

PARTIAL DRAFT

The dynamic brain model

Alan H. Bond¹

*Center for Cognitive Phenomics, Neuropsychiatric Institute,
University of California at Los Angeles, Los Angeles, California 90095*

Abstract We briefly summarize our overall thinking and design for what we will call the *dynamic* version of our brain model. By this we mean that the model now takes into account memory and thinking of several kinds, and has the ability to learn new knowledge. Our intention is that it provides for:

- (i) short term memory, working memory, episodic memory and long term memory,
- (ii) goal-driven problem solving with elaboration of ideas and choice of courses of action,
- (iii) mental imagery and top-down control of perception and eye movement,
- (iv) routinization as procedural memory, but in particular
- (v) the integration of all these different functions, processes and abilities into a unified working model of the brain, its structure and function.

This is an extremely ambitious objective, and this brief report gives an outline of our current ideas. We believe we have now reached sufficient clarity and understanding of the issues, phenomena and required mechanisms that a trial implementation can be attempted.

Motivation and overview

We have described a simple model of the neocortex [Bond, 1999] [Bond, 2004b] [Bond, 2004a] based on the neural architecture of the cortex. This enabled us to model some simple social behaviors [Bond, 1996] [Bond, 1999] [Bond, 2004a] problem solving strategies [Bond, 2002] and language processing [Bond, 2005a].

We learned a great deal from this research, but the model used manually coded knowledge in each module. It had some memory in each module, mainly short term but some long term, however it had no episodic memory learning mechanism. We therefore wanted to extend the model to allow it to learn new knowledge autonomously and to be able to use this new knowledge in solving problems. We studied learning by doing and were lead to an approach to episodic memory and its use in forming new planning rules [Bond, 2005c]. We also developed some ideas concerning learning procedural memory in the basal ganglia, and coordination of its use with cortical activity [Bond, 2005b].

¹*email address:* alan.bond@exso.com

We have also developed ideas for:

- (i) short term memory, long term episodic memory and long term memory for data items, their access and use,
- (ii) top down visual attention and eye movement,
- (iii) mental imagery.

This paper briefly describes all these ideas and how they can form an integrated set of mechanisms which all run on an augmentation of our existing brain architecture.

Memory in modules We assume that modules contain associative memories, so items can be stored and then they can be retrieved by giving a cue which is a part of the item.

We already have some experience with this in our existing model:

- (i) there may be more than one item matching the cue, in which case in the current model we execute all rules matching all of them.
- (ii) on entry of an item, it can match an existing stored item in various ways. We identified four possibilities,
 - (a) identical, in which case update the stored item. So incoming `color(dog,black,0.8)` and stored `color(dog,black,0.6)` would give something like `color(dog,black,0.6)` updated to `color(dog,black,0.7)`.
 - (b) should be replaced, in which case replace the stored item incoming `action(alan,walking)` and stored `action(alan,sitting)` would replace `action(alan,sitting)` by `action(alan,walking)`, on the other hand `action(alice,walking)` would have no effect on `action(alan,sitting)`.
 - (c) corresponds to a stored item, then replace For example an incoming item `position(object,[102.0,200.0,0.0])` and stored item `position(object,[100.0,200.0,0.0])` should result in `position(object,[100.0,200.0,0.0])` being replaced by `position(object,[102.0,200.0,0.0])` because the difference in their arguments 100.0 and 102.0 is sufficiently large and also sufficiently small i.e. is within realistic bounds.
 - (d) no existing stored item matching, in which case simply store it.
- (iii) items attenuate with time, with a rate dependent on the datatype, so more permanent items can be set to have attenuation factor 0, and totally transient items which are just in the module for one cycle would have attenuation factor 1.0. I have exponential attenuation until a noise level is reached at which time the item is deleted.
- (iv) a computed item which is not confirmed is kept for a certain period and then becomes refractory for a certain period. This involves storing this information in the

item itself. During the refractory period it is not attenuated. A computed item that is confirmed is simply boosted in weight as long as it continues to be confirmed.

The above was arrived at as a result of getting the model to work, i.e. to carry out behaviors. It may have to be revised in various ways. In the dynamic model, we will need to have short term and long term memory items. The latter will attenuate but will not be deleted but remain at some low value. The use of a data item does not boost its value in the existing model, but now I think it probably should, particularly for long term memory items. This is of course priming. With a uniform attenuation method, a temporal series of different primed items will probably form a time dependent series with decreasing weights, giving some temporal information, but the main temporal information would come from its connection to episodic memory.

It also seems that we will need to store module events in the module itself, we will discuss this in the episodic memory section.

Working memory by most accounts is really a frontal process which primes and maintains an item in a posterior area, so this would just be some rule being repeatedly activated, probably in the planning module as part of an evoked plan.

Almost all kinds of neural nets will work as associative memories, where a partial record will evoke some or all of the stored records which match to it. Or one can think of it as completing the partial record or produce a complete record. So this is not so far from rule execution. One main issue is that we would like to transition to rules which do not have repeated variables, in which case rule execution could be achieved with a simple associative match. Since rules are created by the system, removing repeated variables will be one of our aims in the design.

Finally, note that each module has more than one datatype which it stores, and indeed one can view its activity as that of combining some given set of data types to produce a new datatype.

Episodic memory We have discussed our approach in a recent paper [Bond, 2005c]. I will not repeat all that discussion here, and assume the reader will read it.

Modular events. We will have modular events which occur in each module and the module also constructs a representation of the event. This involves only information available within the module, however this does include inputs received and outputs generated. Different modules may produce their modular events differently. One main

example would be an auditory area involved in the formation of phonemes and corresponding to what psychologists call the phonological buffer. This seems to form a record with 14 seconds of processed sound in the form of phonemes.

Modular events however should probably be events in the sense of state \rightarrow change, and therefore this corresponds pretty well to a rule instantiation. So, our initial idea is to take the modular event to be a representation of the dominant rule instantiation. This would have the instantiated left hand side, in other words the items in the state that caused this rule to fire, and the instantiated right hand side, in other words, the effect of the rule in the form of the items constructed by its action.

Some people have suggested that the entire modular event is not sent to the event, but instead a key by which it can be retrieved from the module store. Further, some people suggest that this key should be a random pattern of some kind. We are not sure about any of this. First, we would prefer that a key be something which is not an arbitrary token but be a summary of the event, so that it has meaning, it can be used in logic and it can be constructed and used for retrieval. Second, the neuroanatomy suggests that a lot of data is sent from the modules to the event module, There is in particular massive neural connectivity from scene and object identity visual areas. In any case we probably don't have to decide now, and we can probably change it later without having the change the rest of the system design.

In addition, these modular events will be stored in the module, and provided they are reinforced and integrated by the event module (corresponding to part of the hippocampal complex), they will become long term memory items.

Events. The event module combines all the modular events it receives into one big mental event, using a fixed record structure.

```
event(evkey,[sp1,obj1,obj2,obj3,goal1,wgoal1,cec1,pcec1])
```

This includes the current plan step, the currently activated context, the current goal, and the current working goal.

A context is for us a set of related plan rules, so they are to be evoked together. We are also hoping to be able to represent constraints to be enforced as plan rules also. (This corresponds somewhat to a Newell problem space but is a little more general.) So the entire package is evoked in the context memory and sent to the plan module to be executed. So plans are stored in the context memory and executed in the planning module, and

these correspond to ventral prefrontal and dorsolateral prefrontal respectively, although the medial area SMA may also be involved in plan execution.

The event has an event key which is computed by the event module. The idea is that this event key is a summary of the event, the main things that caused it and the main things it caused.

The main idea of that these event records are stored in the event module, forming the cognitive map. Then when an event is retrieved by an incoming cue it can generate the different fields of its record. It could receive the event key and then generate one, some or all of the modular events, which would be sent to the corresponding modules and activate corresponding representations. It could also receive, from planning or from an individual module, a modular event, and it could complete the event by finding the other components of the event and it could also activate the entire event and its connections to other events in the cognitive map.

Episodes. We derived a form for episodes:

episode(epkey,[epkey1,epkey2,epkey3,epkey4]).

So episodes are limited in size to 3, 4 or 5 events. This means that next relations, i.e. which event is next after a given one, are not so difficult to manage since they are properties of this fixed structure.

So the cognitive map includes episodes as well as events. Our intention is that queries about events can be answered by the event module using the cognitive map. There will be a limit to the depth and length of retrieval that the event module can carry out in response to one goal. What will then happen is that there will be iteration between planning and the event module to examine the results and then to generate another query iteratively in order to answer more complex questions. This issue has been investigated by Burgess and Shallice [Burgess and Shallice, 1996].

Plans

My initial idea is that plans would be learned from experience, and that plan representations would be abstracted from representations of episodes.

There is an important issue of control, when is a given plan evoked, what causes it to be continued, what causes it to be suspended or wait and what causes it to terminate, and what forms of data are needed to keep track of all of this.

Classically, a plan would be evoked by the existence of a goal and would be terminated

when that goal no longer exists, either by being attained or by being superseded by some other goal. However we would like to be able to execute sequences of actions without a driving goal, although at some level of planning there should presumably be some kind of overall goal in existence.

We assume initially at least that the structure of plans is similar to the structure of episodic memories, so that it has some nesting of subplans within plans and also sequencing, but sequences are limited in length to 3 or 4 steps, longer sequences being achieved by using nesting with sequencing.

Plans would be constructed by abstraction from episodes, so that irrelevant detail would be removed and values combined to give sets of possible values. Thus episodes:

goal(own(bike)):go(Walmart), buy(bike), eat(ice_cream).

goal(own(clock)):go(Sears), buy(clock).

might be abstracted to:

goal(own(Object)):go(Store),buy(Object).

It is however pretty clear that one does not learn control principles for executing a particular plan, so we will assume that there are basic control methods which are used to execute plans, including mechanisms for selecting plans, initiating them, continuing them, suspending them and terminating them, and these control methods are innate. They are probably quite weak and general, so we do not assume any complex mechanisms of any kind. What will be learned is just the information needed for these control methods to do their work. This information has to be available at the time the episode occurs.

Thus an episode should have information on:

what goals and behaviors this episode is relevant to

what conditions caused the episode to begin

what conditions are needed for the episode to continue

what effects resulted from the episode occurring.

There may be goals in operation at the start of the episode, but also these goals may only be deduced during the abstraction of a set of episodes to form a plan.

As we saw in the discussion of episodic memory, we are taking the liberty, supported by the published neuroanatomical connectivity data of Kobayashi and Amaral [Kobayashi and Amaral, 1999], of assuming that the episode representation contains information on the goal, working goal, plan step and currently evoked context. Thus the contexts derived from these episodes will also have the same kinds of information.

References

- [Bond, 1996] Bond, A. H. (1996). A Computational Architecture for Social Agents. In *Proceedings of Intelligent Systems: A Semiotic Perspective, An International Multidisciplinary Conference, National Institute of Standards and Technology, Gaithersburg, Maryland, USA, Oct 20-23*.
- [Bond, 1999] Bond, A. H. (1999). Describing Behavioral States using a System Model of the Primate Brain. *American Journal of Primatology*, 49:315–388.
- [Bond, 2002] Bond, A. H. (2002). Problem-solving behavior in a system model of the primate neocortex. *Neurocomputing*, 44-46C:735–742.
- [Bond, 2004a] Bond, A. H. (2004a). A Computational Model for the Primate Neocortex based on its Functional Architecture. *Journal of Theoretical Biology*, 227:81–102.
- [Bond, 2004b] Bond, A. H. (2004b). An Information-processing Analysis of the Functional Architecture of the Primate Neocortex. *Journal of Theoretical Biology*, 227:51–79.
- [Bond, 2005a] Bond, A. H. (2005a). A psycholinguistically and neurolinguistically plausible system-level model of natural-language syntax processing. *Neurocomputing*, 65-66:833–841.
- [Bond, 2005b] Bond, A. H. (2005b). Brain mechanisms for interleaving routine and creative action. To be presented at CNS*05, Madison, Wisconsin, July 2005.
- [Bond, 2005c] Bond, A. H. (2005c). Representing episodic memory in a system-level model of the brain. *Neurocomputing*, 65-66:261–273.
- [Burgess and Shallice, 1996] Burgess, P. W. and Shallice, T. (1996). Confabulation and the control of recollection. *Memory*, 4:359–412.
- [Kobayashi and Amaral, 1999] Kobayashi, Y. and Amaral, D. G. (1999). Chemical neuroanatomy of the hippocampal formation and the perirhinal and parahippocampal cortices. In Bloom, F. E., Bjorklund, A., and Hokfelt, T., editors, *Handbook of Chemical Neuroanatomy, Volume 15: The Primate Nervous System, Part III*, pages 285–401. Elsevier Science Publishers B.V.