International Journal of Neural Systems, Vol. 29, No. 6 (2019) 1850053 (26 pages)
© The Author(s)
DOI: 10.1142/S0129065718500533

# Perceptual Generalization and Context in a Network Memory Inspired Long-Term Memory for Artificial Cognition

Richard J. Duro*, Jose A. Becerra, Juan Monroy and Francisco Bellas
*Universidade da Coruña, Spain*
**richard.duro@udc.es*

Accepted 22 October 2018
Published Online 7 January 2019

In the framework of open-ended learning cognitive architectures for robots, this paper deals with the design of a Long-Term Memory (LTM) structure that can accommodate the progressive acquisition of experience-based decision capabilities, or what different authors call "automation" of what is learnt, as a complementary system to more common prospective functions. The LTM proposed here provides for a relational storage of knowledge nuggets given the form of artificial neural networks (ANNs) that is representative of the contexts in which they are relevant in a configural associative structure. It also addresses the problem of continuous perceptual spaces and the task- and context-related generalization or categorization of perceptions in an autonomous manner within the embodied sensorimotor apparatus of the robot. These issues are analyzed and a solution is proposed through the introduction of two new types of knowledge nuggets: P-nodes representing perceptual classes and C-nodes representing contexts. The approach is studied and its performance evaluated through its implementation and application to a real robotic experiment.

*Keywords*: Long-term memory; cognitive architecture; network memory; perceptual generalization; context.

## 1. Introduction

Cognition encompasses a series of behavioral processes in animals, involving gathering sensory information, converting it into perceptions, making decisions and producing actions, that allow them to deal with dynamic or changing environments.[1,2] A cognitive architecture is a system that tries to implement cognition. In Sun's words, "a cognitive architecture is a broadly-scoped domain-generic computational cognitive model, capturing the essential structure and process of the mind, to be used for broad, multiple-level, multiple-domain analysis of behavior".[3] Thus, as a generic computational model, it is a basic operational structure made up of different processing elements that is not specific to any particular task or domain and which can be adapted to any task or domain through its instantiation with knowledge.

A particular cognitive architecture is just an instance of the structures required to support cognition and their organization. It defines what components participate, what capabilities they display in terms of how knowledge is acquired, represented and acted upon, and how they interact so that the whole chain, from sensing to acting, can successfully operate.[4–6] A cognitive architecture's ultimate function is to provide a means for a motivated system (a system that has goals) to choose actions that allow those goals to be fulfilled. Thus, appropriately deciding on what actions to choose each instant of time is what a cognitive architecture is about.

*Corresponding author.

These decision processes almost always revolve around two main concepts: prospection and experience. Prospection has to do with the anticipation or prediction of future states so that they can be evaluated in order to allow for a selection among the potential actions or policies as a function of the expected achievement of its goals.[7,8] This deliberative process requires performing predictions into the future, usually carried out by models (world models, internal models), and evaluations of the predicted states.

On the other hand, experience has to do with direct statistical associations or relationships the system has found among its knowledge components or knowledge nuggets (models, policies, perceptual classes, etc.) when it was successful at achieving a goal (or, in some cases, even unsuccessful). These relationships allow the system to directly choose an action or policy without any prospection or evaluation if it can determine the context it is in, that is, if it can determine in which world it is operating, what its goal is and what is its current perception. In the terminology of many authors, the decision process has been automated as it does not require any prospection for its completion.[9]

Both of these decision-making approaches call for the availability of different components or knowledge nuggets, such as models of the world and the organism, so that predictions can be made, or representations of the world or of the different goals, so that experiential relationships can be established. Thus, a really adaptive cognitive structure must provide for the capability of learning what are basically associative relationships (among actions and their consequences in state space or among context elements and the actions that led to achieving goals) as well as for a way to efficiently structure and store these components and their relationships so that they can be easily used.

Traditional general purpose cognitive architectures in the literature such as ACT-R,[10] CLARION,[3] EPIC,[11] GLAIR,[12] 4CAPS[13] or SOAR,[14] implement these decision processes within a set of internal structures and representations that encode the different knowledge nuggets as production rules or other types of externally imposed symbolic representations. These representations lend themselves to very explicit and self-explanatory or self-descriptive mechanisms to work with this knowledge, as well as to relatively straightforward computational indexing mechanisms when storing the knowledge nuggets in memory. However, they lack flexibility and robustness when handling perception and present many problems with respect to grounding and self-acquisition of the structures themselves from interaction with changing environments.

It is for this reason that lately, hybrid approaches have become quite popular, recent examples are architectures like OpenCogPrime[15] or MLECOG.[16] Even leading to the hybridization of some of the traditional architectures.[17,13] They use symbolic representations at higher or more abstract processing levels linking them to lower level behavior like procedural (or even sometimes declarative) knowledge in the form of emergent or connectionist paradigms,[18] such as artificial neural networks (ANNs). This type of low level behavior based structures can be directly obtained by the embodied individual through its sensorial and motor mechanisms from interaction with the environment, or in developmental processes,[19] thus overcoming the grounding problem.

In fact, ANN like representations have become very popular for many complex processing tasks in the last decade, especially since the advent of convolutional networks and other deep learning approaches.[20] This has especially been the case in complex sensor processing tasks such as vision related problems[21] and others.[22]

ANNs provide a relatively efficient and homogeneous approach to learning and extracting knowledge from complex perceptual streams. However, the problem with ANN like representations in cognitive architectures is that they are not easy to manage and integrate due to their lack of self-descriptive capabilities. That is, it is almost impossible to determine what an ANN does or represents without running it over an extensive set of cases. In fact, it is very difficult to say whether two ANNs do the same or different things without running them.

As indicated before, a cognitive architecture must acquire and manage different types of knowledge, mostly in the form of knowledge nuggets such as models, behaviors or policies, goals or value functions, perceptual categories, etc. This knowledge has to be stored and related so that it can be readily used as required by the system. When these knowledge nuggets are represented in the form of ANNs or similar structures, approaches based on computer like memory indexing mechanisms for the storage and integration of the nuggets have been shown to be lacking.[23]

This paper addresses the problem of designing an operational mechanism within a robot cognitive architecture that allows for straightforward experience based automatic or reactive processing when the knowledge nuggets the cognitive architecture manages are given in the form of ANNs. This mechanism is inspired by the concepts of Conceptual Spaces[24] introduced by Gärdenfors and of Network Memories[25] as proposed by Fuster. It revolves around the Long-Term Memory (LTM) and we propose a series of components and functionalities that allow for the integrated and task related formation of perceptual classes as well as context representations. This will permit progressively obtaining associative relationships among these structures and the rest of the knowledge nuggets present in LTM (such as models, policies, goals) in order to facilitate the selection of the appropriate policy or behavior for each situation without interference when the robot is faced with *a priori* unknown sequences of environments. That is, in open-ended learning settings.

To achieve this objective, in the following section we will go back over the neuro-psychological literature and concepts in order to define a set of characteristics or functions an LTM should provide as well as to introduce the basic inspiration to our approach. Section 3 is devoted to the description of the LTM structure and the components we propose as well as its operation. In Sec. 4 we describe in detail one possible implementation of the LTM, which is then used in a set of tests carried out in Sec. 5 over a real robotic problem. These tests are designed to show how the characteristics that were required from the LTM are met.

## 2. Background and Inspiration

### 2.1. *Memory*

A large amount of work has been carried out in the identification of individual cognitive structures from a functional perspective and their purported relations in the human brain as well as in animals. Even though there is no consensus on a single unified architecture, there is a certain amount of agreement on some of its main structures and their basic relationships.

First of all, a set of memory structures working at different time scales have been suggested. These structures include what have been called Short-Term Memory (STM) and LTM.[26,27]

Both, STM and LTM, were initially contemplated as storage facilities where information was kept but not really processed and, surprisingly, many authors in the computational realm still implicitly follow this view. STM holds information for short periods of time and LTM for very long periods of time. Later, several authors started to view STM as a processing part of the cognitive system and, to reflect this, started to refer to it as Working Memory (WM).[28] Currently, STM and WM are not always used as interchangeable synonyms, and some authors consider them two different systems.[29]

An objective of WM is to retain a limited amount of information[30] for short periods of time[31] so that prospective exploration can take place.[7,32] When prospective exploration is the decision process of choice, the smaller the exploration space, the higher the probability of finding a solution. Consequently, having small capacity WMs favors agile learning processes. In fact, humans in general can only learn very simple relationships in one shot, it is through the progressive addition of new information that more complex knowledge can be acquired. Thus, a need arises for a structured or multistep progressive approach to constructing complex relationships.

This is, where LTM comes into play. Well-learned material, held in LTM, can be cued as a unit in working memory and is thus not affected by the limitations imposed by WM.[33] This well-learned material acts as a single component when creating new models or performing prospective processes.

In many cases, one can even do away with prospection. When the relationships are well established in LTM, a decision on an action can be directly reached without any prospective process, by just following the association graph. Some authors call this the automation of learned material and everything that is learned can, with practice, become automated. Schneider and Shiffrin demonstrated the complementarity of prospective and automated or experiential processing.[34,9] Also, Kotovsky and col.[35] demonstrated the enormous benefits of automated processing in problem-solving skills.

The main point here is that, after practice, specific categories of information can be processed with decreasing prospective load, which would require intensive use of WM, through a different type of associative processing that propagates activations in the association graph linking the different cognitive components. This type of processing is what are called experiential decision processes in this paper and their implementation leads to a need for an associative structure that allows the representation of compound knowledge.

From the previous paragraphs it is easy to see that LTM is critical for addressing open-ended learning and cognition,[23] but as humans are not conscious of the contents of LTM except when they are brought into working memory, its critical role in our cognitive activity is often ignored. Most authors creating artificial cognitive architectures have paid very little attention to this system except as a passive storage container for knowledge with discrete encoding, storage, and retrieval functions.[23] A computer architecture like analogy of the mind has been the predominant paradigm in this regards: memory as a hard disk.

De Groot's work[36] and Chase and Simon's paper on chess[37] provided the first insights into the pivotal importance of LTM in higher level cognition. In fact, some researchers started to see LTM as an integral component of cognition, including high level problem solving and other aspects instead of just a mere storage system.

More recently, authors such as Wood and col.[23] or Fuster[25] have argued that in order to achieve properties such as adaptability, flexibility and robustness, biological system memories, and in particular LTM, must be a distributed and active component of cognition situated within the perception-action cycle of adaptive behavior. Consequently, memory is proposed as the central component of any cognitive architecture. Furthermore, authors like Oberauer[38] or Fuster[25] do not even adhere to the classical division of memory into two separate subsystems: STM and LTM. They take STM as the currently activated parts of LTM. Anyway, whatever the view, this general approach purports that some of the most relevant mechanisms for cognition are those related to an associative LTM and its operation.

As a consequence, a need arises to establish a memory structure that can operate as a dynamic associative mechanism that can handle the conjunctive representations of different knowledge nuggets and that supports the decision processes required for the open-ended operation of a cognitive architecture.

When considering conjunctive representations, one has to go back into the associative learning literature to understand the extent and nuances of the topic, as they have a large bearing to the structures that are necessary for their implementation in artificial systems.

## 2.2. *Associative learning*

Associative learning[39–41] is basically defined as the learning process by means of which an association is established between two or more stimuli or a behavior and some stimuli. The idea is that the presentation of a stimulus can activate or inhibit the expectation of another and this relationship is learnable through the creation and strengthening or weakening of associative links among stimuli or stimuli and responses.

Many theories of associative learning that explain a progressively larger number of experimentally detected phenomena have been proposed.[42–48] They all assume that the repeated presentation of two events in succession will result in the growth of a connection, or association, between their internal representations.

Associative learning theories differ in many aspects, but one of the most important is on how they handle the associations that are formed when the first event is a compound of two or more components (stimuli, compound stimuli...). Elemental theories, provide the opportunity for each element of the compound to enter into an association with the representation of the second event.[42–44,47,49–52] They base the response to a set of stimuli using the summation of the activations provided by the association of the individual stimuli to the response. This presents problems when addressing certain situations such as negative patterning and many others as described by Wasserman and Miller.[53]

Configural learning theories arise to solve these problems. They consider that a representation of the entire compound pattern of stimulation that constitutes the first event will be formed associated to the second event. In other words, the whole set of stimuli become a unit with a single association to the response, much like if a new pseudo-stimuli were created and were activated when the set of basic stimuli that make it up is present. This approach started with the work of Gulliksen and Wolfle[54] and almost fifty years later, Bellingham and colleagues[55] demonstrated how a configural version of the Rescorla–Wagner model[42] could be used to explain patterning discriminations. Pearce proposed a formal configural theory of associative learning[45] applying it to a broad range of learning cases such as shadowing, blocking, patterning, etc. More recently, he even suggested a connectionist theory of configural learning.[56]

The key here, is to progressively create and associate different knowledge nuggets in a meaningful and general manner, that is, to provide compact experiential representations so that hypotheses can be made on the actions to take when faced with similar perceptions under different contexts. As Mante and col. indicate, "our interactions with the world are inherently flexible. Identical sensory stimuli, for example, can lead to very different behavioral responses depending on "context", which includes goals, prior expectations about upcoming events, and relevant past experiences. Animals can switch rapidly between behavioral contexts, implying the existence of rapid modulation, or "gating", mechanisms within the brain that select relevant sensory information for decision-making and action".[57] Whatever the mechanisms in natural brains, from an informational perspective, context dependence is an associative learning problem which is often configural in its solution and where gating must somehow be provided by the associative connectivity.

## 2.3. *Constructing a long-term memory*

Consequently, when creating a cognitive architecture all of the processes described in the previous pages must be supported by an underlying operational structure and now, the problem becomes a design or organizational problem. That is, how is an artificial associative LTM to be structured so that it can provide the capabilities that are required from it.

This work takes inspiration from Fuster's network memory model.[58] According to this model, memory consists of the modulation of synaptic contacts between distributed networks of interconnected cortical cells. Memory is achieved through the potentiation or inhibition of synaptic links between neural aggregates as a response to perceptual or other types of activations. These activation patterns, supported by the connections between neural populations, which are acquired through experience, is what he calls memory networks or cognits.[25,7] He defines a cognit as "a network of neuronal assemblies of the cortex that represents sensory stimuli and/or motor actions that have occurred at the same, or nearly the same, time. It is formed by synaptic modulation according to Hebbian principles and is subject to modification by subsequent stimuli and/or actions that are associated with it".

In terms of organization, Fuster arranges cognits in a hierarchical manner between two tiers of cortical association areas: the perceptual tier and the executive system.[59,25] He defines hierarchy in an anatomical manner as the synaptic distance from sensors and actuators (muscles). The higher the level the higher the complexity of the cognits and the abstraction level they represent. On the sensor side (in the posterior cortex), we have the perceptual hierarchy of cognits that represent progressively more abstract categories of sensory-based knowledge. On the executive side (in the frontal cortex), the executive hierarchy of cognits represents progressively more abstract knowledge of action. Fuster also alludes to the heterarchical nature of these tiers, linking multiple levels of the same hierarchy and cognits from both hierarchies.

Cognits may share network nodes as lower level cognits are nested in several higher level cognits. For instance, in the sensory side, a cognit may represent "large size" or "green color" and there are many higher level cognits that may share one of these features such as a "large house" or a "large mountain" or a "green car" or a "green field" or even a "large green car". The same can be extended to the executive hierarchy where a "raise an arm" cognit may be associated to "picking up boxes on top of the cupboard" or to "waving". In fact, nodes could even be shared heterarchically. Anyway, this general framework proposed by Fuster, whereby unstructured networks conform cognits hierarchically through synaptic modulation, is supported by different studies in primates, including humans (for references to these studies see Fuster and Bressler).[7]

A consequence of this model is that memories can be taken as distributed throughout large areas of the associative cortex, with nodes corresponding to neural aggregates with particular processing functions that are linked through synaptic links that connect them, much in the same way as graphs. It follows that, in this view, memory shares the same neurons and networks used by perception. In other words, representation and function have the very same substrate; the latter is the activation of the former within a given spatiotemporal pattern of neural activity. In fact, this model leads to the distributed network structure, hierarchical organization and relational coding of connectionist cognitive models.

From a computational perspective, this approach is similar to that of traditional symbolic based semantic networks, as considered for instance in ACT-R[60] or in CLARION.[61] They use symbols as nodes usually representing declarative knowledge and use the spread of symbol activations to combine them according to the task. However, apart from declarative knowledge, LTM contains procedural knowledge. Both types of knowledge, in order to be appropriately grounded, should arise from the interaction of the robot with the world and, thus, be generally represented in sub-symbolic form (as ANNs, for instance). Consequently, these types of approaches must be expanded and re-examined to allow for autonomously acquired network memories that grow directly from the embodied perceptual apparatus of the system all the way to the executive part of the cognitive architecture, without any explicit externally imposed symbolic structure. In this line, Wood and col. indicate that the storage of semantic information is a "property of the memory system as embedded in the wider cognitive architecture"[23] and not something that is explicitly encoded. In fact, these ideas become very important in architectures such as the one we are working on, which aim at exploiting development as a facilitator of open-ended learning. In this case, LTM becomes one of the most important parts of the architecture as it is where the knowledge the system has acquired, and upon which it must developmentally construct new knowledge, is stored and processed.

From all of this, it can be extracted that a cognitive architecture must basically be a motivated prospection machine that has the capability of progressively establishing associations among knowledge nuggets so that regularities in its interaction with the environment can be "automated", that is, it can be converted into an associative experience based decision machine. This paper concentrates on this "automation" aspect when the knowledge nuggets the cognitive architecture handles are represented as ANNs. It is mostly concerned with the LTM and how to endow it with straightforward mechanisms that allow it to operate as an integral and central part of the cognitive architecture's experience based processing structure without resorting to any externally imposed symbol structure.

## 3. Network Memory Based LTM Structure

As reviewed in the background section, there are several characteristics that seem essential for the operation of a LTM structure within a cognitive architecture:

(1) The LTM should be able to use regularities in its interaction with environments under different motivations to establish associative relationships among the different knowledge nuggets it has acquired. It must be able to use this relationship structure to effectively decide on an action without necessarily having to resort to prospection.
(2) For these relationships to be useful, they should be able to reflect the contexts in which different actions or policies must be executed, supporting configural relationships and processing.
(3) It must also be able to aggregate events into context dependent categories that allow for generalized or more abstract processing. This is especially relevant in continuous domains.
(4) In perceptual terms, these categories must be autonomously obtained by the system, making use of its embodiment and sensors so that grounding does not become a problem.

To this end, this paper proposes a LTM structure designed to accommodate the relationship structure that would be necessary for experience based action selection through the introduction of a series of concepts inspired by the memory network ideas proposed by Fuster.[58]

By experience based action selection we mean that as the system interacts with the world it can relate a perceptual state $\boldsymbol{S}_i$ and a policy $\pi_r$ or action that was successful (even though initially the policy could have been chosen at random) and save this relationship in some type of memory, so that the policy can be reused when the same state arises. In general, this is a bit more complex, as the validity of a policy to produce a result given an initial perceptual state (its repeatability) also depends on the desired result (the goal) and on the world the system is in (we assume the world includes the agent), as different worlds may work differently, e.g. it is not the same to walk on solid ground than on ice.

In general, the world the system is in can be identified by determining which world model is most successful at predicting it or, alternatively, by having a particular sensor that helps identify worlds. A world model is just a forward model that given a state and a policy predicts the next state. In this paper we will mostly refer to them as forward models (FMs).

Thus, it is necessary to relate into a conjunctive representation the perceptual state, $\boldsymbol{S}_i$, the goal, $G_k$, or its related value function, $VF_k$, the policy, $\pi_r$, and the most successful forward model $FM_j$ that have co-occurred when something relevant was experienced by the system. As there is no order in the elements, we will use set notation to represent them. That is, a set of the type $\{\boldsymbol{S}_i, FM_j, G_k, \pi_r\}$ must be identified and stored. This way, when, for instance, the subset $\{\boldsymbol{S}_i, FM_j, G_k\}$ arises, the system can infer that applying policy $\pi_r$, should lead to a successful result, that is, to the same relevant event.

However, just storing this $\{\boldsymbol{S}_i, FM_j, G_k, \pi_r\}$ set does not provide for the coverage of the first characteristic of LTMs as listed above. These type of sets do not represent regularities, just events. Regularities are a function of the frequency of occurrence of events. Consequently, they must be associated to some type of intensity vector $(\boldsymbol{I}_n)$ that represents the relative frequency of the different associations that are present or, at least, that require storage in order to be able to really remember regularities in the system's interaction with the world.

## 3.1. *Association, context and C-nodes*

Defining the associative structure is not straightforward due to the second requirement that has been set above for LTMs. In fact, and as mentioned in the introduction, associative learning theories differ, among other things, in how they handle associations between two events when the first event is a compound of two or more components. Thus, following elemental theories of association,[49,42] one could hypothesize that associations in LTM should be made solely among all of the element pairings in the tuples of the previous section. This would create an association or intensity matrix that would reflect the activation frequencies of the different knowledge nuggets as related two by two. That is, it would provide first-order associations among knowledge nuggets. Obviously, these associations would be strengthened when those instances of the elements co-occurred. Some examples following this approach and using the concept of Network Memory can be found in Ref. 62.

However, this simple connectivity structure does not satisfy all of the requirements that were defined for the type of associative LTM that is being sought. In particular, it does not satisfy the second requirement. The main problem is that only first-order relationships between components can be directly reflected. This implies that higher-order relationships or, more precisely, some types of configural relationships[54,45] among several components are not possible unless they can be constructed as aggregations of first-order associations, which is not always the case. In addition, there is no specific structure that stores the relationships other than the intensity vector (or matrix), which is constantly being modified through the co-occurrences of knowledge nuggets in different contexts as the system interacts with different worlds and tasks (defined by their goals). As a consequence, as the number of worlds and tasks the architecture is faced with grows, the intensity vector tends to drift depending on the sequence in which the system is faced with the worlds or goals it has to work with, and on how long it is presented with each combination (which in real systems cannot be predicted).

Thus, the associative intensity vector or matrix is quite local in time and tends to forget co-occurrence relationships related to world-goal (WG) combinations it has not seen in a while or even get stuck in associations related to WG combinations it has been exposed to for too long or that are simpler. These problems lead to interference based forgetting of previously acquired context relationships and, sometimes, to the inability to exit local minima and learn new context relationships.

To address these issues, a new type of knowledge nugget or LTM component is proposed here. This component is a relational element in charge of encoding relationships among LTM components in a more permanent manner when relevant events occur. In the experiments presented in this paper, relevant events are taken as those that produce rewards. However, any other type of event could be considered (e.g. emotions). The rationale is that whenever a reward is obtained, it could be useful to remember in a relatively permanent manner the context in which this occurred.

This relational nugget has been called "context node" or C-node for short (see Fig. 1 for a depiction of the LTM structure with C-nodes). A context node is a node that is created when the co-occurrence of a series of elements within the LTM leads to a relevant event. All the elements that co-occur are linked to the newly created C-node through weighed connections. To make the representation configural, C-nodes present a product type activation structure, thus allowing for higher-order relationships among knowledge nuggets.

The output of a C-node can be written in general as

$$O_i = \prod_{\text{partial set}} I_{ji} K_j,$$

where the product is over the intensities or weights of the connections $I_{ji}$ between each element $K_j \,\epsilon\, \{\boldsymbol{S}_n, FM_m, G_k\}$ of the partial set associated to C-node$_i$ and C-node$_i$ (as shown in Fig. 1) and $O_i$ denotes the output of C-node$_i$. This output value is taken as an activation value, indicating how active

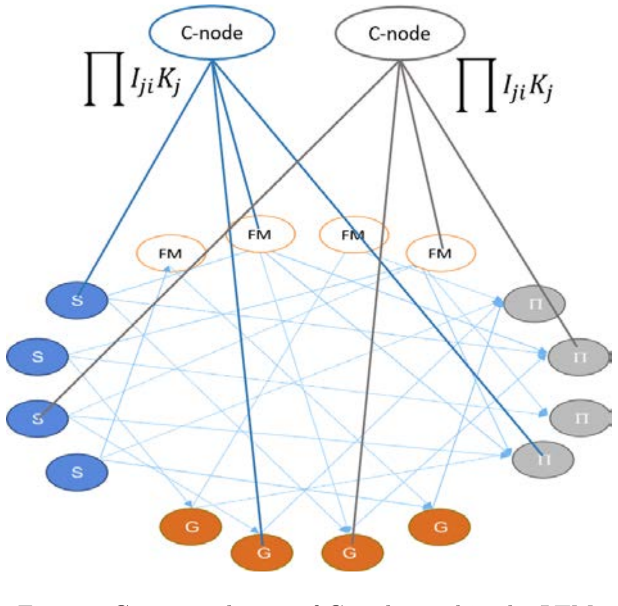


Fig. 1. Conceptual view of C-nodes within the LTM.

the C-node is. It is propagated to those nodes in the LTM to which the output of the C-node is connected. We use the term partial set to denote one of the previously mentioned sets without one of its elements, which will be the one that is activated by the C-node. In particular, in this paper we will concentrate on policy related C-nodes, and thus the partial tuple would be: $\{\boldsymbol{S}_n, FM_m, G_k\}$. C-nodes become active when their outputs surpass a threshold.

This C-node is permanently stored and, therefore, new contexts for which it is not relevant (it is not active and thus will not participate in the adjustment of the intensities), but that involve some of its associated knowledge nuggets will not lead to drifts in the association intensities and the consequent interference related forgetting of the relationship.

Summarizing, C-nodes represent memories in LTM of contexts in which relevant events occurred. Thus, in a hypothetical case in which a finite number of world-goal combinations (domain-task combinations) are considered, once the cognitive architecture has identified, through interaction with the world, all of the contexts (world, goal, state space area, policy $\{\boldsymbol{S}_i, FM_j, G_k, \pi_r\}$) that lead to relevant events, it will have C-nodes for every one of them. Therefore, in this extreme case, it would be able to directly choose or activate the appropriate policy (series of actions) in order to obtain a reward, or reproduce the relevant event in any case where this is possible, without resorting to any type of prospection.

## 3.2. *Perception, abstraction and P-nodes*

The cognitive architecture continuously receives information in the form of a stream of new perceptions. These perceptions can be raw, directly coming from physical sensors, or more elaborate re-descriptions of other perceptions. The sensory system of a robot encompasses sensors that gather data from the environment, as well as internal sensors, which provide information about the state of the robot itself. In general, all of the raw perceptions can be processed to produce higher-level perceptual representations, that is, perceptual re-descriptions.[63,64] An example is when a pixel intensity representation of an image is transformed into a blob position representation. For the discussion in this paper it does not matter whether a perception is low level or a

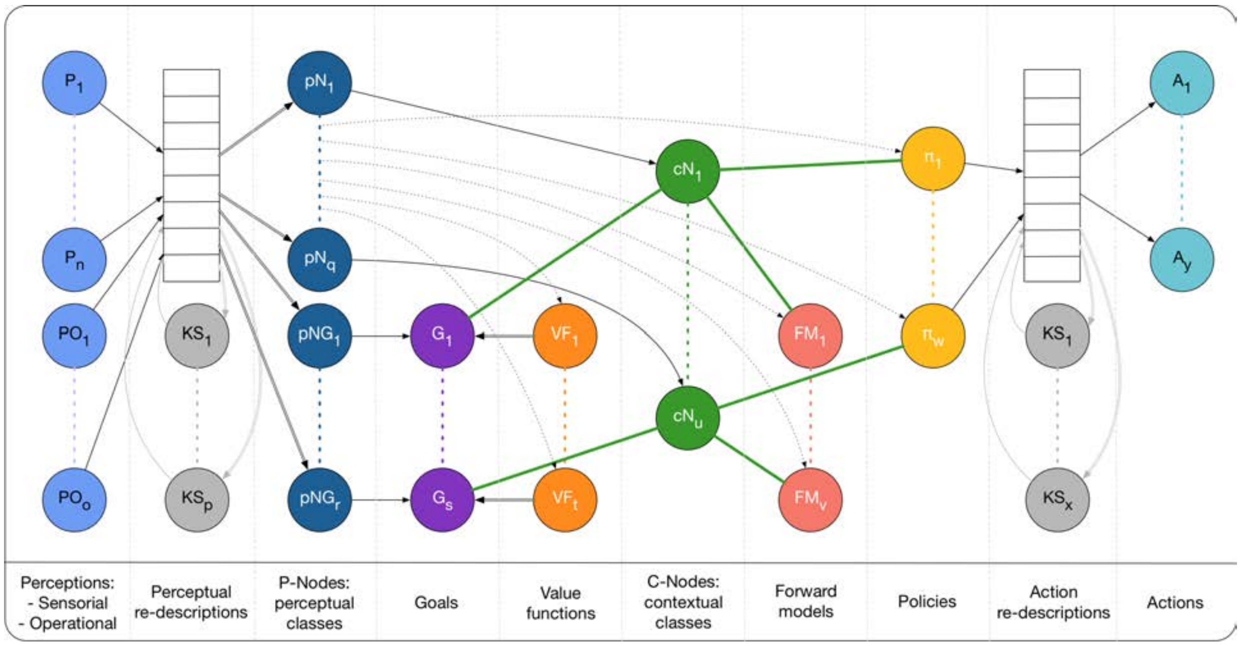


Fig. 2. LTM structure.

higher level re-described representation and, consequently, we will just use the term perception. Nevertheless, in Fig. 2 the basic perceptual layer and the re-description layer are shown for completeness.

What is important, is that a LTM structure should contemplate mechanisms to handle continuous perceptual domains, which is the main case considered in this paper as indicated by the third LTM requirement. This type of domains present many problems. In fact, the concept of C-node does not even make sense in general when considered with respect to individual perceptions (perceptual points) in a continuous domain. It is very improbable that an organism will perceive exactly the same perceptual vector twice. Noise, slight changes in the environment, and other causes will ensure this. Thus, if C-nodes were to be associated to a particular perceptual point, they would be almost useless as they would represent a context that the system will most probably never see again.

It is also necessary to consider that most of the knowledge nuggets (forward models, policies, value functions, C-nodes) that are stored in LTM are not useful in the whole perceptual or state space. For instance, a given forward model is only useful (or reliable) when the robot is operating in a set of perceptual circumstances similar to the ones faced when the forward model was obtained. In other words, all the elements in LTM, including C-nodes, are usually relevant or reliable, that is, they should be activated, only within a particular area of state or perceptual space. Thus, to be able to create a LTM operational structure that can take this into account and at the same time contribute to solving the problem of the inadequacy of individual points in continuous domains as determiners of context, it is necessary to generalize the concept of perceptual state, $\boldsymbol{S}_i$, or perception, to that of perceptual class, $\hat{S}_j$.

A perceptual class, $\hat{S}_j$, can be defined as an area of perceptual space for which all of the points share some common operational trait. This operational trait may be that a given forward model is reliable for these points in the case of forward model related perceptual classes. It may also be, as in the case of C-node related perceptual classes, that for all of the points in the perceptual class the response of the system is the same when faced with the same world and the same goal. Therefore, a perceptual class is an abstraction or generalization of perceptions into a higher level, perhaps discrete, representation that is linked to a given response of the system.

In the particular case of C-nodes, which are the focus of this paper, a perceptual class is a task related abstraction of perception. This concept of abstract representations (whether labeled or not) is quite important in artificial intelligence and developmental robotics as it provides the link from the sensorimotor space, as perceived by the artificial organism, to that of traditional planning and decision systems. It is through successive abstraction and re-description processes that ever more intelligent organisms are postulated.

What is really important here is that these abstractions be constructed in a bottom up fashion directly and autonomously by the artificial organism from the perceptions it acquires when interacting with the environment. This way, all of the abstractions will be grounded on the embodied sensorimotor interactions of the system with its environment. Consequently, these ideas can be used to construct artificial organisms that can autonomously learn the appropriate abstract representations for the contexts they are immersed in.

In this work, perceptual classes will be represented by a LTM component called perceptual nodes or P-nodes. A P-node, denoted as $P_j$, is a functional component that is activated when a perceptual state $\boldsymbol{S}_i$ belongs to a given perceptual class $\hat{S}_j (P_j = 1 \leftrightarrow \boldsymbol{S}_i \,\epsilon\, \hat{S}_j)$ that is,

$$p_j(\boldsymbol{S}_i) = \begin{cases} 1 & \text{if } \boldsymbol{S}_i \,\epsilon\, \hat{S}_j, \\ 0 & \text{otherwise.} \end{cases}$$

Basically, a P-node is a function that acts as a filter over the perceptual state space that allows determining when points belonging to a given perceptual class $\hat{S}_j$ are present.

As a first approximation, we will consider that each C-node will be linked to an associated P-node that determines the area of perceptual space that has to do with the particular policy the C-node is linked to in that specific world and for that concrete goal. In an ideal world with perfect sensing and no ambiguities, once learned, a P-node, $P_j$, will be perfectly active when a perception, $\boldsymbol{S}_n$, belongs to its corresponding perceptual class $\hat{S}_j$.

However, the real world is not ideal and, consequently, a more pragmatic and probabilistic approach is chosen here. According to this approach, a P-node will produce as output an activation level that depends on how confident it is that the current perception vector $\boldsymbol{S}_n$ belongs to the state space area it delimits, that is, to its associated perceptual class $\hat{S}_j$. This confidence function in perceptual space, denoted as $\Gamma_j(\boldsymbol{S}_n)$, is created through the integration of the information from all the points the system has seen and that it has tried to associate to the perceptual class $\hat{S}_j$. In other words, the membership with regards to a perceptual class $\hat{S}_j$ of a state represented by P-node $P_j$ is given by

$$P_j = \Gamma_j(\boldsymbol{S}_n) p_j(\boldsymbol{S}_n),$$

where $p_j$ denotes the activation of the P-node in an ideal world.

In the following sections we provide a more algorithmic description of some basic procedures to create, maintain and use C-nodes and P-nodes so that some cognitive experiments can be carried out.

## 4. An Implementation of the LTM Structure

Figure 2 displays the structure of the LTM addressed in this work. It is made up of instances of the knowledge nuggets associatively linked to each other. This structure evolves in time as the other components of the architecture (Motivational Engine and Prospective learning system) send goals, forward models, value functions, policies and re-description functions to LTM for their storage. It may start with very few components (or even none) and grow as it receives knowledge nuggets from other parts of the architecture or when it creates C-nodes and P-nodes.

The main operational principle of this type of associative LTM is that all of the knowledge nuggets that arrive in LTM can be linked to other components if they co-occur in a relevant situation (as stated before, in the experiments presented here, relevant means rewarded). If the system finds itself in a context or situation it has already experienced or that is very similar to one it has already experienced, when these links are followed and the activations propagated, a final most active policy will result that should be the most adequate for the situation according to the system's experience.

In terms of operation, the activation flow is conceptually asynchronous. Moreover, activations not only result from the propagation of other activations through the associative links of the LTM. They are also a consequence of the activity of other components in the cognitive architecture outside the LTM. For instance, the Motivational Engine can modulate the activation of a series of goal LTM nodes. Also forward models in LTM may become activated due to their success at predicting the current world. These additional activations are the way in which other components of the architecture influence the behavior of the LTM processing structure.

In this paper we are studying the behavior and possibilities of this type of LTM implementation, especially in terms of the behavior induced by C-nodes and P-nodes. Thus, as a first approximation we will assume that goal node activations are externally given by the Motivational Engine according to the goals it wants the system to pursue. We will also assume that the activation of Forward Models will be given by their adequateness to the current world. In other words, in the LTM, forward models are used to identify the world the system is in. Obviously, in the prospective component of an architecture, forward models have another essential function, that of predicting the consequences of actions, but that is not relevant to basic experience based LTM operation.

Taking the previous assumptions into account, a very high level description of the operation of the LTM is given in Fig. 3.

The first element that can be found in the activation flow are P-nodes. P-nodes, as indicated in the previous section, produce a probabilistic output given as an activation value in the range [0:1]. Consequently, every node with an activation value greater than a given threshold will be as active as its value.

```
0.  Read initial perceptions.
1.  while alive or last iteration not reached
2.      Update activation of every node
3.      Select most active policy to be executed
4.      Execute policy
5.      Read new perceptions
6.      Get reward
7.      If (reward < threshold) then
8.          For each activated C-Node connected to the policy
9.              For each activated P-Node connected to the C-Node
10.                 Add previous perception to P-Node as anti-point
11.     else
12.         If (there are C-Nodes connected to the policy) then
13.             For each activated C-Node connected to the policy
14.                 For each active P-Node connected to the C-Node
15.                     Add previous perception as point to P-Node
16.         else
17.             Create a new C-Node and P-Node using perception
18.         end if
19.     end if
20. end while
```

Fig. 3. Algorithmic description of the operation of the LTM.

As commented above, P-nodes are representations of areas of perceptual space, of perceptual classes. Generally, the perceptual space considered by a cognitive architecture can be very high-dimensional and, therefore, a P-node representation must involve the definition of some kind of high-dimensional delimitation within that hyperspace. Any kind of function that achieves the required resolution and precision could conceivably be used to produce this representation.

Two types of points are defined: activating points (or just points) and inhibiting points (or anti-points). Activating points are points the system has experienced in perceptual space where the P-node should have an activation value of 1.0. Anti-points, on the other hand, are experienced points in perceptual space, where the P-node should be inhibited, that is, where the activation should present a value of $-1.0$. In general, points and anti-points are used as the ground truth information in order to adapt whatever representation is being used for the P-nodes.

In the examples considered in this paper, and to make things simple to follow we have resorted to a very simple point based representation. That is, a P-node is represented by a set of characteristic points and some distance rules that delimit the areas around those points for which the system hypothesizes that the P-node should be active. In particular, both, points and anti-points that have resulted from the operation of the P-node are stored in a P-node related structure and represent it. All of the other points within the class will present lower confidence values as a function of their distance to points that were really experienced by the system.

Other types of structures, such as Artificial Neural Networks (ANNs) or even Spiking Neural Networks (SNNs),[65–67] could be used in order to represent the P-node. In this case, learning stages should be inserted (lines 10, 15 and 17 of the algorithm in Fig. 3) during the acquisition of points and anti-points, (either on-line learning as each point comes in or a mixture or on-line and batch learning as the working memory is filled with new points). Most of the discussion that follows is valid, whatever the representation of the P-node.

Given a new perception, $\boldsymbol{S}_i$, the activation of a given P-node represented by a set of points and

```
P_j={p_1,…, p_n, a_1,…,a_m} P-node given by points and anti-points
P_j(x) denotes the activation of P_j for perception x
S_i is a new perception

1.  Calculate k as the closest point/anti-point to S_i out of those
    representing P_j
2.  if (k ∈ {p_1,…, p_n} ) then
3.      P_j (S_i)=1/(||k-S_i||+1)
4.  else if k ∈ {a_1,…, a_n} then
5.      Calculate the centroid of {p_1,…, p_n} as C
6.      if ( ||C-S_i|| < ||C-k|| ) then
7.          Calculate k' as the closest point to S_i out of those
            representing P_j
8.          P_j (S_i)= 1/(||k'-S_i||+1)
9.      else
10.         P_j (S_i)= -1.0
11.     end if
12. end if
13. if (P_j (S_i)< ε) then
14.     P_j (S_i)= 0.0
15. end if
```

Fig. 4. Calculation of the activation of a P-node.

anti-points, $P_j = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\}$, that corresponds to a perceptual class, $\hat{S}_j$, is calculated using the algorithm of Fig. 4. This procedure defines an area in perceptual space for which the P-node should be active. Obviously, if the P-node was represented using an ANN, the membership would be given directly by the output of the trained ANN. Once a P-node produces an activation value, it is propagated to the rest of the nodes in the LTM structure graph that are linked to it.

The next elements in the LTM activation flow are C-nodes. As commented before, a C-Node represents a context in the LTM. For the examples presented in this work, and for the sake of simplicity, we are going to assume Markovian processes and thus we consider that a context is made up of the current perceptual point, $\boldsymbol{S}_i$, the current world or environment, represented by the most activated Forward Model, $FM_j$, and the current Goal of the System, $G_k$. Thus, a C-node is a product type node that links instances of these elements that have co-occurred in rewarded situations to the policy, $\pi_r$, that was applied in order to achieve the reward.

In general, the association defined by a C-node is probably valid for more than a single point in perceptual space, but not for the whole perceptual space. Consequently, C-nodes are not linked to particular states, $\boldsymbol{S}_i$, but rather to perceptual classes, $\hat{S}_j$, represented by P-nodes, $P_l$, that generalize the areas of state space for which the C-node is valid. In this initial implementation each C-Node is connected to just one P-Node, one forward model, one goal and one policy. However, there is no reason to think that a C-node could not be linked to more elements.

The activation of C-nodes is propagated to the policies to which they are connected. Policies are nodes that, for a given perception(s), carry out an action or chain of actions. They are called behaviors in Evolutionary Robotics. Their activation is calculated as the sum of the activation values of the connected C-nodes.

Out of all activated policies, the one with the largest activation is selected for execution. If there is no sufficiently activated policy, which is something that can happen during the first iterations of the system or when it is facing a new environment, another approach that is not based on experience must be chosen in order to perform policy selection. Several approaches could be chosen to achieve this. For instance, when appropriate models of the world and value functions exist, prospection based techniques[68] could be an adequate mechanism. However when there are no appropriate models or value functions and the system needs to explore and learn, intrinsic motivation based techniques may be more adequate.[69–72]

In the experiments of the next section, and in order to dissociate the study of the operation of the LTM from that of the other components of the cognitive architecture, when no policy is sufficiently activated, the active policy will be randomly selected from the set of policies present in LTM. Whatever the method used to select the policy by the cognitive architecture (experience, prospection, random or intrinsic motivation) this policy is then applied by the embodied system in the environment, producing new information in terms of a new perceptual point and sometimes a reward, thus closing the perception-action loop.

Whenever a policy is executed and the results of its application are sensed, a learning phase is initiated. It is in this phase where P-nodes and C-nodes arise and change over time.

At system start-up, depending on how the system is configured, the LTM may have no previous information about past experiences, so there are no P-nodes or C-nodes present. As a result, policies cannot be selected based on experience and some other approach, such as using prospection or following an intrinsic motivation, may be used until a reward is achieved. This phase could be called a babbling phase and to avoid interference from other components of the cognitive architecture, in the examples we will use a random policy selection strategy when in this situation.

As soon as some reward is achieved, the first C-node and P-node are created. Regardless of whether the LTM is empty or not, the process for the creation of C-nodes and P-nodes is always the same. It starts by determining the reward after applying a policy. If the value of this reward is below a given threshold (usually a very small positive number), the algorithm tries to determine what C-nodes, if any, out of those connected to the executed policy were active. If they exist (that is, the policy was not selected for execution through a mechanism other than LTM experience), the activated P-nodes connected to those C-nodes are found, and the point in the input space corresponding to the previous perception is added as an anti-point (value of $-1.0$) to all those P-nodes. The rationale is that all those P-nodes and C-nodes should not have been activated because the robot did not achieve enough reward. On the other hand, when the value of the reward is larger than the threshold, the previous perception is added to those P-nodes as a point (with a value of 1.0).

In the case of relevant reward, if there are no activated C-nodes (policies were chosen through a mechanism different from LTM experience), a new C-node is created linking the executed policy, the forward model with the highest activation value, the goal with the highest activation value, and a new P-node. This new P-node is created using the point corresponding to the previous perception (with a value of 1.0) as its first representation.

By continuous interaction with the environment and using this mechanism, new C-nodes will be created whenever new rewarded contexts arise. These new C-nodes will trigger the creation of new P-nodes, which are initially represented by a single previously experienced perceptual point and an area around it in which the system hypothesizes that this P-node will be valid. Obviously, how the system hypothesizes the valid area depends on the algorithm used to learn and represent the P-node and, in the case of point based algorithms, on the distance rules that define this area.

What is important here, is that, whatever the representation, every time a point of anti-point is determined, the P-node will take it into account in order to configure and delimit this hypothesis area until it represents the complete area of perceptual space that is relevant to that C-node. Thus, in the case of point-wise representations, as the system interacts with the world, if new perceptual points that confirm this hypothesis are explored, they will be added to the list of points that represent the P-node. On the other hand, if these new perceptual points contradict the hypothesis, they will be added to the representation of the P-node as anti-points.

This approach will progressively tend to evolve the LTM graph of the system in terms of connectivity and by adding C-nodes and P-nodes. What is very relevant here is that once a C-node and its corresponding P-node have been created, they will be stored in LTM and will be activated whenever that context (whether complete in the case of C-nodes or just perceptual in the case of P-nodes) is found again, regardless of however many contexts the system may have experienced between two successive activations of a given one. In other words, this approach should allow cognitive architectures to learn multiple world-goal (WG) settings even when they are experienced in an interspersed manner. That is, when they are only partially learnt before the system finds itself in a different WG setting. Through successive returns to any given WG setting, this setting can progressively be learnt in full.

## 5. Results and Discussion

This section aims at showing that an associative LTM with the structure presented in the previous sections can achieve the properties required from a basic LTM. It also seeks to study the behavior C-nodes and P-nodes within this LTM so as to understand how to use them.

### 5.1. *Experiment design*

To this end, the results of a series of experiments that were carried out using a scenario based on a two armed Baxter robot in front of a table (see Fig. 5) are presented. The robot will see on the table either a large object (a cylinder of 7 cm radius and a height of 14.7 cm) or a small object, (a cylinder of 3 cm radius

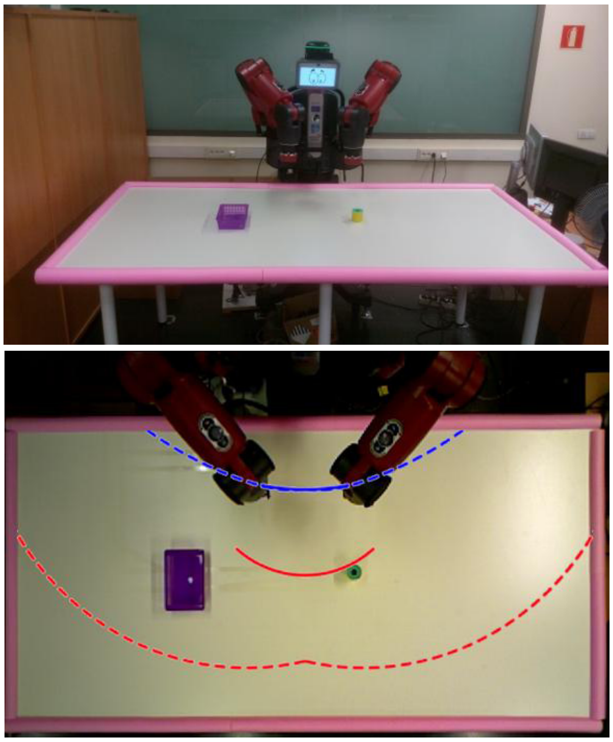


Fig. 5. Scenario (top) and reachable areas (bottom). The dashed lines delimit the reachable area using one gripper, continuous lines delimit the reachable area lifting objects using both arms simultaneously.

and a height of 6 cm) as well as a basket. The large cylinder does not fit in the grippers.

The robot is not able to reach the whole table (the lines in the bottom image of Fig. 5 display the reachable area). Consequently, if it wants to pick up an object that is out of reach it must first ask the experimenter to bring it closer. Also, if it wants to put an object in a basket that is out of reach, it must throw it to the basket.

For actuation, the Baxter has two 7 DOF arms. The sensors used in the experiments are an RGB camera on its head and force sensors associated to all the joint motors. To reduce the dimensions of the perceptual space to something manageable that would allow for plotting P-node representations and discussing them, camera information was re-described in the form of a distance and angle to the center of two blobs as well as their radii. One of the blobs corresponds to the object and the other to the basket. An additional set of re-description functions based on the information from the force sensors is assumed so that a binary representation is provided on the presence of objects in the grippers or held between both arms. Consequently, this LTM works with an eight dimensional perceptual space with six continuous and two discrete dimensions.

The experiments contemplated the use of the LTM isolated from the other components of the cognitive architecture in order to discard any uncontrolled effects due to their mutual interactions. To achieve this isolation, it was first assumed that a given set of basic knowledge nuggets (policies, forward models and goals) from previous interactions with different worlds and under different goals were initially present in LTM. Eight policies (shown in Table 1) were considered. During the tests no new basic knowledge nuggets were added.

The second assumption that was made was that the external activation connections to the forward models (from the prospective component) and to the goals (from the motivational engine) were disconnected and were modulated by hand by the experimenter. This way, the other components of the cognitive architecture had no influence on the behavior of the LTM which was exclusively under the control of the experimenter in terms of determining the task (goal) and domain (world). In fact, and to make things easier to analyze, while contemplating all possible cases, the experimenter was only allowed a choice between two worlds (forward models) and two goals resulting in four WG combinations. Four WG

Table 1. Policies in LTM.

| No. | Policy | Description |
|---|---|---|
| 0 | *Grasp_object* | Use a gripper to grasp an object. |
| 1 | *Ask_nicely* | Ask experimenter to bring something to within reach. |
| 2 | *Change_hands* | Move object from one gripper to the other. |
| 3 | *Put_object_in* | Place an object in a receptacle. |
| 4 | *Sweep_object* | Sweep an object to the central line of the table. |
| 5 | *Throw_object* | Throw an object to a position. |
| 6 | *Hold_with_two_hands* | Use both arms to grasp an object between their ends. |
| 7 | *Put_object_near* | Deposit an object close to the robot base. |

Table 2. All possible C-nodes for the experiment.

| C-node | World (FM) | Goal | Policy |
|---|---|---|---|
| 0 | $W1$: *Grippers low friction* | $G1$: *Object in basket* | *Grasp_object* |
| 1 | | | *Ask_nicely* |
| 2 | | | *Change_hands* |
| 3 | | | *Put_object_in* |
| 4 | | | *Sweep_object* |
| 5 | | | *Hold_with_two_hands* |
| 6 | | | *Throw_object* |
| 7 | | $G2$: *Object near robot* | *Ask_nicely* |
| 8 | | | *Grasp_object* |
| 9 | | | *Put_object_near* |
| 10 | | | *Sweep_object* |
| 11 | | | *Hold_with_two_hands* |
| 12 | $W2$: *No grippers high friction* | $G1$: *Object in basket* | *Ask_nicely* |
| 13 | | | *Sweep_object* |
| 14 | | | *Hold_with_two_hands* |
| 15 | | | *Put_object_in* |
| 16 | | | *Throw_object* |
| 17 | | $G2$: *Object near robot* | *Ask_nicely* |
| 18 | | | *Sweep_object* |
| 19 | | | *Hold_with_two_hands* |
| 20 | | | *Put_object_in* |

combinations is the minimum number that permit all the relevant situations where the world and/or the goal changes.

One of two goals can be activated by the experimenter. The robot may either be required to put an object (whatever its size or position on the table) in the basket ($G1$) or to bring it close to the robot base ($G2$). These goals may be active in two different worlds. In one of them the robot has grippers and can pick up small objects with them ($W1$), and in the other the robot has no grippers and thus needs to use both arms to pick up objects ($W2$). Whatever the world, the robot can only pick up objects with both arms when they are right in front of it. That is, before picking them up it must swipe them towards the central part of the table. When it does this with small objects in the first world, they will tend to overshoot (due to low friction), making it very difficult to set them in that position. This will not happen in the second world, as friction is much higher.

Thus, an experimental setup has been created where the sequence of policies the robot must apply depends heavily on the context. Context here is given by the world the robot is in, its current goal and what it perceives. Thus, depending on the world, the robot must apply different sequences of policies to achieve the same goal. This can be clearly seen in Table 2, which displays all the possible contexts that may arise in the experiment and the C-node they are associated to. Also, it is important to note that the eight dimensional perceptual space is basically continuous (it is continuous in six out of its eight dimensions) and, therefore, for perceptual context to be meaningful it must be stated in terms of perceptual classes. The robot must be able to autonomously delimit these perceptual space areas from its environmental interaction establishing perceptual categorizations it can use to reason with.

No prospection is allowed during these experiments. Therefore, the cognitive architecture is constrained to experience based decisions in order to achieve its objectives.

The different experiments will first address how the proposed LTM can effectively store and reuse configural knowledge. Then, how it reacts when changes in the environment or goals take place. Finally, the dynamics of the P-nodes in time, as well as the efficiency in their representation will be analyzed in terms of how good these representations are at separating and delimiting areas of interest in perceptual space.

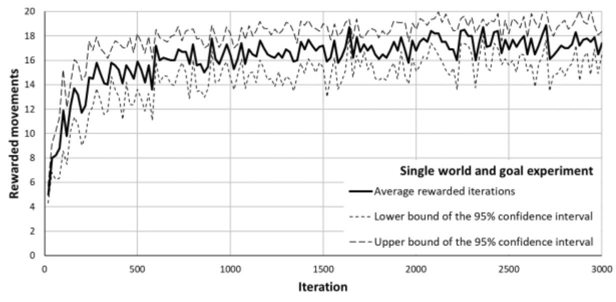
## 5.2. *Single world and goal*

The objective of this first very simple experiment is to determine whether the LTM is able to identify and store all of the relevant contexts in a particular controlled WG combination. To this end, the experiment has been set up with the LTM using a single forward model and a single goal.

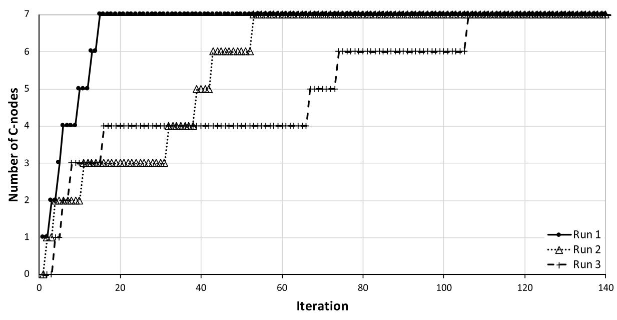
Even though this experiment has been carried out 10 times for each of the WG combinations that were described in the previous subsection, for the sake of brevity and as the results for all cases lead to the same conclusions, only the results corresponding to one WG combination are shown. In particular, the world corresponds to the robot with grippers ($W1$) and the goal is to put the cylinders it sees in the baskets ($G1$). For this WG combination only 7 relevant contexts need to be remembered to fully control the world (see Table 2).

Figure 6(a) displays the results for 10 runs of the experiment of the experiment in terms of the number of rewards obtained including the lower and upper bounds of the 95% confidence interval. Figure 6(b) displays number of C-nodes that were created in three different runes. All of the runs, except run 1, were provided with different contexts in a random manner. In the case of run 1 they were provided sequentially. Figure 7 displays an image of the final LTM that was achieved.

For the sake of clarity, nodes (circles) are grouped by type using boxes, and activation flows from left to right. The darker the color, the larger the activation of that node. For instance, in Fig. 7 there are several activated C-nodes and, therefore, several activated policies, being the most activated one "hold with 2 hands", which is the one executed at the end of that iteration. If we focus on perceptions, it can be seen that the object size perception is very dark (the object is large) and that the object angle perception is white (that means that it is in front of the robot at an angle of 0°), therefore, holding it with 2 hands is the right policy to be executed. As this experiment involves only one forward model and one goal, all the C-nodes are connected to them.



(a)



(b)

Fig. 6. (a) Average number of rewards obtained every 20 iterations by 10 runs of the single world and goal experiment. (b) Detail of the creation of the 7 C-nodes required for this case for three of the runs.

As shown in Fig. 6(b) as well as in Fig. 7, the system creates all of the C-nodes that are necessary. In fact, the speed at which it acquires these C-nodes is given by how long it takes for the different contexts to arise at least once during the experiment. As shown for Run 1 in Fig. 6(b), if these different contexts are forced to appear in a sequence, it only takes the system 15 interactions with the world to create the corresponding C-nodes.

Obviously, this does not imply that the system is proficient in this environment, as shown by the low success rate of its interactions when it has just learnt the C-nodes (iterations 1–120 in Fig. 6(a)). This is due to the fact that the P-nodes corresponding to these C-nodes have experienced very few representative points (only one each, and one or two anti-points in some cases, after the 15th iteration of Run 1 in Fig. 6(b)) and, consequently, they represent the perceptual classes very poorly. The evolution of the delimitation of perceptual classes and its relationship to the proficiency of the LTM will be discussed in Sec. 5.4.

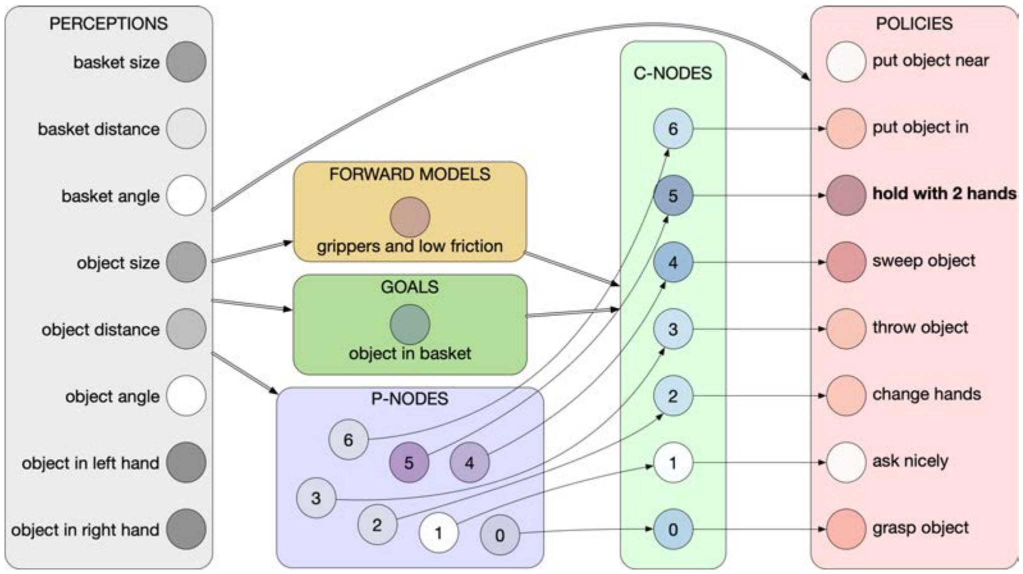


Fig. 7. Snapshot of the operation of the LTM, corresponding to iteration 3000, of an execution of the experiment with one world and one goal.

## 5.3. *Multiple world-goal combinations*

As shown before, for a given WG combination it is straightforward for the proposed LTM structure to produce all of the C-nodes it needs. Now the problem of changing WG combinations will be considered, as it is quite important to determine whether and how much interference occurs among different WG combinations in the long-term storage of information.

To ascertain the effects of these changes on C-nodes, a series of experiments were run in which WG changes occurred at different time intervals. Figure 8 displays, in its left column, the results of these runs in terms of the average of 10 runs as well as the lower and upper bounds of the 95% confidence interval. The right column provides a power model regression curve for the probability of rewarded movement in each case. Figure 9 displays the results for two particular runs including the C-node/P-node acquisition rate.

These figures show how, for this architecture, there are actually no interference effects. In Figs. 8 and 9 (bottom), the intervals between WG changes were so long that the system faced all of the possible contexts within a WG combination and was able to produce all of the C-nodes corresponding to each WG situation before changing into the next one. In fact, in Fig. 8 bottom it can be appreciated how during the first 800 iterations, every 200 iterations, as the system changes WG, there is a large decrease in performance, until it learns the new WG setting. This is, not the case in the case of a period of 25 iterations. Here, the system is gradually learning all the WG settings in an interspersed manner and the evolution of performance is smoother. That is, whenever the system returns to a given WG, it reuses previously created C-nodes when they are deemed relevant for the context. At the same time, it keeps on correctly detecting new contexts as they appear and creating their corresponding C-nodes, until all of them have been experienced and created (Fig. 9). Thus, a conclusion that is obtained is that due to the way C-nodes are preserved in LTM, the creation of new C-nodes corresponding to new WG combinations does not affect others present in LTM. This process is independent from the WG switch interval, confirming the nonexistence of interference effects in a C-node based configural associative LTM. Even though this could seem quite obvious due to the design of the C-node structure, it needed to be ascertained in real settings to be completely rigorous.

Figure 10 displays two steps during the growth of the LTM in one of the runs of this experiment. All of the C-nodes as well as their corresponding P-nodes are progressively created and appropriately linked to their corresponding contexts up to a total of 21, which is the theoretical number of contexts in this experiment as shown in Table 2.

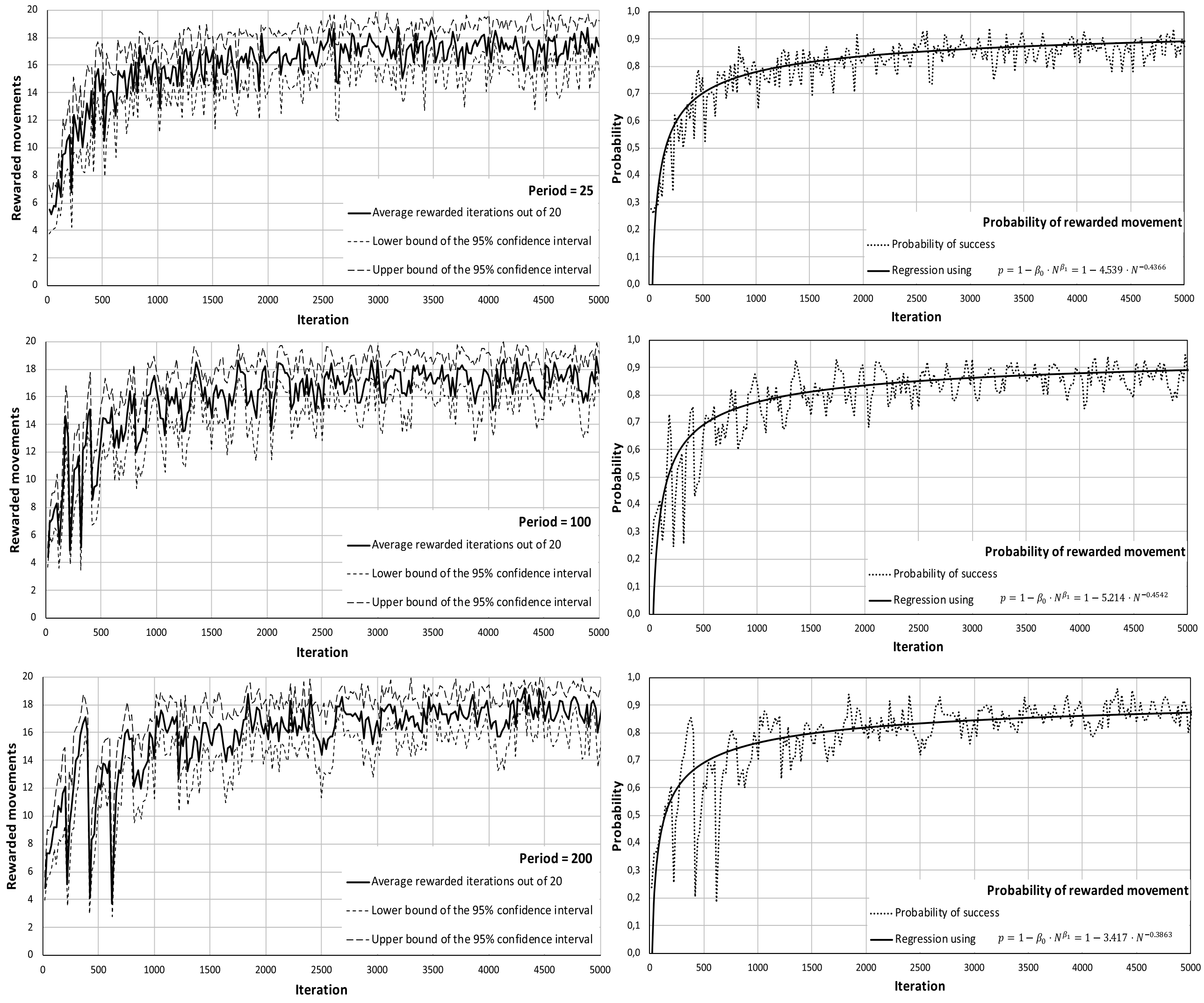


Fig. 8. Results for three world-goal switch periods in the multiple world-goal experiment with (from top) 25, 100 and 200 iterations between changes of world-goal combination (the sequence was always a cyclic $W1G1, W1G2, W2G1, W2G2$). The left column shows the results for ten runs of the experiments in each case. The right column displays the probability of rewarded movement.

Again, the fact that the robot may have detected all the relevant contexts and created the pertinent C-nodes and their associated P-nodes does not mean that it is able to operate proficiently in the WG contexts it faces. Initially, it is representing perceptual classes through P-nodes, without experiencing many different perceptual situations in that WG combination, the representation of the perceptual class provided by the P-node is poor. The hypothesis of the P-node on what area of state space corresponds to the class is very sketchy. This is evident in Figs. 8 and 9 which display the number of times the system reached the goal every 20 interactions with the world. It is only after around 1800 iterations, when it has experienced a relevant number of different cases corresponding to a given perceptual class, that the success level starts to improve, reaching a situation where it mostly achieves the goals between 18 and 20 times out of 20. It never reaches a perfect sustained 20 out of 20, but we will comment on this later, after studying the evolution of the P-nodes.

## 5.4. *P-node behavior*

P-node dynamics are a key factor in understanding how this type of LTM structure operates. It

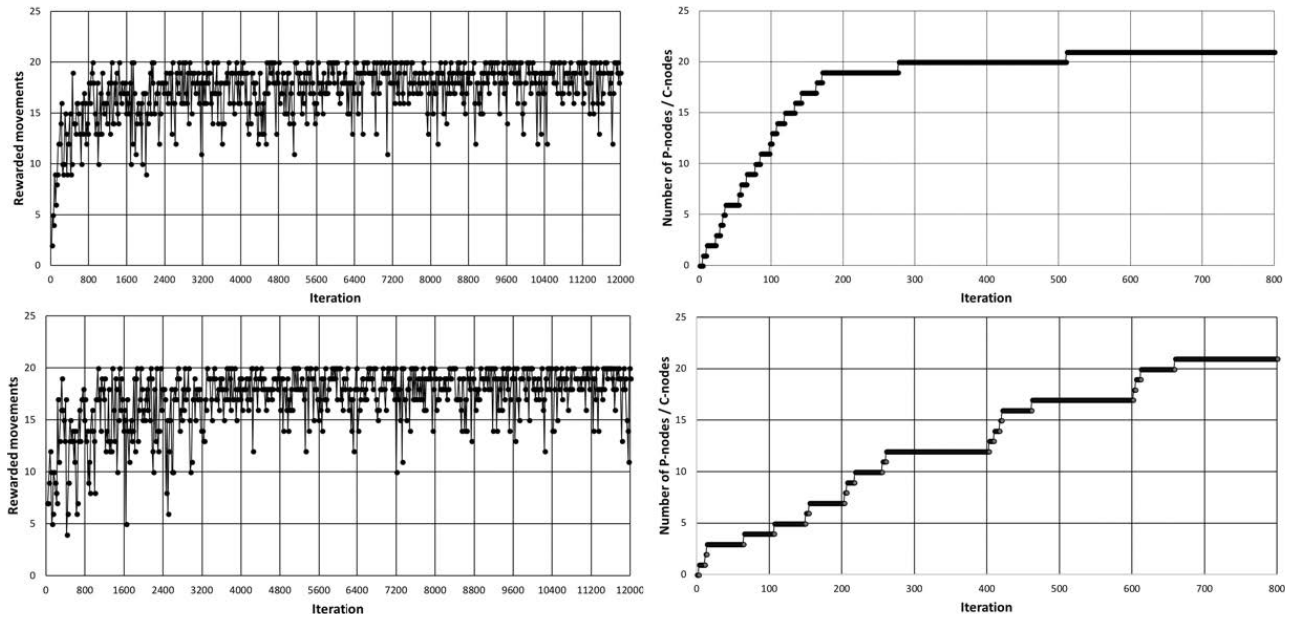


Fig. 9. Results for two particular runs of the multiple world-goal corresponding (from top) to 25 and 200 iterations between changes of world-goal combination.

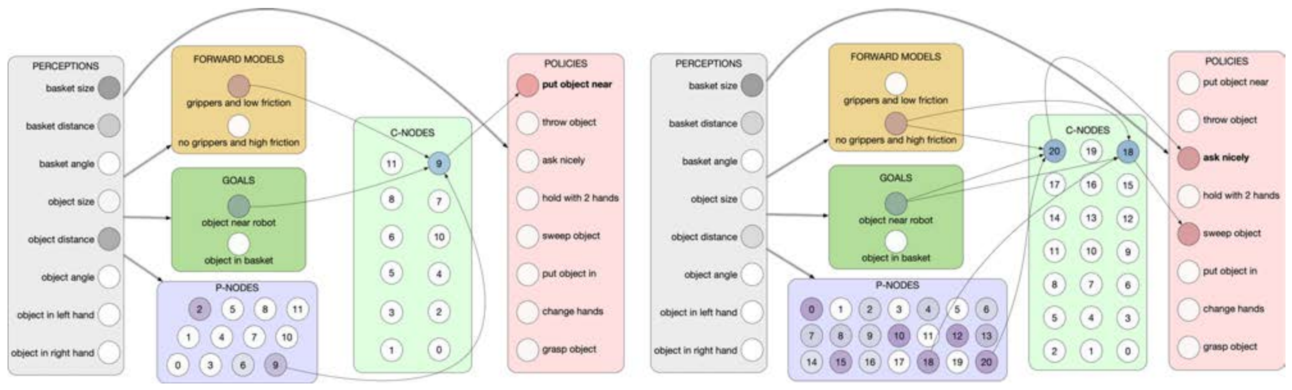


Fig. 10. Snapshots of the LTM for iterations 400 and 12,000 of an execution of the multiple WG experiment starting from an LTM with no C-nodes or P-nodes. For the sake of clarity, only connections for C-nodes that are active in that instant are depicted.

has been shown how C-nodes are progressively created and updated as reward is obtained, how this is not affected at all by context changes, and how retrieval is instantaneous and effective. Now, the objective is to analyze in detail how P-nodes are created and updated and provide some examples of their behavior.

As indicated above, in the first implementation of this LTM structure, the creation of a C-node entails the creation of an associated P-node. This P-node is initially represented by a single point corresponding to the perception that produced a reward when the C-node was created. Thus, the P-node is activated with a value of 1.0 for its only representative point and with lower values in the hypothesis area surrounding the point depending on the representation used. The hypothesis area is given by the P-node activation algorithm, but it is just a hypothesis and, consequently, part of the area it encompasses may actually not be a part of the perceptual class. In this case, eventually, the P-node will be activated by a point that is in its hypothesis area but that does not belong to the particular perceptual class it represents. This will most probably cause the activation of a policy that will not lead to reward. As a consequence, this perceptual point will be classified as an anti-point and added to the P-node as such, changing the shape of its hypothesis area. Now, the P-node will not be activated whenever that particular perceptual point as well as its surrounding points are experienced again as the hypothesis area around the anti-point will also be inhibited.

As time passes, and more interactions with the world involving the particular P-node take place, it will be informed by more points (corresponding to perceptions that activated the P-node and that produced a reward) and anti-points (corresponding to perceptions that activated the P-node and did not produce a reward). This will lead to more defined hypothesis areas for the P-node and thus a better correspondence between P-node output and membership to the perceptual class it represents. Depending on the rules that are implemented for the P-node with respect to the distance to known points in the case of point-wise representations or the training algorithm in other representations, such as ANNs, its operation may be quite different.

Here, a highly exploratory strategy has been chosen for the rules. In it, slope of the probability around a given point in the P-node is very small. This way, the initial point that generates the P-node basically assumes as an initial hypothesis that the whole perceptual space belongs to the perceptual class. It is only by making mistakes, that the system incorporates anti-points and these, when paired with points, delimit the real borders of the class.

An important side effect of this choice of approach is that whenever there is no information on an area of perceptual space, the hypothesis the system makes on the membership of a new point to a class depends on whether the new point is closer (no matter how far it is from it) to a point or to an anti-point. In the first case it will assume positive membership and in the second one negative membership.

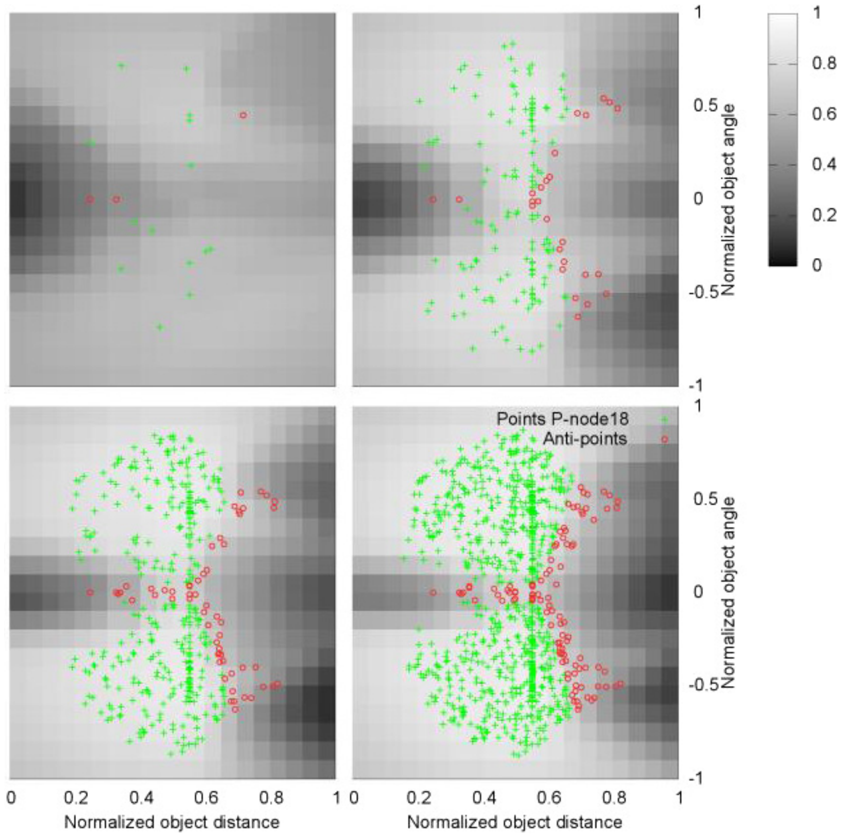


Fig. 11. P-node 18 activation map for iterations 700, 3000, 6000 and 12,000 in an experiment with 12,000 iterations (crosses are points, circles are anti-points, and the lighter the grey, the larger the activation). This P-node was created at iteration 605.

In other words, depending on the side of a border the new point is in, and independently of how unknown the area where this new point is located is, a clear hypothesis is made (even if it is wrong).

Figure 11 displays the evolution in time of a P-node hypothesis area (P-node 18), including the representative points the P-node has acquired in order to construct it (points denoted by crosses and anti-points by circles). As in this example we are working with an eight dimensional perceptual space, the graphs show the activation map (the lighter the color, the larger the activation) over two dimensional sections. In fact, we have chosen the two most significant continuous dimensions (sensors) in each case. By "more significant sensors" we mean those sensors that play the most relevant role when calculating the activation. For instance, in the case of gripping an object, the most significant sensors are those corresponding to distance and angle to the object. This representation provides an idea with respect to the areas of state space in which the P-node will be activated (lighter areas) and in which it will be inhibited (darker areas) and it can thus be taken as a map of the perceptual class (lighter areas correspond to perceptions that are included in the perceptual class).

P-node 18 corresponds to the execution of the *Sweep_object* policy in the world where the robot has no grippers and when the goal is to bring the object as close as possible to the robot. This P-node is very easy to interpret. As long as the object is reachable, the robot can sweep the object to the area where it can pick it up with both arms (the central area), so the reachable area is full of points except in the area where it can pick the object up, because the object is already there. The anti-points are delimiting the out of reach area, and are located also in the area where the objects can be picked up with both arms.

It is clear from the time evolution of the images in the figure that, initially, when the robot has not interacted with the environment very much and there are few points and anti-points, the delimitation of the perceptual class is quite poor. However, as more points and anti-points are acquired, the delimitation obtained by the robot is much more detailed and, consequently, the decisions that can be made based on it will be much more precise.

In Fig. 12, a similar graph to that of Fig. 11 is shown for some P-nodes at the end of the run. Table 2 displays the corresponding context for each P-node, as there is a one-to-one relationship in this experiment between a given P-node and the C-node with the same number. The concentration of points is always larger than the concentration of anti-points and this is due to how P-nodes are updated. Whenever, the execution of a policy produces a reward, the perception becomes a point of the corresponding activated P-node. However, an anti-point is only added if the policy does not produce reward and the P-node was activated. If the P-node was not activated, the anti-point is not created. Thus, the algorithm reinforces only the activation areas with points. As time passes this asymmetry leads to the anti-points delimiting a frontier separating areas where the P-node should and should not be activated.

In addition to this, there is an artifact in some of the graphs that deserves an explanation. In the graph corresponding to P-node 0, and in some of the others, such as P-node 1 (anti-points in this case), which is basically its complementary node, there is something that may look un-random or un-statistical and that is that there is an important concentration of points for a particular value of object distance (a line can be discerned on the graph). This is because the value for the distance from the robot at which the experimenter sets the object after the robot applies the *Ask_nicely* policy is always the same (although the angle is not necessarily same).

An interesting graph is the one corresponding to P-node 2, associated with the *Change_hands* policy when the goal is to put the object in the box and the robot has grippers. The most significant sensors in this case are the angle of the object and the angle of the basket. To be able to put an object in the basket, the robot first needs to have this object in the gripper that is on the same side as the basket. Consequently, this policy should be activated whenever this is not the case and inhibited when it is. This is what the perceptual class shows, when object is in the opposite side to that of the basket the robot must move it from its current gripper to the other one.

It is also easy to see that P-node 0 and P-node 1 are complementary. The first one corresponds to the *Grasp_object* policy, which needs to define an area in which the object is reachable. The other one is related to the *Ask_nicely* policy, which requires a definition of the areas in which the object cannot be reached.

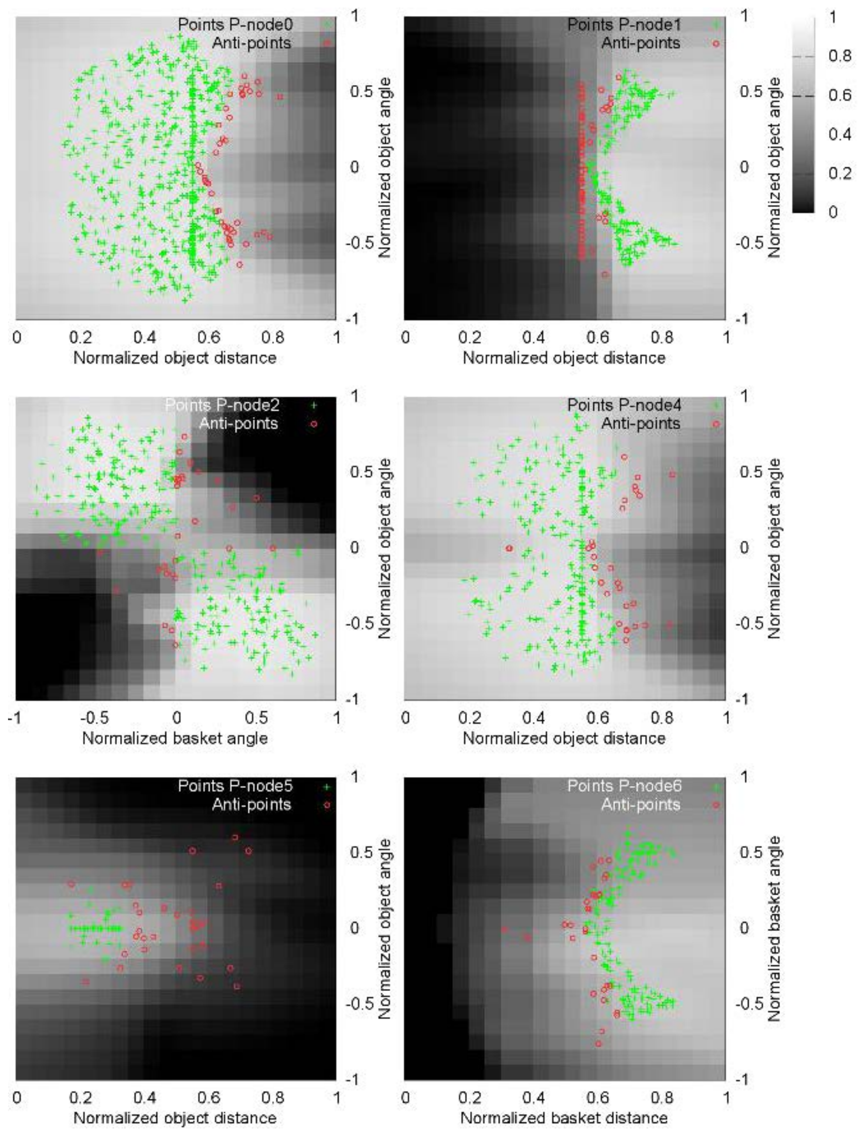


Fig. 12. Final activation maps for several P-nodes in an experiment with 12,000 iterations.

On the other hand, the case of P-node 5 is a clear example of the grounding of the P-nodes. This P-node corresponds to the *Hold_with_two_hands* policy for the same goal and in the same world as P-node 0. To be able to grasp the object with its two hands, the object must be right in front of the robot and, due to the structure of its arms and the position of the table, there is a very small window of distances from the robot and angles with respect to the central line of the table where, it is feasible to do it. This is what the P-node is showing by defining the activation of the perceptual class in the distance up to around 0.38 and the angle interval of a maximum of between around $-0.25$ to $+0.25$ in the widest point.

The main conclusion that can be derived from these figures is that the approach presented here is able to identify and clearly define the P-nodes it requires in order to segment the perceptual space as a function of the context (domain and task) that must be addressed. It creates well established borders delimiting where the P-node should be activated and, if one looks closely at the resulting distributions, they actually make a lot of sense in terms of perceptual classes from a human point of view. Resorting to the cases analyzed in the previous paragraphs, P-node 0 would correspond to an "objects within reach" perceptual class, P-node 1, to an "objects out of reach" perceptual class, P-node 2 to an "object in the wrong hand" perceptual class, and so on.

Obviously, the LTM does not label the classes (at least not yet), but that does not mean that they do not present semantic content. These classes are each represented by its P-node and, therefore one could think that these semantics could be attributed to the P-nodes representing the classes and that higher level planning and reasoning processes could be carried out based on this segmented context dependent representation of the perceptual space of the robot. However, this is the topic of another paper.

Now that we have studied how P-nodes behave, and how they progressively delimit the perceptual classes for the different contexts, we can go back and observe the figures corresponding to the interaction of the system with the world in terms of goal achievements in time (Figs. 6 and 8).

As commented previously, it is easy to see that a relatively fast initial improvement stage takes place in which the system finds the required contexts, encodes them into C-nodes and incorporates the first points into P-nodes representing perceptual classes. P-nodes allow it to make hypothesis on whether new points belong or not to the perceptual classes they represent. Obviously, as we have seen in Fig. 11, these P-nodes initially make hypotheses that are too general and slowly delimit the perceptual class profile through the exploration of points and anti-points in their borders.

These initial delimitations of classes, even though they are far from perfect, allow a very fast increase in the performance of the system. However, this increase is full of valleys, as shown in Fig. 9, which occur when a point is misclassified leading to a mistake in the choice of policy. This problem is even worse when the misclassification leads to no P-node being activated, as the system will start to randomly test policies until one is found that produces a reward. This babbling may lead to a whole set of unrewarded interactions, thus decreasing the perceived performance of the system.

Depending on the dimensionality and complexity of the perceptual space the LTM is operating in, the delimitation of perceptual classes may be easier or harder. For instance, some perceptual classes are characterized along an independent binary sensorial dimension (ball in gripper or not in gripper). These classes are very easy to delimit with a very small number of samples. However, other classes, like those dealing with higher continuous dimensionalities, where there are dependencies among the different dimensions, as in the case of reaching to grasp an object (P-node 0, for instance), are much more difficult to learn. In fact, they usually require a very large number of perceptual instances, trials of the system in the world, especially in the class extensive and often tortuous borders, so that they can delimit these borders with an appropriate resolution. Thus, there is room for improvement in the algorithms that characterize P-nodes so that more precise delimitations can be achieved with fewer trials.

Notwithstanding this, the performance of the system is quite robust. It is capable of identifying regularities in its interaction with environments under different motivations to establish associative relationships among the different knowledge nuggets it has acquired reflecting and remembering contexts in configural representations. This allows it to operate in different contexts without interference effects. It is also able to aggregate events into context dependent categories, which in the case of perception means establishing and delimiting the appropriate task related perceptual classes. In other words, it fulfils the requirements that were established at the beginning.

## 6. Conclusions

This paper proposes a general structure for a LTM within a cognitive architecture in open-ended learning settings based on knowledge nuggets represented as ANNs. It is inspired on the Memory Network concept proposed by Fuster.[58] The basic architectural structure is enhanced by the addition of two new classes of knowledge nuggets: C-nodes, which allow for the storage of configural associations in a very straightforward manner, and P-nodes, which represent perceptual classes.

A mechanism is proposed for the general operation of this type of LTM in terms of experience based decision processes. This mechanism is agnostic to the algorithms used for the delimitation of P-nodes. However, in order to provide a complete description and analysis of the operation of the system, we have proposed a very simple point-based algorithm for the autonomous delimitation of perceptual classes by the P-nodes as the system interacts with the world. P-nodes, together with the determination of relevant contextual relationships as provided by C-nodes, allow for the creation of a LTM capable of learning and performing experience based or "automatic" decision processes that can help the system to fluidly interact with the world.

One very important characteristic of this type of memory that is basic for open-ended learning, is the lack of memory interference effects with regards to context, thus allowing for the system to be able to learn to operate in different worlds under different goals when it experiences them in an interspersed manner, much like living beings do. A second very important characteristic is that the capability of autonomously generating perceptual classes associated to different worlds and tasks permits segmenting the world into semantically loaded categories, associated to contexts, that can later be used in higher level reasoning and planning processes.

In this paper we have concentrated mostly on the perceptual and context related aspects of LTM, taking the executive component as given. However, the executive aspects are also quite important in terms of achieving cognition. In this line, policies may be parametrized and, consequently, may also define a continuous parameter space that would require of its autonomous generalization into executive classes (E-nodes). Additionally, both P-nodes and E-nodes may be organized into perceptual and executive hierarchies, parts of which can be reused for new contexts. That is, they are compositional knowledge nuggets from which we may take advantage. Finally, as indicated in the paper, different representations of P-nodes and E-nodes could lead to more precise delimitations of these areas and, thus, better performance of the systems that use this approach. In fact, for the sake of homogeneity, as the rest of the knowledge nuggets in the architecture are represented through ANNs, an ANN based representation of these new structures would make a lot of sense. These are all problems we are working on now and that provide good avenues of future research.

## Acknowledgments

This work has been partially funded by the EU's H2020 research program Under Grant No. 640891 (DREAM) as well as by the Xunta de Galicia and the European Regional Development Funds Under Grants ED431C 2017/12 and redTEIC network (ED341D R2016/012).

## References

1. S. J. Shettleworth, *Cognition, Evolution and Behavior* (Oxford University Press, 1998).
2. M. Roveri and F. Trovo, An ensemble approach for cognitive fault detection and isolation in sensor networks, *Int. J. Neural Syst.* **27** (2017) 1650047.
3. R. Sun, The importance of cognitive architectures: An analysis based on CLARION, *J. Exp. Theor. Artif. Intell.* **19** (2007) 159–193.
4. D. Vernon, G. Metta and G. Sandini, The iCub cognitive architecture: Interactive development in a humanoid robot, *2007 IEEE 6th Int. Conf. Dev. Learn.* **4** (2007) 122–127.
5. D. Vernon, *Artificial Cognitive Systems* (MIT Press, 2014).
6. M. Saignavongs *et al.*, Neural activity elicited by a cognitive task can be detected in single-trials with simultaneous intracerebral EEG-fMRI recordings, *Int. J. Neural Syst.* **27** (2017) 1750001.
7. J. M. Fuster and S. L. Bressler, Past makes future: Role of pFC in prediction, *J. Cogn. Neurosci.* **27** (2015) 639–654.
8. D. Vernon, M. Beetz and G. Sandini, Prospection in cognition: The case for joint episodic-procedural memory in cognitive robotics, *Front. Robot. AI* **2** (2015) 1–14.
9. R. M. Shiffrin and W. Schneider, Controlled and automatic human information processing: II. Perceptual learning, automatic attending and a general theory, *Physiol. Rev.* **84** (1977) 127.
10. J. R. Anderson, Production systems and the ACT-R theory, in *Rules of Mind* (Lawrence Earlbaum Associates, 1993), pp. 1–16.
11. M. Meyer and D. Kieras, A computational theory of executive control processes and human multiple-task performance. Part 1. Basic mechanisms, *Psychol. Rev.* **104** (1997) 3–65.
12. S. C. Saphiro and H. O. Ismail, Anchoring in a grounded layered architecture with integrated reasoning, *Rob. Auton. Syst.* **43** (2003) 97–108.
13. M. A. Just and S. Varma, The organization of thinking: What functional brain imaging reveals about the neuroarchitecture of complex cognition, *Cogn. Affect. Behav. Neurosci.* **7** (2007) 153–191.
14. J. E. Laird, A. Newell and P. S. Rosenbloom, SOAR: An architecture for general intelligence, *Artif. Intell.* **33** (1987) 1–64.
15. B. Goertzel, OpenCogPrime: A cognitive synergy based architecture for artificial general intelligence, in *8th IEEE Int. Conf. Cognitive Informatics* (Hong Kong, 2009), pp. 60–68.

16. J. A. Starzyk, S. Member and J. Graham, MLECOG: Motivated learning embodied cognitive architecture, *IEEE Syst. J.* **11**(3) (2015) 1–12.
17. J. R. Anderson *et al.*, An integrated theory of the mind, *Psychol. Rev.* **111** (2004) 1036–1060.
18. I. Kotseruba and J. K. Tsotsos, A review of 40 years of cognitive architecture research: Core cognitive abilities and practical applications, arXiv: 1610.08602.
19. F. Dawood and C. K. Loo, Developmental approach for behavior learning using primitive motion skills, *Int. J. Neural Syst.* **28** (2018) 1750038.
20. J. Wang, L. Shang, Y. Chen and Z. Yi, A new delay connection for long short-term memory networks, *Int. J. Neural Syst.* **28** (2018) 1750061.
21. M. Koziarski and B. Cyganek, Image recognition with deep neural networks in presence of noise — Dealing with and taking advantage of distortions, *Integr. Comput. Aided. Eng.* **24** (2017) 337–350.
22. A. Antoniades *et al.*, Deep neural architectures for mapping scalp to intracranial EEG, *Int. J. Neural Syst.* **28** (2018) 1850009.
23. R. Wood, P. Baxter and T. Belpaeme, A review of long-term memory in natural and synthetic systems, *Adapt. Behav.* **20** (2012) 81–103.
24. P. Gärdenfors, Conceptual spaces as a framework for knowledge representation, *Mind Matter* **2** (2004) 9–27.
25. J. M. Fuster, Cortex and memory: Emergence of a new paradigm, *J. Cogn. Neurosci.* **21** (2009) 2047–2072.
26. R. C. Atkinson and R. M. Shiffrin, Human memory: A proposed system and its control processes, *Psychol. Learn. Motiv. — Adv. Res. Theory* **2** (1968) 89–195.
27. N. C. Waugh and D. A. Norman, Primary memory, *Psychol. Rev.* **72** (1965) 89.
28. A. Baddeley and G. Hitch, Working memory, in *The Psychology of Learning and Motivation Advances in Research and Theory* (Academic Press, 1974), pp. 47–89.
29. R. W. Engle, S. W. Tuholski, J. E. Laughlin and A. R. A. Conway, Working memory, short-term memory, and general fluid intelligence: A latent variable approach, *J. Exp. Psychol. Gen.* **128** (1999) 309.
30. G. A. Miller, The magical number seven, *Psychol. Rev.* **63** (1956) 81.
31. L. Peterson and M. J. Peterson, Short-term retention of individual verbal items, *J. Exp. Psychol.* **58** (1959) 193.
32. A. Baddeley, Working memory, *Philos. Trans. R. Soc. B Biol. Sci.* **302** (1983) 311–324.
33. K. A. Ericsson and W. Kintsch, Long term working memory, *Psychol. Rev.* **102** (1995) 211.
34. W. Schneider and R. M. Shiffrin, Controlled and automatic human information processing: I. Detection, search, and attention, *Psychol. Rev.* **84** (1977) 1.
35. K. Kotovsky, J. R. Hayes and H. A. Simon, Why are some problems hard? Evidence from Tower of Hanoi, *Cogn. Psychol.* **17** (1985) 248–294.
36. A. De Groot, *Thought and Choice in Chess* (Mouton, 1965).
37. W. G. Chase and H. A. Simon, The mind's eye in chess, in *Visual Information Processing* (Elsevier, 1973), pp. 215–281.
38. K. Oberauer, Access to information in working memory: Exploring the focus of attention, *J. Exp. Psychol. Learn. Mem. Cogn.* **28** (2002) 411–421.
39. E. L. Thorndike, *Animal Intelligence*: *Experimental Studies* (Macmillan, 1911).
40. I. P. Pavlov, *Conditional Reflexes*: *An Investigation of the Physiological Activity of the Cerebral Cortex* (H. Milford, 1927).
41. I. P. Pavlov, *Lectures on Conditioned Reflexes* (International, 1928).
42. R. A. Rescorla and A. R. Wagner, A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement, *Class. Cond. II Curr. Res. Theory* **2** (1972) 64–99.
43. N. J. Mackintosh, A theory of attention: Variations in the associability of stimuli with reinforcement, *Psychol. Rev.* **82** (1975) 276.
44. J. M. Pearce and G. Hall, A model for Pavlovian learning: Variations in the effectiveness of conditioned but not of unconditioned stimuli, *Psychol. Rev.* **87** (1980) 532.
45. J. M. Pearce, A model for stimulus generalization in Pavlovian conditioning, *Psychol. Rev.* **94** (1987) 61.
46. J. M. Pearce, Similarity and discrimination: A selective review and a connectionist model, *Psychol. Rev.* **101** (1994) 587.
47. A. R. Wagner, SOP: A model of automatic memory processing in animal behavior, in *Information Processing in Animals* (Psychology Press, 2014), pp. 15–58.
48. A. R. Wagner and S. E. Brandon, Evolution of a structured connectionist model of Pavlovian conditioning (AESOP), in *Contemporary Learning Theories*: *Pavlovian Conditioning and the Status of Traditional Learning Theory* (L. Erlbaum Associates, 1989), pp. 148–189.
49. C. L. Hull, *Principles of Behavior* (Appleton-Century-Crofts, 1943).
50. C. L. Hull, *A Behavior System* (Yale University Press, 1952).
51. A. R. Wagner and R. A. Rescorla, Inhibition in Pavlovian conditioning: Application of a theory. in *Inhibition and Learning* (Erlbaum, 1972), pp. 301–334.
52. E. J. Kehoe, A layered network model of associative learning: Learning to learn and configuration, *Psychol. Rev.* **95** (1988) 411.
53. E. A. Wasserman and R. R. Miller, What's elementary about associative learning, *An. Rev. Psychol.* **48** (1997) 573–607.

54. H. Gulliksen and D. L. Wolfle, A theory of learning and transfer, *Psychometrika* **3**(3) (1938) 127–149.
55. W. P. Bellingham, K. Gillette-Bellingham and E. J. Kehoe, Summation and configuration in patterning schedules with the rat and rabbit, *Anim. Learn. Behav.* **13** (1985) 152–164.
56. J. M. Pearce, Evaluation and development of a connectionist theory of configural learning, *Anim. Learn. Behav.* **30** (2002) 73–95.
57. V. Mante, D. Sussillo, K. V. Shenoy and W. T. Newsome, Context-dependent computation by recurrent dynamics in prefrontal cortex, *Nature* **503** (2013) 78.
58. J. M. Fuster, Network memory, *Trends Neurosci.* **20** (1997) 451–459.
59. J. M. Fuster, Cortical dynamics of memory, *Int. J. Psychophysiol.* **35** (2000) 155–164.
60. J. R. Anderson and B. J. Reiser, Production systems and the ACT-R theory, *Mind Readings Introd. Sel. Cogn. Sci.* (MIT Press, 1998) 59–76.
61. R. Sun, E. Merril and T. Peterson, From implicit skills to explicit knowledge: A bottom-up model of skill learning, *Cogn. Sci.* **25** (2001) 203–244.
62. R. J. Duro, J. A. Becerra, J. Monroy and P. Caamano, Considering memory networks in the LTM structure of the multilevel darwinist brain, in *GECCO 2016 Companion — Proc. 2016 Genetic and Evolutionary Computation Conf.* (ACM, 2016), pp. 1057–1060.
63. A. Karmiloff-Smith, *Beyond Modularity*: *A Developmental Perspective on Cognitive Science* (MIT Press, 1995).
64. A. Carassa and M. Tirassa, Representational redescription and cognitive architectures, *Behavioral and Brain Sciences* **17** (1994) 711–712.
65. E. M. Izhikevich, Simple model of spiking neurons, *IEEE Trans. Neural Networks* **14**(6) (2003) 1569–1572.
66. L. Pan, G. Paun, G. Zhang and F. Neri, Spiking neural P systems with communication on request, *Int. J. Neural Syst.* **27** (2017) 1750042.
67. T. Wu, F. D. Bilbie, D. Paun, L. Pan and F. Neri, *Int. J. Neural Syst.* **28** (2018) 1850013.
68. F. Bellas, R. J. Duro, A. Faina and D. Souto, Multilevel Darwinist Brain (MDB). Artificial evolution in a cognitive architecture for real robots, *IEEE Trans. Auton. Ment. Dev.* **2** (2010) 340–354.
69. P. Y. Oudeyer and F. Kaplan, What is intrinsic motivation? A typology of computational approaches, *Front. Neurorobot.* **1** (2009) 6.
70. R. Salgado, A. Prieto, F. Bellas, L. Calvo-Varela and R. J. Duro, Motivational engine with autonomous sub-goal identification for the Multilevel Darwinist Brain, *Biol. Inspired Cogn. Archit.* **17** (2016) 1–11.
71. M. Mirolli, V. G. Santucci and G. Baldassarre, Phasic dopamine as a prediction error of intrinsic and extrinsic reinforcements driving both action acquisition and reward maximization: A simulated robotic study, *Neural Networks* **39** (2013) 40–51.
72. Y. Gatsoulis and T. M. McGinnity, Intrinsically motivated learning systems based on biologically-inspired novelty detection, *Rob. Auton. Syst.* **68** (2015) 12–20.