# Context Nodes in the Operation of a Long Term Memory Structure for an Evolutionary Cognitive Architecture

Richard J. Duro
GII-Universidade da Coruna
Spain
richard@udc.es

Jose A. Becerra
GII-Universidade da Coruna
Spain
ronin@udc.es

Juan Monroy
GII-Universidade da Coruna
Spain
juan.monroy@udc.es

Luis Calvo
GII-Universidade da Coruna
Spain
luis.calvo@udc.es

## ABSTRACT

This paper describes the creation and use of context nodes, or Cnodes, as an integral part of the structure of a network based Long Term Memory that has been constructed within the Multilevel Darwinist Brain cognitive architecture. Context nodes are networks with multiplicative inputs that support the storage of context related information, that is, a Cnode relates the world the system is in, as well as the system's goal and current state, to the most adequate policy to operate in this context in terms of its previous experience. These structures provide a simple, yet very effective way of retrieving (or activating) long term memory or experience based information when part of a context is detected. A simple example of the operation of the Long Term Memory using Cnodes is presented and discussed.

## CCS CONCEPTS

• **Computing methodologies → Cognitive robotics**;

## KEYWORDS

Long term memory; cognition; Multilevel Darwinist Brain; network memory.

**ACM Reference format:**
Richard J. Duro, Jose A. Becerra, Juan Monroy, and Luis Calvo. 2017. Context Nodes in the Operation of a Long Term Memory Structure for an Evolutionary Cognitive Architecture. In *Proceedings of Genetic and Evolutionary Computation Conference, Berlin, Germany, July 2017 (GECCO'17),* 5 pages.
https://doi.org/10.1145/3067695.3082465

## 1 INTRODUCTION

Long term memory (LTM) is critical for addressing lifelong learning and cognition [1]. However, in most of the work on cognitive architectures, the LTM has been taken as a passive storage container for knowledge, much like a hard disk, to be used by other parts of the architecture. In other words, a computer architecture-like analogy of the mind has been the predominant paradigm in this regards.

However, authors such as [1] and [2] argue that to achieve the properties displayed by biological systems, such as adaptability, flexibility and robustness, memories need to be a distributed and active component of cognition situated within the perception-action cycle of adaptive behavior and not only passive elements that are addressed by other cognitive structures just to retrieve a piece of knowledge that is required by some other process [3]. In other words from this point of view, LTM is central to a cognitive architecture and the remaining components are there to service it.

Following this line of thought, and taking into account that the final function of a cognitive architecture is to provide a means for a motivated system (a system that has goals) to choose actions that allow those goals to be fulfilled [4], the operation of a cognitive architecture revolves around the LTM and it is about appropriately deciding on what actions (or policies) to choose each instant of time. These decision processes involve two main concepts: prospection and experience.

Prospection is related to the prediction and evaluation of future states when actions are performed. This allows the system to select among the potential actions or policies as a function of the expected fulfillment of its goals in a deliberative process whereby it performs prospection of multiple actions and chooses the best evaluated one. Consequently, prospection requires performing predictions into the future, and to allow for predictions the system must generate models (models of the behavior of the world and its elements, of itself. In general, we will call these models forward models). It also requires being able to evaluate the goodness of the states it predicts in terms of expectancy of achieving a goal so that it can decide on the state it should aim for. These evaluations are carried out through the use of so called value functions.

Experience, on the other hand, is not about predicting the future, but rather, it is about compressing the past into reusable relationships that allow the system to replicate a behavior that was successful in the past in a context that is similar to the current one. Thus, it has to do with storing the components that were active at that time and the relationships the system finds among these knowledge components (models, policies, perceptual classes, etc.), that is, its operational context when it is successful at achieving a goal (or, in some cases, even unsuccessful). These relationships allow the system to recall an action or policy as a function of the


*GECCO'17, July 2017, Berlin, Germany*

ACM ISBN 978-1-4503-4939-0/17/07…$15.00
https://doi.org/10.1145/3067695.3082465

context it is in. Context, in this case, means the world it is operating in, its goal, its current perception and the policy or action that was being executed.

Both of these decision making approaches require of a long term memory where the system can store what it has learnt. In other words, where it can store knowledge elements such as models, policies, perceptual classes, value functions, etc. in whatever representation it is using (ANNs, rules...). Additionally, to be able to implement experience based mechanisms, and as indicated before, it also needs to store the experience based relationships among these knowledge elements. It needs to somehow represent the contexts in which these elements were successfully active.

This paper is concerned with the introduction of the concept of Context Nodes within the LTM structure of the Multilevel Darwinist Brain cognitive architecture in order to provide a simple and learnable mechanism that allows for the storage of the relationships among components in a manner that is stable and generalizable. We will also present the results of an initial experiment carried out to demonstrate their effectivity. Thus, section 2 is devoted to a brief introduction of the Multilevel Darwinist Brain. The basic LTM structure and the concept of Cnode are described in section 3. An example of the use of Cnodes in a real robot is presented in section 4. Finally, some conclusions are provided in section 5.

## 2 MULTILEVEL DARWINIST BRAIN

The Multilevel Darwinist Brain (MDB) is a cognitive architecture whose operation is fully described in [5]. The MDB is not intended as a biologically plausible path, but rather, as a computationally effective way of providing the required functionality in real time robotics. It follows a developmental approach and it is based on 4 basic types of elements:

- Models: prediction structures in the form of forward models and satisfaction models (value functions in reinforcement learning terms) that are usually instantiated as Artificial Neural Networks. They conform the declarative knowledge acquired through interaction with the world. MDB relies on evolutionary algorithms for model learning,
- Policy or Behavior: a policy is a decision structure that needs to be learnt and that provides the action to be applied in time t+1 from the sensorial input in t.
- Episodes: real world samples that are obtained from the robot sensors and actuators after applying an action. Typically, within the MDB these episodes are made up of the sensorial information plus the applied action in time t and the sensorial information including sensed rewards or satisfaction derived from the execution of the action in time t+1. These episodes are used as targets for model learning.
- Memories: two main kinds of memory elements were considered in the first implementations of the MDB: Short-Term (STM) and Long-Term (LTM)
  - The STM is made up of a model memory, which contains models and behaviors that are relevant to the current task, and an episodic buffer (EB) that stores the last episodes experienced by the robot. The EB has a very limited capacity according to the temporal nature of the STM.
  - The LTM is made up the models that have been consolidated due to their significance and reliability, and the consolidated behaviors.

Additionally, we could consider a fifth element, Perceptual Class. A perceptual class is really an area of the perceptual space whose components behave in a similar way under some circumstances (for instance, they may be the domain of the same policy or forward model).

In terms of operation, and very briefly as the full details can be found, for instance in [5], the MDB interacts with its environment by performing actions, these produce new perceptions and satisfactions or rewards (when appropriate) which are stored together in the episodic memory as episodes. The elements in the episodic memory (the ground truth) are used to determine the fitness of evolving populations of models by testing them over the episodic memory instances. These populations are evolved just for a few generations for every interaction with the environment (we do not want the models to converge to a particular content of the small episodic memory, but rather to slowly converge to the series of episodic memories it is being exposed to). The best current models are selected and used in order to evaluate possible policies in a second evolutionary process. These policies are applied to the forward model in order to determine their effect and this effect is evaluated using the current satisfaction model or value function. The best policy is chosen and is used to select the next action to apply to the environment. This policy will be active until a new policy that improves on it is provided by the evolutionary component. Those models and policies that are successful are copied to LTM for their preservation and possible reuse.

## 3 MDB LTM STRUCTURE AND CNODES

The main focus of this paper is on the long term memory structure of the MDB. This structure, as indicated before, should cover two needs. On the one hand, it must be used as a storage facility for the different knowledge elements acquired by the system. These elements are: Forward models ($FM_r$), goals ($G_j$) and their associated Value Functions ($VF_j$), Policies ($k$) and, finally, Perceptual Classes ($S_t$). On the other hand, it must provide for the storage of experience by establishing context based relationships among the elements in LTM.

By context we mean under what circumstances something has occurred or a policy has been applied successfully. Thus, in addition to remembering a given element of the architecture, such as a policy or forward model, we would like to remember under what circumstances it led to success (in terms of achieving goals or improving the state of the system), so that when those circumstances arise again, we know that it could be used. For instance, in terms of a policy, we would like to know in what world, for what perceptions and seeking what goal it was useful. Consequently, the context of policy k is given by the tuple $\{FM_r, G_j, S_t\}$ and when this tuple arises during the operation of the system, we can directly infer that k may be the policy to choose.

A first approach to establishing these context relationships (as presented in [6]) could be just to create direct links among the elements that participate in the successful context. As the system finds itself in different situations, the links that participate in successful

contexts are reinforced, and those that lead to unsuccessful results are debilitated. Basically, in this approach, the level activation of an elements is given by a function of the sum of the activations of the elements linked to it weighed by the strength of their links.

This approach is valid in simple scenarios, however, when life-long learning is addressed, problems start to arise in terms of drift of the weight values, forgetting relationships, and other artifacts. Thus, for the very long term we cannot rely only on this type of connectivity to determine relationships, and a more specific way of storing these context relationships must be sought. This is where the concept of Context Node (Cnode) arises.

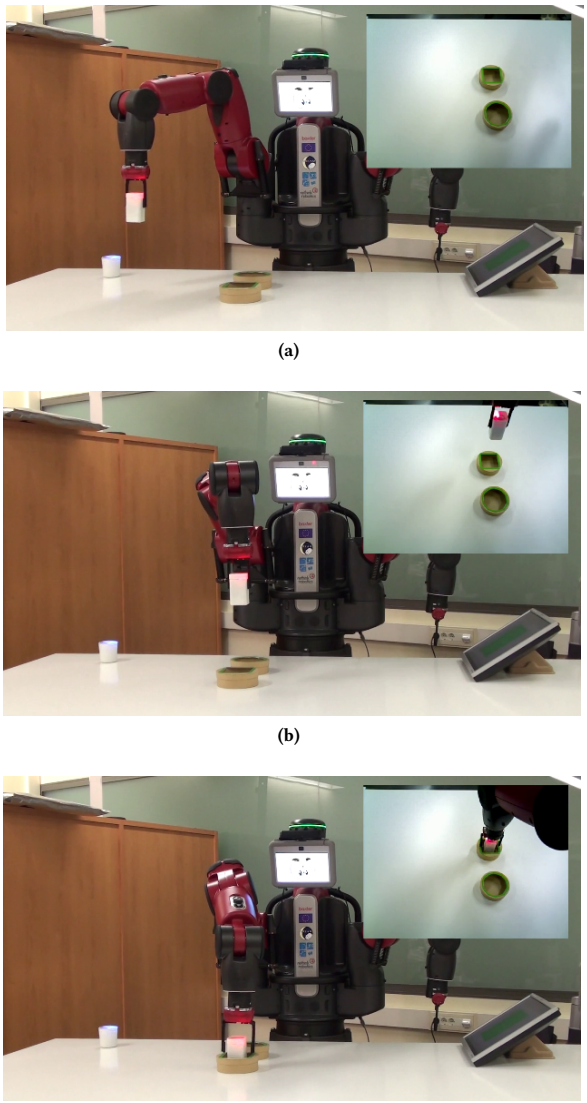
Cnodes are a new type of element within the LTM structure which can have the same type of representation as the other types of elements we have cited before (ANNs, rules sets, functions...) and whose function is to store a context. Cnodes are product units, that is, their activation level is given by a function of the product of their inputs (which are the activations of other elements of the LTM) and their output is connected to the element whose context we want to store. For instance, in the case of a Cnode for a policy, its inputs would correspond to the activations of a forward model (which determines the world the system is in) a Goal or combination of Goals (which determine what the system strives for) and a Perceptual class or a set of Perceptual classes (which determine in what states the policy has been successful before for this world and goal) and its output would provide an activation signal to the policy.

A Cnode is created when a context that needs to be remembered occurs and there is no previously created Cnode that covers this situation or that could be made to cover this situation. Things that need to be remembered can be determined by rewards or emotions. That is, in the simplest case, when the system obtains a reward, the context that was active in this situation is a candidate for Cnode incorporation. Thus, Cnodes permit remembering, in the very long term, relationships and contexts that led to successful results. Once a situation or context is correctly represented by a Cnode, whenever the context occurs again, the appropriate policy can be directly activated.

## 4 EXAMPLE

With the objective of showing how the LTM works when considering Cnodes within its structure, we present a very simple experiment using the Baxter Robot. The images of Figure 1 display the scenario used for this experiment. A Baxter robot perceives through its camera a workspace in which we find two boxes, one with a round hole and one with a square hole. Two objects are available to the robot, a cube and a cylinder, which fit in both boxes. These objects have red or blue leds on top of them. The robot's perceptual system is not capable of extracting shapes from the camera image information, it can only detect the color of the objects. Thus, the only way the Baxter can tell whether an object is a cylinder or a cube is by means of its color.

With this basic scenario we have run an experiment that considered two different worlds and two different tasks. The tasks are given by their goals, that is, in one case reward is obtained when a hole (the screen in the scene indicates which hole) is full with the correct shape and in the other case reward is obtained when all of the holes are empty. With respect to the two worlds, in one world, cylinders have a red light on top and cubes a blue one. In the other world, the lights are assigned the other way around (cylinders are blue and cubes red). As a consequence, depending on what world the robot is in, it has to pick blue or red objects to fill square hole. The experiments can start from any initial state, including one of the holes having the wrong object in it. This situation would imply taking it out before being able to put the other one in. To be as realistic as possible with such a simple setup, the world the robot is in will be switched after a random number of interactions. Also the goal of the robot will switch after a different random number of interactions. Consequently, the robot will have to concurrently learn to achieve maximum reward in four goal-world combinations.



(a)

(b)

(c)

**Figure 1: Three instants of one of the interactions of the robot with the world.**

This simple experiment is very well suited to our purposes as we can determine in advance all the possible contexts that should lead to the creation of Cnodes in the different worlds and for the different goals, that is, all of the relevant events. In terms of goals and worlds, we have four combinations. For each of these cases, there are six relevant situations, leading to a total of 24 Cnodes that should be obtained through interaction with the different worlds with different goals.

Finally, as we are only interested in showing the operation of the LTM, we assume that the individual forward models, policies, perceptual classes and goals (represented here by their associated value functions) have been previously learnt and stored in the long term memory when addressing other tasks and in this experiment we only consider the relationships (Cnodes) that are obtained for these problems.

A sequence of three views of a run of the robot's LTM is displayed in Figure 2. The top view corresponds to the structure of the LTM at the beginning of the process, where there are no Cnodes (no relevant events have been detected by the system). The view in the middles corresponds to the LTM after 70 interactions of the robot with the different world-goals. Finally, the view at the bottom of the figure represents the state of the LTMA after 167 interactions.

The sequence starts with an LTM that is blank in terms of Cnodes, the robot has not started to interact with the world in order to be able to detect relevant events and create the corresponding Cnodes. It is basically performing a trial and error process. As shown in the second image, after 70 interactions with the different worlds, the robot has already learnt some Cnodes, and when the associated situations arise, the robot chooses the right policy directly. Exploration is only taking place with regards to the ones it has not learnt yet. Finally, after 167 interactions, the robot has learnt all of the possible relevant events in the environment and, thus, when faced with any state in any of the four world-goal combinations, it directly chooses the optimal policy. It is very important to note here that this process was quite efficient and did not take very long.

Figure 3 shows the evolution of the number of Cnodes acquired by the system in another run as it interacts with the different world-goal combinations it is faced with. As stated before, there are four world-goal combinations (fill the hole-cube is red/fill the hole-cube is blue/empty all holes-cube is red/empty all holes-cube is blue). They are denoted in the figure with the numbers from 1 to 4. It is clear in this figure, that initially, the robot explores a little bit and, after a while it starts discovering situations that provide rewards and thus it begins creating Cnodes for them. After about 37 interactions it has learnt all the rewarding contexts in the first world-goal setting. Then the world-goal combination changes and it has to learn new contexts for this case, which it does efficiently. Then the first world-goal combination becomes active again. As it has already learnt all the rewarding contexts in this setting, the robot uses the information it has in LTM to perform perfectly, but it does not create any new Cnode. The process is repeated as new world-goal combinations arise until it has created the 24 Cnodes that allow it to perform optimally in the four worlds. Figure 1 shows three snapshots of one of the interactions of the robot with the world doing one of the tasks.

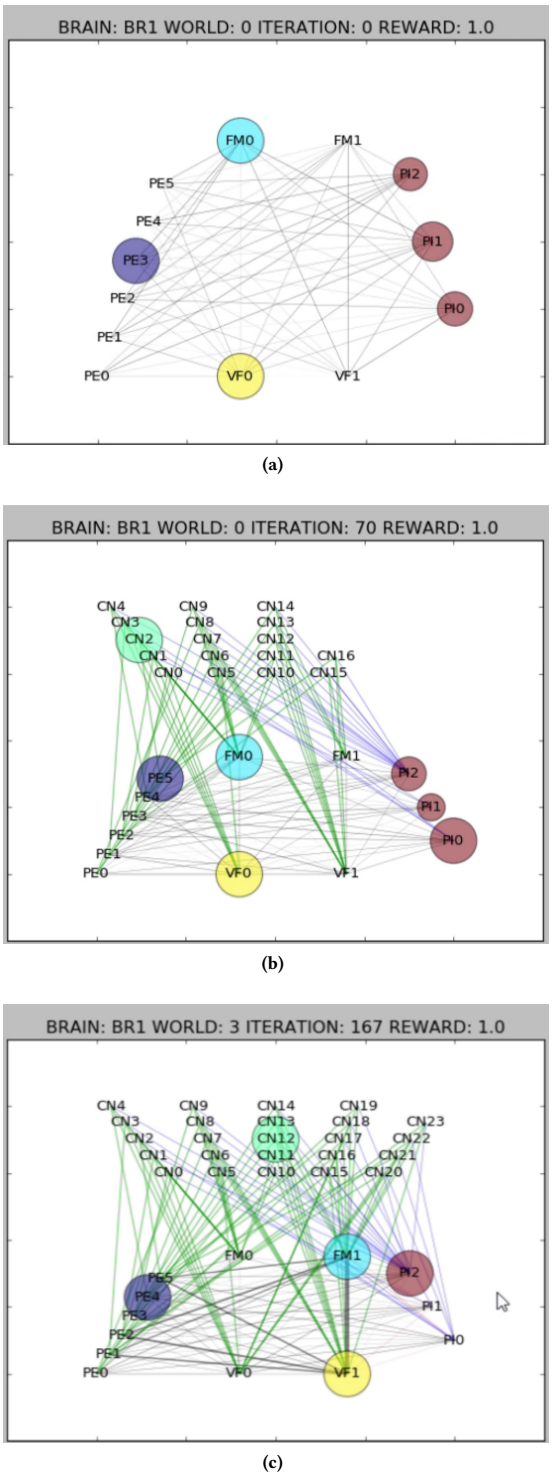



**Figure 2: State of the LTM at three different points of the interaction of the system with the world.**

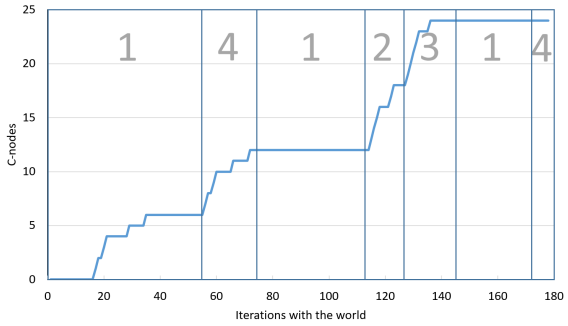



**Figure 3: Evolution of the number of C-nodes acquired by the system as it interacts with the world. The numbers on top designate which one of 4 different world-goal combinations is active.**

## 5 CONCLUSIONS

In this brief paper we describe the concept of Cnode or Context node as one very convenient element to have in the Long Term Memory structure of a cognitive architecture. This type of element helps to store relationships among knowledge elements that have led to successful or relevant results in the past, thus allowing for easy retrieval of relevant knowledge when the context arises again. We have presented a simple example to show how this would work in the framework of the Multilevel Darwinist Brain architecture.

## ACKNOWLEDGMENTS

This work has been partially funded by the EU's H2020 research programme grant No 640891 (DREAM) as well as by the Xunta de Galicia and the European Regional Development Funds grant redTEIC network (ED341D R2016/012).

## REFERENCES

[1] Wood, R., Baxter, P., Belpaeme, T.: A review of long-term memory in natural and synthetic systems. Adaptive Behavior 20–2, 81–103 (2012)
[2] Fuster, J.: Cortex and memory: emergence of a new paradigm. Journal of cognitive neuroscience, 21âĂŤ11, 2047âĂŤ2072 (2009)
[3] Fuster, J.: Network memory. Trends in neurosciences, 20–10, 451–459 (1997)
[4] Langley, P., Laird, J.E., Rogers, S.: Cognitive architectures: Research issues and challenges. Cognitive Systems Research. 10, 2, 141âĂŞ-160 (2009)
[5] Bellas, F., Duro, RJ., Faiña, A., Souto, D.: Multilevel Darwinist Brain (MDB): Artificial evolution in a cognitive architecture for real robots. IEEE Transactions on autonomous mental development, 2, 340–354(2010)
[6] Duro, RJ., Becerra, JA., Monroy, J., Caamano, P.: Considering Memory Networks in the LTM Structure of the Multilevel Darwinist Brain. Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion,1057–1060 (2016)