consequences to imprint additional device outputs with higher randomly assigned inputs. These additional outputs would not change the context for existing outputs. This option will be tested by future simulations.

Input condition type	A	B	C	D	E
Numbers of gammas firing in each cluster in final 25 presentations with 125 offset between input condition types, one gamma	(0 0 0 0 25) (0 0 0 0 17) (0 0 0 0 15) (0 0 0 0 16) (0 0 0 0 15)	(14 0 0 0 0) (26 0 0 0 0) (23 0 0 0 0) (25 0 0 0 0) (22 0 0 0 0)	(0 0 20 0 0) (0 0 17 0 0) (0 0 25 0 0) (0 0 12 0 0) (0 0 19 0 0)	(0 16 0 0 0) (0 17 0 0 0) (0 5 0 0 0) (0 15 0 0 0) (0 16 0 0 0)	(0 0 0 16 0) (0 0 0 22 0) (0 0 0 21 0) (0 0 0 18 0) (0 0 0 23 0)
Numbers of gammas firing in each cluster in final 25 presentations with 100 offset between input condition types, one gamma	(30 0 0) (28 0 0) (29 0 0) (29 0 0) (28 0 0)	(43 0 0) (52 0 0) (45 0 0) (48 0 0) (45 0 0)	(25 0 28) (25 0 11) (26 0 12) (23 0 12) (24 0 14)	(9 0 34) (9 0 33) (9 0 33) (9 0 29) (10 1 33)	(4 68 18) (5 73 18) (5 70 18) (4 68 18) (4 70 18)
Numbers of gammas firing in each cluster in final 25 presentations with 75 offset between input condition types, one gamma	(0 50 0) (0 46 0) (0 42 0) (0 46 0) (0 48 0)	(31 31 0) (28 39 0) (29 37 0) (27 39 0) (32 30 0)	(89 15 0) (81 20 0) (84 23 0) (53 21 0) (78 20 0)	(66 1 14) (68 4 15) (48 2 14) (71 2 13) (58 3 21)	(17 0 25) (22 0 29) (16 0 29) (24 0 26) (20 0 25)
Numbers of gammas firing in each cluster in final 25 presentations with 50 offset between input condition types, one gamma	(56 21) (55 17) (49 16) (53 17) (42 16)	(65 74) (75 72) (65 71) (65 73) (67 68)	(50 97) (64 89) (62 95) (52 76) (59 113)	(45 102) (50 100) (42 91) (44 113) (39 92)	(23 92) (40 53) (22 27) (24 72) (22 66)

(x1 x2 x3, ...)	actual number of γs producing output in clusters 1, 2, 3, ... during the presentation of the condition

Table 5 The numbers of γ neurons firing in response to the final 25 presentations. These simulations were with input conditions presented in an ordered sequence and with a maximum of one γ imprinted per presentation.

γ	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
Cluster 1 output activity frequency	
A	14 17 0 2 25 0 20 5 20 23 22 10 25 6 25 25 20 23 21 25 25 22 20 1 25 25 25 25 25 23 25 25 24 22 25 25 1 25 25 25 25 25 25 22 0 24 0 23 22 22 22 0 22 22 22 0 22 0 0 17 0 22 0 0 0 0 0 0 0 0 17 0 0 0 0 22 25 0 0 0
B	1 5 0 0 24 0 10 1 12 15 15 5 25 0 24 24 13 24 12 24 25 16 10 25 25 25 25 25 25 25 25 25 25 25 25 25 22 25 25 25 25 25 25 25 8 25 24 25 25 25 25 25 25 25 25 25 1 25 0 25 25 0 25 25 25 24 25 24 25 24 25 25 25 0 25 25 25 25 25 24
C	0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 16 0 0 25 14 16 14 10 17 16 17 25 25 25 25 25 22 25 25 25 25 19 25 25 18 25 25 25 25 25 25 25 25 25 25 17 25 2 25 25 13 25 25 25 25 25 25 25 25 25 25 25 6 25 25 24 25 25 25
D	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 22 9 7 21 25 4 25 7 25 25 0 25 25 3 21 25 25 25 25 25 25 25 25 25 12 25 1 25 25 22 25 25 25 25 25 25 25 25 25 25 25 25 17 25 25 8 25 25 25
E	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 7 0 5 4 0 4 4 0 0 8 7 4 7 5 7 10 7 4 0 7 0 23 7 3 22 23 23 22 25 25 25 22 22 24 23 25 4 25 23 0 25 25 25
Cluster 2 output activity frequency (neurons with strongly correlated firing omitted)	
C	0 0 0 0 1 3 0 0 3 3 3 1 1 11 4 1 6 3 3 1 4 0 11 9 13 8 2 8 0 17 11 0 13 19 14 9 12 7 25 25 8 1 15 25 23 1 9 8 25 10 16 25 7 25 0 8 8 19 6
D	1 0 1 0 15 17 4 6 17 16 19 18 9 25 22 11 23 18 21 7 25 6 25 25 25 25 10 25 4 21 25 16 8 4 2 25 1 0 3 0 25 0 0 14 0 17 25 25 19 25 0 9 0 20 0 25 25 5 16

Table 6 The numbers of times each individual γ fired in response to each type of input condition in the final 125 presentations with input types with 50 offset. These simulations

were with input conditions presented in an ordered sequence and with a maximum of one γ imprinted per presentation.

The results of the simulation which added an input condition type later are illustrated in table 7. The top set of gamma outputs are those generated just before condition F was introduced. At that point there was a some cluster output in response to every input condition, and the particular mix of clusters generating the output indicates the cateory of the input condition. The gamma outputs from each cluster for the final set of 25 presentations of one member of each of the six input condition types is also shown in figure 7. The presentations in which type C was not present and type F was introduced have resulted in generation of a new cluster 6 which generates outputs only in response to type F. An additional effect has been that cluster 4, which previously generated outputs only in response to type C input conditions now also produces outputs in response to types B and D, although the relative weight of cluster outputs still contains enough information to identify the type. This expansion of the repetition similarity range of cluster 4 resulted from a simulation condition that if a cluster only produces output in response to a small proportion of input conditions in a wake period, neuron thresholds are reduced until some outputs result in subsequent wake periods (table 1). This condition is intended to ensure that each cluster expands to cover roughly the same proportion of the input conditions. A simple modification to only apply the threshold reduction condition if the proportion has never been reached in the past, and making the proportion an average over several wake periods, would eliminate this problem.

Input condition type	**A**	**B**	**C**	**D**	**E**	**F**
Numbers of gammas firing in each cluster in final 25 presentations before condition F added. 100 offset between input condition types.	(0 0 37 0 0)	(77 0 21 0 0)	(29 33 0 101 0)	(0 87 0 0 0)	(0 0 0 0 66)	
	(0 0 48 0 0)	(79 0 24 0 0)	(31 42 0 54 0)	(0 86 0 0 0)	(0 0 0 0 63)	
	(0 0 42 0 0)	(84 0 18 0 0)	(35 11 0 4 0)	(0 66 0 0 0)	(0 0 0 0 37)	
	(0 0 46 0 0)	(96 0 24 0 0)	(39 21 0 132 0)	(0 94 0 0 0)	(0 0 0 0 45)	
	(0 0 48 0 0)	(79 0 24 0 0)	(25 41 0 95 0)	(0 80 0 0 0)	(0 0 0 0 82)	
Numbers of gammas firing in each cluster in final 25 presentations after condition F had been added and learned. 100 offset between input condition types.	(0 0 45 0 0 0)	(76 0 21 60 0 0)	(29 37 0 190 0 0)	(0 86 0 9 0 0)	(0 0 0 0 52 0)	(0 0 0 0 0 22)
	(0 0 47 0 0 0)	(76 0 24 22 0 0)	(42 38 0 194 0 0)	(0 89 0 19 0 0)	(0 0 0 0 52 0)	(0 0 0 0 0 15)
	(0 0 37 0 0 0)	(103 0 24 130 0 0)	(24 50 0 194 0 0)	(0 81 0 18 0 0)	(0 0 0 0 36 0)	(0 0 0 0 0 30)
	(0 0 40 0 0 0)	(87 0 24 55 0 0)	(37 40 0 194 0 0)	(0 88 0 14 0 0)	(0 0 0 0 69 0)	(0 0 0 0 0 20)
	(0 0 48 0 0 0)	(79 0 22 64 0 0)	(33 30 0 194 0 0)	(0 85 0 14 0 0)	(0 0 0 0 77 0)	(0 0 0 0 0 26)

(x1 x2 x3, ...)	actual number of γs producing output in clusters 1, 2, 3, ... during the presentation of the condition

Table 7 The numbers of γ neurons firing in response to the final 25 presentations in the simulation in which one extra condition was added after cluster outputs in response to five conditions had been developed. Maximum of four γs imprinted per presentation.

For the simulation in which the creation of clusters occurred with presentation of input condition types in random order, with different frequency for different types, the resulting cluster set could still be used to distinguish between input condition types on the basis of relative cluster output strengths. The simulation results under these conditions are not

distinguishably different from ordered presentation or condition types at the same frequency, as can be seen in table 8.

Input condition type	A	B	C	D	E
Numbers of gammas firing in each cluster in final 25 presentations with 100 offset between input condition types, four gammas, until learning was complete conditions were presented in random order with different relative probabilities of occurence	(0 0 0 88 0)	(0 31 0 33 0)	(0 0 0 0 36)	(98 0 0 0 0)	(66 0 64 0 0)
	(0 0 0 86 0)	(0 15 0 19 0)	(0 0 0 0 40)	(122 0 0 0 0)	(71 0 64 0 0)
	(0 0 0 88 0)	(0 11 0 55 0)	(0 0 0 0 21)	(92 0 0 0 0)	(67 0 53 0 0)
	(0 0 0 88 0)	(0 45 0 37 0)	(0 0 0 0 40)	(94 0 0 0 0)	(55 0 68 0 0)
	(0 0 0 88 0)	(0 16 0 19 0)	(0 0 0 0 27)	(100 0 0 0 0)	(61 0 60 0 0)

(x1 x2 x3, ...) actual number of γs producing output in clusters 1, 2, 3, ... during the presentation of the condition

Table 8 The numbers of γ neurons firing in response to the final 25 ordered presentations in the simulation in which the input conditions were presented randomly, with probability of occurrence different for different types. Maximum of four γs imprinted per presentation.

Run type \ Input condition type	AB	AC	CD	BE	DE
50 offset between input condition types ***(393 γs in 4 clusters)***	(147 0 27 0)	(155 0 55 0)	(91 20 116 96)	(33 0 111 92)	(2 0 102 39)
	(143 0 27 0)	(151 0 45 0)	(93 20 116 96)	(36 0 112 83)	(0 0 70 32)
	(147 0 27 0)	(133 0 54 0)	(72 20 107 58)	(33 0 111 67)	(0 0 101 48)
	(147 0 27 0)	(155 0 55 0)	(47 20 112 96)	(33 0 111 67)	(0 0 69 20)
	(147 0 27 0)	(155 0 55 0)	(86 20 116 96)	(33 0 104 62)	(0 0 69 32)
75 offset between input condition types ***(324 γs in 5 clusters)***	(70 0 0 0 0)	(27 80 0 0 0)	(0 0 60 28 0)	(0 0 0 68 0)	(0 0 0 15 12)
	(58 0 0 0 0)	(29 43 0 0 0)	(0 0 34 28 0)	(0 0 0 68 0)	(0 0 0 12 0)
	(46 0 0 0 0)	(19 0 0 0 0)	(0 0 0 28 0)	(0 0 0 60 0)	(0 0 0 12 4)
	(47 0 0 0 0)	(18 76 0 0 0)	(0 0 60 28 0)	(0 0 0 53 0)	(0 0 0 12 4)
	(60 0 0 0 0)	(20 80 0 0 0)	(0 0 60 28 0)	(0 0 0 65 0)	(0 0 0 12 0)
100 offset between input condition types ***(322 γs in 5 clusters)***	(0 93 0 0 0)	(0 0 40 0 0)	(0 0 0 0 12)	(68 0 0 0 0)	(0 0 0 4 0)
	(0 55 0 0 0)	(0 0 40 0 0)	(0 0 0 0 7)	(68 0 0 0 0)	(0 0 0 44 0)
	(0 52 0 0 0)	(0 0 40 0 0)	(0 0 0 0 0)	(62 0 0 0 0)	(0 0 0 7 0)
	(0 55 0 0 0)	(0 0 40 0 0)	(0 0 0 0 0)	(45 0 0 0 0)	(0 0 0 27 0)
	(0 94 0 0 0)	(0 0 40 0 0)	(0 0 0 0 14)	(68 0 0 0 0)	(0 0 0 44 0)
125 offset between input condition types ***(529 γs in 9 clusters)***	(0 56 0 0 0 0 0 0 0 0)	(0 0 0 0 45 44 0 0 12)	(0 0 0 0 0 0 0 0)	(114 0 0 0 0 0 0 0 0 0)	(0 0 0 0 0 0 0 4 0
	(0 38 0 0 0 0 0 0 0 0)	(0 0 0 0 0 0 0 0 0)	(0 0 0 0 0 0 0 0)	(114 0 0 0 0 0 0 0 0 0)	(0 0 0 0 0 0 0 12 0
	(0 26 0 0 0 0 0 0 0 0)	(0 0 0 0 0 0 0 0 0)	(0 0 0 60 0 0 44 0 0)	(104 0 0 0 0 0 0 0 0 0)	(0 0 100 0 0 0 0 0 12 0
	(0 56 0 0 0 0 0 0 0 0)	(0 0 0 0 37 44 0 0 12)	(0 0 0 15 0 0 44 0 0)	(11 0 0 0 0 0 0 0 0 0)	(0 0 89 0 0 0 0 0 12 0
	(0 56 0 0 0 0 0 0 0 0)	(0 0 0 0 0 0 0 0 12)	(0 0 0 0 0 41 0 0)	(114 0 0 0 0 0 0 0 0 0)	(0 0 100 0 0 0 0 0 12 0

Table 9. Number of γ neurons firing in the second layer of clusters in response to the final 25 ordered presentations of outputs from the first layer of clusters corresponding with the presentation of sequential pairs of system input conditions.

The outputs from the second layer of clusters is shown in table 9. These results demonstrate that the outputs from the second layer of clusters contain enough information to easily distinguish between different pairs of objects. Interestingly, for the more similar sets of input conditions it appears somwhat easier to distinguish between input pairs than between individual conditions.

The simulations with a 1000 input space in general resulte in an information compression between input and output of a factor of 2 when a four gamma maximum imprinting per presentation applied, and a factor of 7 when a one gamma maximum applied. This compares very favourably with the expansion of the output space by a factor of about 15 in simulations with a 100 input space, although large functional duplication of outputs meant that the outputs were practical for determining behaviour. When the input space is expanded to 10,000 a compression by a factor of several hundred is found. The achievable compression for analogous input types in a larger input space thus imcreases with the size of the input space.

Discussion and Conclusions

The simulations demonstrate that the set of simulation conditions for a recommendation architecture illustrated in table 1 can be given predefined algorithms which use input experience to heuristically generate sets of ambiguous repetition clusters in response to a wide range of input condition types. The size of the input space can be derived from experience and used to adjust the algorithms appropriately. The outputs from the cluster sets contain enough information to discriminate effectively between different types of input conditions, but are a compression of the input space. The degree of compression increases with the size of the input space. The outputs can be combined with simple correct/incorrect feedback to converge on high integrity behaviour.

A functionally useful cluster set can be generated independent of the order and relative probability of the different types of input condition, although these factors do affect the overall effectiveness of the final cluster set. A second stage clustering of the outputs of the first set of clusters can be used to generate outputs which can be used to generate high integrity behaviour in response to sequences of objects. The advantage of this second clustering stage over using a more complex competitive subsystem which used combinations of outputs from the first stage is that time delay between objects can be more easily managed.

The conclusion is that the architectural approach is able to heuristically organize a wide range of possible input types into repetition similarity clusters which generate outputs that contain enough information to control functionality. An architecture which can heuristically define a complex functionality is feasible. Although the simulation is implemented in a fashion which results in sequential determination of the state of each device, there is no functional reason why the states of all devices in a level could not be determined in parallel. The ambiguity of information thus allows massive parallelism provided the hardware will support the necessary operations.

References

Coward, L. A. (1990). Pattern Thinking, New York: Praeger.

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

Coward, L.A. (2000). A cognitive architecture based on the recommendation architecture, to be published.

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, pp 78-84.