Constraints on the Design Process for Systems with Human Level Intelligence

L. Andrew Coward

Abstract—**Any system which must learn to perform a large number of behavioral features with limited information handling resources will tend to be constrained within a set of architectural bounds. Unless design for a system with human like intelligence is performed within these bounds, the system will require excessive resources, and learning will introduce large numbers of undesirable side effects on prior learning. The design process must be focused on finding adequate compromises between the conflicting demands of resource economy and ease of learning. A number of detailed constraints on this design process are described.**

I. Introduction

A number of attempts have been made to place theoretical limits on information processing systems. Church [1] argued that there were families of problems that could not be decided by any algorithm. Turing [2] defined a conceptual machine and postulated that such a machine could perform any process that could be performed by a human being "working mechanically". Bremermann [3] argued that there was an upper limit to the information processing capacity of a given element of physical matter based on the number of energy levels within the mass of the element.

Practical considerations also generate constraints on the architecture of the most complex commercial electronic systems [4, 5, 6]. It has more recently been argued [7] that a range of practical considerations place theoretical bounds on the form of information processing systems which are much more severe and specific than constraints based on theories of computation. These practical considerations are (a) the need to perform a large number of behavioral features with relatively limited physical resources for information recording, information processing and internal information communication; (b) the need to add and modify features without side effects on other features; (c) the need to protect the many different meanings of information generated by one part of the system and utilized for different purposes by each of a number of other parts of the system; (b) the need to maintain the association between results obtained by different parts of the system from a set of system inputs arriving at the same time (i.e. maintain synchronicity); (e) the need to limit the volume of information required to specify the system construction process; (f) the need to limit the complexity of the system construction process; and (g) the need to recover from construction errors and subsequent physical failures or damage. As these needs become more severe, systems will be more and more tightly constrained within the theoretical bounds.

Two of the most significant of these practical considerations are the need to perform large numbers of behavioral features with limited information handling resources, and the need to change features without undesirable side effects on other features. These two considerations frequently conflict, and any system with a high ratio of features to available resources experiences very strong constraints on its architectural form, driven by the requirement to find an adequate compromise between resource requirements and modifiability.

One architectural constraint is that resources must be organized into a modular hierarchy defined in such a way that each module performs a group of similar system processes, and information exchange between modules is minimized as far as possible consistent with the need to limit use of resources. Such modules do not correspond with behavioral features, the relationship between modules and features can be very complex. This complexity results in the major differences between system architecture descriptions and user manual descriptions of the same system.

Other architectural constraints result from the need to maintain the often very complex behavioral meanings of information exchanges between modules when feature changes require changes to modules.

It is plausible that there will be natural selection advantages for brains which can perform more features with fewer resources, and [7] has therefore argued that the mammal brain is organized into a modular hierarchy of physiological resources in such a way that information exchange between modules is minimized as far as possible.

Information exchanges between modules in complex commercial electronic systems have unambiguous behavioral meanings. In other words, they can be interpreted as commands or instructions. This type of meaning is resource efficient but creates immense practical difficulties if features must be learned rather than defined under external intellectual control [7]. Learning is only practical if such information exchange meanings are partially ambiguous, in other words they can only be interpreted as recommendations. Support of partially ambiguous meanings is more resource intensive, but makes learning without severe interference with prior learning feasible.

If a system must support unambiguous information exchange meanings, it will be constrained into the familiar

L. Andrew Coward is with the Department of Computer Science at the Australian National University (phone: +61 0431 529 197; fax: +61 2 6125 0010; e-mail: andrew.coward@ anu.edu.au).

von Neumann form including the separation between memory and processing. If a system must support meaningful but partially ambiguous meanings, it will be constrained into a qualitatively different form which has some strong resemblances with the mammal brain [8].

Given a complex pattern of information exchange, it would be difficult to mix unambiguous and ambiguous information exchanges within the same architecture. The presence of ambiguous information would tend to make all information ambiguous. However, a system utilizing ambiguous information can be emulated on an underlying system using unambiguous information using pseudorandom processes at some points [7].

A system to perform a given set of features could be implemented in an immense number of different ways even within the same physical information handling technologies. If the ratio of features to total available resources is high, there are still many different ways in which the system could be implemented, but there will be some general architectural similarities between the different implementations reflecting the architectural constraint. If features are defined under external intellectual control, these similarities will include a modular hierarchy with minimized information exchange, and the von Neumann form to maintain contexts for unambiguous information exchanges. If features must be learned, the similarities will include a modular hierarchy with minimized information exchange, but qualitatively different constraints to maintain adequate contexts for partially ambiguous information exchanges.

The implication for the design of systems with human like learning and intelligence is that there may be many ways to build systems to implement particular aspects of human intelligence, but a system to implement a wide range of human capabilities will tend to be architecturally constrained. Attempts to design such systems are more likely to succeed if design proceeds within these architectural constraints.

II. The Impact of Resource Constraints on Architecture

If the information handling resources available to a system are limited, similar processes must be collected into groups which can be performed by a small set of resources customized to efficiently perform processes of the similar type. These small sets of resources are called modules. The definition of "similar" is only that all the processes can be performed efficiently on the same resources.

System processes can in general be regarded as condition detections [9]. Resource economy arises because a module detects a portfolio of similar conditions, and outputs from the module indicate detection of different subsets of its portfolio.

There could be a somewhat smaller degree of similarity between the processes performed by several different modules, in which case there could be some degree of resource sharing between those modules. A group of modules sharing a proportion of their resources forms an intermediate level module. Higher level modules will be defined by a yet lower (but still significant) degree of resource sharing between intermediate modules and so on.

The resultant modular hierarchy is thus driven by the need to economize on resources. Its form will be defined by the types of system processes which are "similar" given the available implementation technology.

Similar information processes from the point of view of resource sharing may be required by many different behavioral features, and any one such feature will require many different types of information process. As a result there will be little correspondence between behavioral features as defined by an outside observer and modules defined to minimize the requirement for physical information handling resources. This lack of correspondence is the reason for the very complex relationship between system architectures and user manuals in commercial electronic systems [8].

However, the lack of correspondence is driven only by resource constraints, and if the constraints can be relaxed in some cases then more correspondence may exist. For example, in the personal computer, there are completely separate software applications for user defined features like word processing, graphics processing, web access etc. One resource penalty of this approach is that if the same information is required by multiple applications, such as a diagram shared between the three applications, the information will in general be duplicated in different format for each application. All of a wide range of similar editing features must be separately implemented in each application.

In any system with human like intelligence it will be a requirement to select, record and organize a vast amount of information derived from experience. Resource constraints will in general mean that this requirement will be met in such a way that the same information store can be accessed in support of many different cognitive processes [10]. In the mammal brain it has therefore proved difficult to identify physical structures corresponding with cognitive features. However, for a critical process some specialized assignment of resources may be justified because improved learning and performance for a particular feature is worth the extra resources. A possible example is human face recognition, since physical damage to one part of the brain impairs face recognition but leaves object recognition intact [11] and different damage results in impaired object recognition but leaves face recognition intact [12]. However, the separation of resources is not complete, since certain specialist object recognition skills appear to use some face recognition resources [13].

The detailed organization of resources into modules is thus the best achievable compromise between some conflicting pressures including resource economy and feature performance.

III. The Impact of Modifiability

Organization of resources into a resource economy driven modular hierarchy presents a major problem for feature modification. The problem is that the implementation of a feature change will require changes to one or more modules, but such changes risk introducing undesirable side effects on other features that utilize the same modules. Furthermore, if a module receives information from a changed module, there is the risk of undesirable side effects on features utilizing that target module, and on features utilizing modules that receive information from that target module and so on. The result could be an exponentially increasing wave of side effects propagating out from the directly changed modules.

To reduce the degree of secondary side effects, modular hierarchies must be defined in such a way that the overall level of information exchange between modules is minimized as far as possible. However, such minimization tends to conflict with the need to economize on resources, because reducing exchanges will often mean that the same information must be independently generated within multiple modules. The modular hierarchy is therefore the best achievable compromise between resource limitation and minimized information exchange.

The process for design of complex commercial systems involves separation of system operations into groups, then rearranging the groups somewhat to minimize information exchange [7]. In the mammal cortex there is evidence of a similar pressure to minimize information exchange. Such information exchange will be carried by physical connectivity. In the cortex there is a hierarchy of physiological structures including columns and areas in which a primary means of identifying the structures is greater internal connectivity relative to external [14, 15, 16, 17]. [7] has argued that one role of REM sleep is to minimize the degree of information exchange increase between cortex modules participating in learning.

Minimization of information exchange reduces but does not eliminate the side effect problem. In the case of complex electronic systems, feature changes require a process that investigates the impact of possible changes on other features by design analysis and testing, and finds a set of module changes which implement the desired feature change with minimal side effects. This process requires strong intellectual control from outside the system itself using knowledge of the internal workings of the system, and is not feasible for a system which must learn such changes. The difficulty for a learning system is that such a system must typically wait until a feature is invoked to discover the effects of earlier changes in support of other features, while external intellectual control can consider the effects of changes on the full spectrum of features before the change is implemented.

A further factor which makes the learning of large numbers of behavioral features difficult within the architectures used for very complex electronic systems is the meaning of information exchanges between modules in such systems. Such exchanges can be viewed in two complementary ways [7]. One meaning is that an exchange indicates the detection of a condition within the information available to the source module. The other is that an exchange limits the range of currently appropriate system behaviors. Of the range of behaviors influenced by the target module, an exchange excludes a subset. Because many modules may be recipients of the same condition detection, that detection may have multiple different behavioral meanings in different targets.

In the case of commercial electronic systems, all behavioral meanings are unambiguous. In other words, there is 100% confidence in the behavioral guidance provided by a condition detection, and information exchanges can therefore be interpreted as commands or instructions. For example, a software instruction may have the format if: x = a [do: ….]. In other words, if a condition is detected (x = a) then the system is commanded to perform a process.

Unambiguous information exchange is resource efficient, but creates extreme difficulties for learning. Consider, for example, trial and error learning. Such learning requires an experimental change to behavior, modified by consequence feedback. An experimental change must be implemented by changes to one or more modules, which will in general result in changes to the conditions detected by those modules.

If a condition is changed, the module output indicating the detection of that condition will no longer be generated under exactly the same circumstances. However, that output will continue to be interpreted as a behavioral command appropriate in response to the original condition by any module targets. Secondary and tertiary etc. targets will also be affected. Consequence feedback only corrects for recently performed behaviors, and such consequence feedback could itself introduce additional changes which will be interpreted as changes to commands.

Trial and error learning on one behavior will therefore result in a large number of changes to unambiguous commands affecting other behaviors. Such changes can only be addressed the next time each behavior is invoked, and any correction will introduce yet more command side effects. The probability of such a process converging on a consistent set of high integrity behaviors is very low.

The alternative is that information exchanges could be meaningful but partially ambiguous. With this alternative, the behavioral interpretation of a condition detection is a recommendation. In general, many more conditions will need to be detected to create enough recommendation strengths to support high integrity behaviors, and this alternative will be much more resource intensive.

However, partially ambiguous information exchanges have a considerable advantage for learning, An accepted behavior B1 at some point in time will be supported by detection of many different conditions with recommendation weights (among others) in favor of B1. Slight changes to some of the conditions supporting B1 in order to implement learning of another behavior B2 may not be sufficient to

change the predominant recommendation weight in favor of B1 next time it is appropriate., and at that time consequence feedback can readjust recommendation weights. There is therefore the possibility of convergence on a consistent set of high integrity behaviors through trial and error learning. The critical problem is the definition of "slight".

Note that an alternative would be completely separate resources for each different behavior, or at least much less sharing. If available resources make this option feasible, then of course the architectural constraints do not apply.

IV. The Impact of Synchronicity

In general, a system will at each point in time need to select a subset of its current inputs, and detect conditions only within that subset. The detected conditions will indicate the appropriate behavior in response to the subset of inputs. Such subsets might correspond with different objects within the total perceived visual environment. This selection process is the system attention function.

The practical synchronicity issue is the need to maintain the association between results obtained by different modules from a set of system inputs arriving at the same time, i.e. from the one current attention object.

There are in principle two approaches to maintaining synchronicity: global and local [7]. In the global approach, the set of system inputs at one point in time are all recorded in a memory. Modules then take information on the synchronous set of inputs from the memory and detect conditions within that set, placing a record of their condition detections in the memory. Because some modules detect conditions which contain conditions detected by other modules, there is a specific sequence in which module condition detection must occur. This approach is thus essentially the von Neumann architecture, with a separate memory and sequential processing.

The alternative approach is local synchronicity management. In this approach, modules must be arranged in layers. A set of system inputs arrive at the first layer at the same time. The layer detects conditions and communicates condition detections to the next layer of modules. The next layer detects conditions that are combinations of the conditions detected by the first layer, and communicates those condition detections to the next layer and so on.

A disadvantage of the layered approach is that exact synchronicity would require every module in a layer to take exactly the same time to perform condition detection, and every communication between layers to also take exactly the same time. Any differences in processing time would introduce synchronization errors. If all information exchanges between modules were unambiguous commands, such synchronization errors could not be tolerated. Commercial electronic systems therefore universally employ the global synchronization approach with the associated von Neumann architecture.

However, if information exchanges are partially ambiguous, some degree of synchronicity errors can be tolerated. Hence a system utilizing such exchanges could avoid the high cost and complexity of a separate memory system and rely on a layered approach to synchronicity. In this approach, condition detection devices and modules will be arranged in a sequence of layers, with the majority of condition definition connectivity being from one layer to the next.

A further architectural constraint results from conflict between resource conservation and synchronicity. This conflict arises if it is necessary to detect conditions simultaneously within attention objects at different times, but resource constraints mean that all the conditions must be detected in the same modules. Such simultaneous detection would be required if it was sometimes necessary to detect conditions containing information derived for a number of different objects in a controlled fashion, in order to determine an appropriate behavior in response to the group of objects. The implication is that it must be possible for multiple populations of conditions to be detected and communicated within the same physical group of condition defining devices, without interaction between the information derived from the different objects, then to combine some of these conditions into more complex conditions.

As discussed in [18], one possible mechanism to support attention and separate populations of condition detections in the same physical group of devices resembles apparent cortex physiological activity. In this mechanism, the outputs from a device are sequences of identical voltage spikes. The rate at which spikes are generated indicates the proportion of its programmed conditions that the device is currently detecting. If a device integrates its spike inputs over an integration window, then placing a frequency modulation on spike times with the interval between peaks being larger than the integration window increases the response to modulated inputs relative to unmodulated inputs. Placing such a modulation on a subset of inputs (e.g. all the inputs from within one closed boundary in the visual field) means that only the subset will contribute to condition detections. Placing the same frequency but different phases of modulation on different objects means that independent populations of conditions will be detected in response to the different objects, provided that the phase difference is larger than the integration window. The number of independent populations that can be supported is roughly the ratio of the modulation interval to the integration window, which in the cortex corresponds with ~ four objects that can be retained in working memory [8, 18].

V. Architectural Implication of Change Management During Learning

The key problem in managing change is to find a compromise which allows change without requiring excessive resources. The key parameter for achieving this compromise is the degree of allowed change. If the allowed degree of change is too small, excessive resources will be

required because a module supporting one feature will not be able to change enough to be useful to any other features. If the degree is too large, changes to support one feature will result in unmanageable side effects on other features.

Degree of change can be understood in terms of the portfolio of information conditions detected by a module. If conditions within a module are changed, the circumstances in which some module outputs will be generated will also change. Each such output has many different behavioral meanings in the modules that are targeted by the output. The requirement is therefore to preserve all the meanings to an adequate degree.

If conditions within a portfolio can be changed by consequence feedback in response to individual behaviors, very large changes could occur which left little relationship between the conditions before and after the change. As the number of features supported by each module increases, it becomes increasingly probable that such changes will result in an unmanageable proliferation of side effects.

Hence as the ratio of features to resources increases, it will become less feasible to allow consequence feedback to act directly on modules. However, consequence feedback is needed to guide the recommendation weights in favor of different behaviors that are assigned to conditions. The result will be a separation between the modular hierarchy that defines and detects conditions (called *clustering* because it clusters similar conditions into modules) and a separate subsystem which uses consequence feedback to assign recommendation weights in favor of different behaviors. This subsystem is called *competition* because it manages the competition between alternative behavioral recommendations.

Because competition receives consequence feedback, information exchanges within competition cannot sustain behaviorally complex meanings. Competition will therefore be organized into components corresponding with different behaviors, and groups of components corresponding with different types of behavior. An input to a component from clustering can only be interpreted as a recommendation in favor of or against the behavior corresponding with the component, and an input from another component can only be interpreted as a recommendation against the behavior.

This separation between clustering and competition is a fundamental architectural separation which will appear in any system in which large numbers of behavioral features must be learned and the ratio of behavioral features to resources is high [7].

VI. Change Algorithms

If consequence feedback cannot affect the definitions of conditions, the question is how can module changes be managed. One extreme would be that portfolios be fixed a priori. In this case only limited learning is possible (i.e. only by changes to recommendation strengths).

A somewhat higher degree of change would be if portfolios could be expanded by adding similar conditions. In this case a module will always produce an output if it has produced an output in identical circumstances in the past, but will also produce the same output under similar but slightly different circumstances. It can be demonstrated that this degree of change permits continuous learning of significant numbers of different features with limited side effects on features learned earlier [19].

Gradually expanding portfolios require careful management of when and in which module such expansion can occur, and the circumstances in which a new portfolio can be initiated. This management is needed to ensure that resource requirements are not excessive and that an adequate degree of behavioral meaning is preserved. The only information available to guide this process is condition detection within the modular hierarchy, and the management requirement results in devices which can record gradually increasing portfolios being arranged in layers, columns and areas, with connectivity detecting the overall level of activity in specific groups of devices being used to determine whether portfolios will be expanded in other specific groups of devices [7, 8].

Higher degrees of change, for example with conditions that rarely occur being eliminated from a portfolio, could be possible depending on the compromise required between resources and preservation of earlier behavioral meanings, but in all cases the system must address how to manage change to achieve that compromise.

VII. Indirect Activation of Information

Modules in clustering generate outputs (indicating condition detections) to competition. Every such output is interpreted as a range of recommendations in favor of different behaviors, each recommendation having a different weight. Competition determines and implements the most strongly recommended behavior, and adjusts recently accepted weights using consequence feedback following the accepted behavior.

In the discussion so far, it has been implicitly assumed that all the conditions contributing recommendation strength are present within current system inputs. However, there are other conditions which under some circumstances may contribute possibly relevant recommendations.

For example, modules that are currently not detecting conditions, but that have often detected conditions at the same time in the past as modules that are currently detecting conditions, may have relevant recommendation strengths. Similarly, inactive modules that recorded conditions in the past at the same time as currently active modules, and inactive modules that have recently detected conditions, may have relevant strengths.

Indirect activation of such inactive modules may under some circumstances add important recommendations to the recommendations available from conditions actually present in current system inputs. However, to prevent the system being swamped by the activation of marginally relevant

information, indirect activations must be behavioral recommendations that compete with, for example, direct behaviors. Thus, a module output will have recommendation strengths in favor of direct behaviors, and also strengths in favor of indirect activation of other modules on the basis of past correlated activity.

Such indirect activations expand in a controlled fashion the volume of information available to influence behavior selection.

VIII. Design Process Systems with Human Like Intelligence

The vital need to find an adequate compromise between resource requirements and modifiability has a number of interrelated implications for any design process aimed at implementing a system with human like intelligence.

Firstly, the initial architectural design must focus on module separations on the basis of resource economy, not on the basis of different types of cognitive process. Secondly, a process must be designed to ensure the minimization of information exchange increases during learning. Thirdly, device learning algorithms must be specified so that the meanings of device outputs are adequately preserved during learning. Fourthly, architectural structures which manage the ongoing compromises between resource requirements and preservation of past learning must be defined.

Only when a system architecture has been established on this basis can the ways in which cognitive processes like memory can be supported by operations in and between modules be defined in detail. This definition will of course result in adjustments to the overall architecture.

This implementation approach has been successfully utilized for a number of moderately complex prototype learning systems. One system has demonstrated the capability to learn with manageable interference between earlier and later learning [19]. A second system [7] has demonstrated the ability to learn to associate appropriate behaviors with emulated visual inputs or sequences of such inputs, and to activate visual images in response to appropriate emulated verbal inputs. A third system [18] has demonstrated an attention and working memory capability.

A. Implementation Principles

As discussed earlier, it is possible to emulate a system using partially ambiguous information exchanges on a system which itself uses ambiguous exchanges. Some random element in the information processes must be present. Implementation can therefore be emulated on a regular computing system.

Within the clustering modular hierarchy at the most detailed level, there must be devices which can define and subsequently detect conditions within their inputs, in such a way that there is a strong tendency for any condition to be detected if it has been detected earlier.

Since there will in general be limited a priori information on the identity of behaviorally useful conditions, there will be a random element in the selection of such conditions. This random element (provided by some pseudorandom number generator) makes it possible to emulate a recommendation architecture system on a regular computer. Devices can be emulated by software rather than by physical creation of connectivity.

B. Architectural Design

There must be a primary separation between a modular hierarchy of condition definition and detection (clustering), and a component hierarchy of behaviors (competition). Complex behavioral management processes will all occur within the modular hierarchy.

The modules within the modular hierarchy must be defined on the basis of resource economy. There is a tendency in many design approaches to define modules that correspond with externally observable cognitive processes like “working memory”, “procedural memory”, “global workspace” etc. (see for example [20]). This approach is analogous with attempting to implement a commercial system with user manual definitions of modules. It is possible in principle, but resource requirements will in general be unmanageable.

The appropriate approach is to define different types of similar operations given the physical implementation technology, and to define modules on the basis of these operations. The most natural module definition, particularly if connectivity is a limited resource, is a collection of similar conditions. Module outputs will then indicate the detection of a subset of the conditions programmed on the module. Such outputs may provide some discrimination between different such subsets, but resource limitations will in general mean that such discrimination will not be provided between every programmed condition. The key requirement is that different combinations of module outputs be able to discriminate between circumstances with behaviorally different implications.

This discrimination capability can be managed. For example, if there are a number of occasions during which a specific group of modules generate outputs, these outputs are followed by a specific behavior, but the reward following that behavior is sometimes positive and sometimes negative, the implication is that the group of modules do not discriminate adequately between similar circumstances with behaviorally different implications. The requirement is to generate additional modules or module outputs in these circumstances. The new outputs will provide additional discrimination without interfering with the multiple other behaviors influenced to some degree by the modules in the group.

Substantial resources are required to manage when and where conditions will be added to a module. This management cannot simply be passive, but must actively determine the condition recording requirement for each module in response to every input state on the basis of the degree of condition detection in other modules.

Competition must be organized into a component hierarchy. Detailed components will correspond with individual system behaviors, higher level components will correspond with groups or sequences of such behaviors. Reward information available within competition means that information exchanges cannot have complex behavioral meanings. An input to a component from clustering can be a recommendation in favor or against the corresponding behavior; an input from another component can only be a recommendation against the recipient component behavior. In general the most appropriate current behavior will be determined by a process of competitive resolution determining the strongest recommendation weights. The first step is between components corresponding with general types of behavior, then between subcomponents of the component corresponding with the most strongly recommended general type to determine a more specific type within the general type, and so on.

C. Reward Systems

Reward systems must manage the adjustment of recommendation strengths in favor of recently accepted behaviors on the basis of rewards following those behaviors.

The conditions under which rewards will occur need to be specified a priori, but some evolution of those conditions could take place through experience.

A priori specification could include predefined conditions which if detected result in decreases (or increases) to recently accepted recommendation strengths. Such conditions can be regarded as indicating pain (or pleasure).

Another a priori specification could be to reward recently accepted recommendation strengths if the conditions detected after the behavior with attention focused on self were similar to conditions detected before the behavior in an external object (i.e. rewarding imitation). Repeating recently heard sounds would be encouraged by this capability and could improve the effectiveness of early learning [8].

D. Learning Process

The process of learning a behavior can be viewed as dividing up the space of input states into relatively independent similarity components corresponding with modules detecting different groups of conditions. These components are in some ways analogous with independent components [21]. The differences are that they can be continuously evolved by a tightly managed process, and their degree of statistical independence is not explicitly managed.

Components are defined corresponding with “atomic” behaviors, and corresponding with random (or biased random) sequences of such behaviors. Random (or biased random) connectivity weights are established between modules and components corresponding with behaviors, and connectivity with negative weights between components.

Thus in response to an input state, initial behavior will be relatively random (depending on initial connectivity bias). Reward (positive or negative) adjusts input weights into the component corresponding with the recently selected behavior. Consistent negative rewards to a behavioral sequence component result in reprogramming for a different sequence.

The combination of initial connectivity bias and use of imitation to generate rewards can result in bootstrapping of complex cognitive behaviors over reasonable periods of experience [8].

E. Device Algorithms

Within competition, device learning algorithms will involve weight adjustments to individual inputs on the basis of reward signals, resembling typical artificial neural network algorithms. However, the condition recording devices within clustering require algorithms which cause the device to respond to a gradually increasing volume of their available input similarity space. Conditions will in general be added to a device but not removed, although a viable algorithm could allow removal of a condition if it did not occur again within some period of time [8]. Various algorithms of this type are possible [7, 8].

Because connectivity resources are not unlimited, provisional conditions will need to be defined within modules in advance of experience. New conditions will the active subset of a provisional condition within a module when condition recording is required in that module.

Furthermore, device algorithms must support the ability to focus processing on subsets of system inputs (i.e. an attention capability) and to process independent populations of condition detections within the same physical resources (i.e. working memory). Algorithms with this capability have been described in [8, 18].

F. Detailed Physical Arrangements

Condition recording devices must be arranged in layers to support synchronicity management, with a modular hierarchy of columns, arrays and areas overlaid on the layers. Connectivity between modules must be defined to manage both detection of conditions and decisions on when and where new conditions will be recorded. There are many different ways in which these arrangements and connectivity could be implemented, depending on the tradeoffs made for the specific system [8].

G. Initial Configuration

A priori information on the identity of useful conditions is limited. However, there are some sources of such information. One source could be design (or for biological systems genetic) information that certain types of information were useful for discriminating between certain types of behaviors.

Such information can be utilized by biasing the initial connectivity within clustering and between clustering and competition. Thus a priori knowledge of the complexity of

conditions most appropriate for discriminating between certain types of behaviorally different circumstances can be used to bias the number of inputs to provisional conditions to some modules in clustering, and to bias the creation of connectivity from those modules in favor of components corresponding with the behaviors.

H. Minimization of Information Exchange

Provisional conditions for a module are defined as random combinations of inputs from other modules. However, completely random selection would result in a high level of information exchange between modules. There is some additional information which can be used to reduce this degree of information exchange. This additional information is past experience. With the assumption that past experience, and especially recent experience, is some guide to the probable form of experience in the immediate future, past experience can be used to bias the definition of provisional conditions. If two modules have often detected conditions at the same time in the past, a provisional condition in one module incorporating outputs from the other module is somewhat more likely to be useful than if the two modules have never previously been active at the same time. Biasing provisional conditions in favor of inputs which have often been active in the past at the same time as their targets will therefore improve the probability of a useful regular condition. The bias configures resources in such a way that they are more likely to be useful for recording information during future experiences, it has no effect on past memories. [22] has argued that sleep, including REM sleep, plays this role in the mammal brain. Simulations indicate that such a process reduces the connectivity resources required to support learning by about 20% [23].

IX. CONCLUSION

Design of systems to implement human like intelligence will require the capability to learn large numbers of behavioral features with relatively limited information handling resources. The need to be able to learn without excessive loss of earlier learning forces both architectural compromises and a specific type of approach to design. The feasibility of designing systems within these constraints has been established, and a range of detailed practical constraints on the design process established.

REFERENCES

[1] Church, A. "An Unsolvable Problem of Elementary Number Theory", American Journal of Mathematics 58, 1936, pp. 345-363.

[2] Turing, A.M. "On computable numbers, with an application to the Entscheidungsproblem", Proceedings of the London Mathematical Society Series 2(42), 1936, 230-265

[3] Bremermann, H. J. "Optimization through Evolution and Recombination". In M. C. Yovits, G. T. Jacobi and G. D. Goldstein (eds), Self-Organizing Systems – 1962, 93 – 106.

[4] R. Kamel, "Effect of modularity on system evolution", IEEE Software, January 1987, pp. 48-54.

[5] D. Garlan, R. Allen and J. Ockerbloom, "Architectural mismatch or why it'e hard to build systems out of existing parts", IEEE Computer Society 17th International Conference on Software Engineering, 1995. New York: ACM.

[6] G. Chastek and L. Brownsword, "A case study in structural modeling", Carnegie Mellon University Technical Report CMU/SEI-96-TR-035, 1996. Available: http://www.sei.cmu.edu/pub/documents/96.reports/pdf/tr035.96.pdf

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

[8] L. A. Coward, A System Architecture Approach to the Brain: from Neurons to Consciousness". New York: Nova Science Publishers, 2005.

[9] L. A. Coward, "The Recommendation Architecture Model for Human Cognition", *Brain Inspired Cognitive Systems 2004*, L. S. Smith, A. Hussain and I. Aleksander, (editors), University of Stirling: Stirling.

[10] L. A. Coward and T. D. Gedeon A model for representation of concepts in the brain. Proceedings of the 2005 International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning. Espoo, Finland.

[11] F. Neuner and S. R. Schweinberger, "Neuropsychological impairments in the recognition of faces, voices and personal names. Brain and Cognition, 44 (2000), pp. 342-366.

[12] M. Farah, "Patterns of co-occurrences among associative agnosias: Implications for visual object representation. Cog. Neuropsych., 8, 1991, pp. 1-19.

[13] I. Gauthier, P. Skudlarski, J. C. Gore and A. W. Anderson, "Expertise for cars and birds recruits brain areas involved in face recognition. Nat. Neurosci., 3(2) 2000, pp 191-197.

[14] F. Crick and C. Asanuma, "Certain aspects of the anatomy and physiology of the cerebral cortex", in "*Parallel Distributed Processing vol. 2*" MIT Press, 1986.

[15] K. Tanaka, "Neuronal mechanisms of object recognition", Science 262, 1993, pp. 685-688.

[16] D. C. Van Essen and C. H. Andersen, "Information processing strategies and pathways in the primate visual system", in *An Introduction to Neural and Electronic Networks*, Academic Press, 1995.

[17] E. M. Calloway, "Local circuits in the primary visual cortex of the macaque monkey", Annual Reviews of Neuroscience 21, 1998, pp. 47-74.

[18] L. A. Coward, "Simulation of a Proposed Binding Model", *Brain Inspired Cognitive Systems 2004*, L. S. Smith, A. Hussain and I. Aleksander, (editors), University of Stirling: Stirling

[19] L. A. Coward, T. D. Gedeon and U. Ratanayake, Managing Interference between Prior and Later Learning. Lecture Notes in Computer Science 3316, 2004, pp 458 - 464.

[20] S. Franklin, B. J. Baars, U. Ramamurthy, and M. Ventura, "The role of consciousness in memory", *Brains, Minds and Media* [Online], 2005. Available: http://www.brains-minds-media.org/archive/150

[21] A. Hyvärinen, J. Karhunen and E. Oja, Independent Component Analysis. (1999). New York: Wiley.

[22] L. A. Coward, "Pattern Thinking". 1990. New York: Praeger.

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