Vector LIDA

Javier Snaider and Stan Franklin
Computer Science and Institute for Intelligent Systems,
University of Memphis, Memphis, TN 38152 USA
jsnaider@memphis.com, franklin@memphis.com

Abstract
The representation paradigm used by a cognitive architecture helps to determine the kind of processes that it can perform more efficiently. Vector LIDA is a variation of the LIDA cognitive architecture that employs high-dimensional Modular Composite Representation (MCR) vectors as its main representation model and Integer Sparse Distributed Memory as its main memory implementation technology. The advantages of this new model include a more realistic and biologically plausible model, better integration with its episodic memory, better integration with other low level perceptual processing (such as deep learning systems), better scalability, and easier learning mechanisms. Here, after briefly recapping the LIDA model and MCR, we describe Vector LIDA and argue for its several advantages.
Keywords: Cognitive Architectures, Semantic Vectors, High Dimensional Representations, Modular Composite Representation

1 Introduction

Several authors (Franklin, 1995; Kanerva, 1988, 2009; Winston, 1992) have pointed out the importance of representations to perform tasks efficiently and solve problems. Winston described the representation principle in these words: “Once a problem is described using an appropriate representation, the problem is almost solved” (1992, p. 18). Franklin (1995, p. 365) discussed the importance of representation for both symbolic AI and connectionist models. Kanerva (2009) pointed out how a representation can facilitate certain tasks at the expense of others. This is particularly true for cognitive architectures.

Many of the more mature cognitive architectures, e.g., SOAR, ACT-R, are symbolic in nature, processing by the manipulation of amodal symbols in the sense of Barsalou (1999). Throughout the early decades of its existence, SOAR was essentially a production system depending heavily on production rules (productions) as the primary data structure (Laird, Newell, & Rosenbloom, 1987). More recently new capabilities have required representations using new data structures (Laird, 2008). Although it is also basically a production system, ACT-R also employs spreading activation (Anderson et al, 1997), moving it a little in the direction of connectionist models (but see also Lebiere & Anderson, 1993). Johnson (1997) points out that “The data structures used for representation in these architectures (SOAR, ACT-R) critically affect their capabilities for particular tasks. … Soar can model extremely flexible control, but has difficulty accounting for probabilistic operator selection and the independent effects of history and distance to goal on the likelihood of selecting an operator. In contrast, ACT-R's control is well supported by empirical data, but has difficulty modeling task-switching, multiple interleaved tasks, and dynamic abandoning of sub-goals."

Other symbolic architectures utilize different basic data structures, for example Icarus with its probabilistic concepts (Langley, McKusick, Allen, Iba, & Thompson, 1991). Still others are hybrid, using both symbolic data structures and artificial neural networks, for example CLARION (Sun, 1997). Langley, Laird and Rogers have provided a very useful review of the state of cognitive architecture research (2009).

High-dimensional vector spaces have interesting properties that make them attractive for representation models for cognitive architectures. The distribution of the distances between vectors in these spaces, and the huge number of possible vectors, allow a noise-robust representation model where the distance between vectors denotes the similarity (or dissimilarity) of the concepts they represent. Moreover, these high-dimensional vectors can be used to represent complex structures, where each vector denotes an element in the structure. However, a single vector can also represent one of these same complex structures in its entirety by implementing a *reduced description*, a mechanism to encode complex hierarchical structures in vectors or connectionist models (Hinton, 1990). These reduced description vectors can be expanded to obtain the whole structure, and can be used directly for complex calculations and procedures, such as making analogies, logic inference, or structural comparison (see Kanerva, 2009; Plate, 2003 for further discussion of these aplications). There are some existing projects that employ high-dimensional vectors and reduced descriptions. For example, Stewart and colleagues produced an application based on spiking neurons that is capable of reasoning and planning. The rules and system states are encoded using reduced description vectors (Stewart & Eliasmith, 2011). Also Jones and Mewhort (2007) employ reduced descriptions for BEAGLE, a semantic space model that mimics several human memory properties. More recently, the Sigma cognitive architecture introduced high dimensional vectors based on the Modular Composite Representation (described below) and BEAGLE for representing words (Ustun, Rosenbloom, Sagae, & Demski, 2014). However, to the best of our knowledge, there is no cognitive architecture that employs these technologies as its primary representation model.

Vector LIDA is a promising improvement of the LIDA cognitive architecture's conceptual model (Franklin, Strain, Snaider, McCall, & Faghihi, 2012) and its implementation (Snaider, McCall, & Franklin, 2011), that employs high-dimensional vectors and reduced descriptions. The previous version of the LIDA model utilizes nodes and links in a directed graph-like structure (node structure) as its main data structure. This implementation introduces several problems in scalability, flexibility, and representation capability. The new implementation described here utilizes high dimensional vectors that will help to solve the current problems, and will produce more biologically plausible mechanisms.

The next section will briefly introduce the LIDA architecture. The following sections will describe some important concepts and technologies necessary to describe the new representation of the LIDA model and its advantages: vector representations, Modular Composite Representation, and Sparse Distributed Memory and some of its extensions. We will then describe the changes introduced in Vector LIDA and their benefits, and finally we will present some conclusions.

2 The LIDA Model and its Architecture

The LIDA model (Baars & Franklin, 2009; Franklin & Patterson, 2006; Ramamurthy, Baars, D'Mello, & Franklin, 2006) is a comprehensive, conceptual and computational model covering a large portion of human cognition[*]. Based primarily on Global Workspace theory (Baars, 1988, 2002) the model implements and fleshes out a number of psychological and neuropsychological theories. The LIDA computational architecture (Snaider et al., 2011) is derived from the LIDA cognitive model. The LIDA model and its ensuing architecture are grounded in the LIDA cognitive cycle. Every autonomous agent (Franklin & Graesser, 1997), be it human, animal, or artificial, must frequently sample (sense) its environment and select an appropriate response (action). More sophisticated agents, such as humans, process (make sense of) the input from such sampling in order to facilitate their decision-making. The agent's "life" can be viewed as consisting of a continual sequence of these cognitive cycles. Each cycle constitutes a unit of sensing, attending and acting. A cognitive cycle can be thought of as a moment of cognition, a cognitive "moment." Higher order cognitive processes (deliberation, planning, volitional decision making, etc.) are implements as sequences of these cognitive cycles.

We will now briefly describe what the LIDA model hypothesizes as the rich inner structure of the LIDA cognitive cycle. More detailed descriptions are available elsewhere (Baars & Franklin, 2003; Franklin, Baars, Ramamurthy, & Ventura, 2005). During each cognitive cycle the LIDA agent first makes sense of its current situation as best as it can *by updating its representation of its current situation, both external and internal*. By a competitive process, as specified by Global Workspace Theory (Baars, 1988), it then decides what portion of the represented situation is the most salient, that is, most in need of attention. Broadcasting this portion, the current contents of consciousness[†], enables the agent to choose an appropriate action and execute it, completing the cycle.

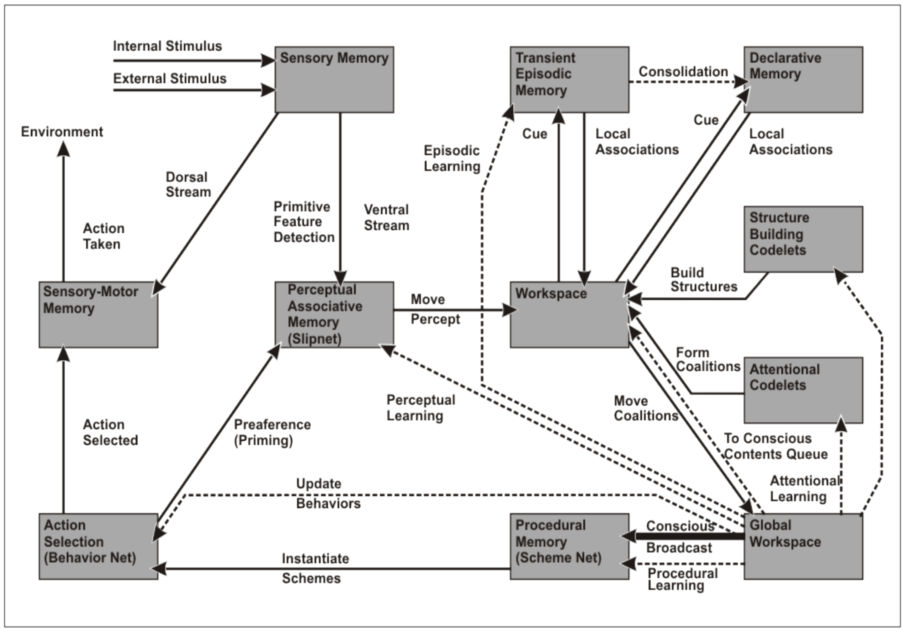

Figure 1. LIDA cognitive model diagram. The boxes represent the different modules of the model, and the arrows the interactions among them. Each module behaves asynchronously.

[*] "Cognition" is used here in a particularly broad sense, so as to include perception, feelings and emotions, and actions.
[†] Here "consciousness" refers to functional consciousness Franklin (2003). We take no position on the need for, or possibility of, phenomenal consciousness.

Thus, the LIDA cognitive cycle can be subdivided into three phases, the understanding phase, the attention (consciousness) phase, and the action selection phase. Figure 1 should help the reader follow the description. It starts in the upper left corner and proceeds roughly clockwise.

Beginning the understanding phase, incoming stimuli activate low-level feature detectors in Sensory Memory. The output is sent to Perceptual Associative Memory where higher-level feature detectors feed in to more abstract entities such as objects, categories, actions, feelings, events, etc. The resulting percept moves to the Workspace where it cues both Transient Episodic Memory and Declarative Memory producing local associations. These local associations are combined with the percept to generate a Current Situational Model, which represents the agent's understanding of what is going on right now.

Attention Codelets[‡] begin the attention phase by forming coalitions of selected portions of the Current Situational Model, and moving them to the Global Workspace. A competition in the Global Workspace then selects the coalition with the most salient (relevant, important, urgent, novel, unexpected, bright, loud, moving, etc.) portion. The contents of this winning coalition become the content of consciousness. These conscious contents are then broadcast globally, initiating the action selection phase. The action selection phase of LIDA's cognitive cycle is also a learning phase in which several processes operate in parallel (see Figure 1). New entities and associations, and the reinforcement of old ones, occur as the conscious broadcast reaches Perceptual Associative Memory. Events from the conscious broadcast are encoded as new memories in Transient Episodic Memory. Possible action schemes, together with their contexts and expected results, are learned into Procedural Memory from the conscious broadcast. Older schemes are reinforced. In parallel with all this learning, and using the conscious contents, possible action schemes are recruited from Procedural Memory. A copy of each such is instantiated with its variables bound, and sent to Action Selection, where it competes to be the behavior selected for this cognitive cycle. The selected behavior triggers Sensory-Motor Memory to produce a suitable motor plan for the execution of the behavior. Its execution, internal or external, completes the cognitive cycle.

The Workspace requires further explanation. Its internal structure is composed of various input buffers and three main modules: the Current Situational Model, the Scratchpad and the Conscious Contents Queue (Snaider, McCall, & Franklin, 2012). The Current Situational Model is where the structures representing the actual current internal and external events are stored. Structure-building codelets are responsible for the creation of these structures using elements from the various submodules of the Workspace. The Scratchpad is an auxiliary space in the Workspace where structure-building codelets can construct possible structures prior to moving them to the Current Situational Model. The Conscious Contents Queue holds the contents of the last several broadcasts, and permits LIDA to understand and manipulate time-related concepts (Snaider et al., 2012).

3 Vector Representations, Reduced Descriptions, and MCR

In many subfields of computer science vector representations constitute one of the main types of data structure. For example in machine learning, a vector of features–often of different data types–represents an element in a training set. A different approach closer to the focus of this work utilizes vectors where all the dimensions share the same data type. Even with this uniformity, the number of possible representation models is limitless. The way of calculating or defining the vector representation for an item, and the distance or similarity measurement, define the representation and its properties. For example, in the last two decades a large number of semantic space models have emerged that use high dimensional vectors to represent words and texts. The most representative models include Latent Sematic Analysis (LSA) (Deerwester, Dumais, Furnas, Landauer, & Harshman,

[‡] A codelet is a small piece of code that performs a specific task in an independent way.

1990) based on statistical analysis; Random Indexing (Sahlgren, 2005), which employs random sparse vectors and random permutations; and BEAGLE (Jones & Mewhort, 2007), which computes vectors using circular convolution. There are recent surveys of semantic space models (Cohen & Widdows, 2009; Turney & Pantel, 2010).

In an even more generic view, vectors can represent any concept or element of interest: objects, features, rules, constraints, actions, etc. As explained above, when a vector belongs to a high dimensional space, interesting properties arise. The distribution of the distances between one given vector and the rest of the vectors in the space produces most of these properties. For example, in a binary space with 1,000 dimensions, the distribution of Hamming distances can be approximated with a normal distribution with mean equal to 500 and standard deviation approximately equal to 15.81. In other words, most of the vectors of the space are at a distance of around 500 bits from a given vector. For example, the probability of choosing two random vectors in this space with a distance among them of 300 bits or less is 4.55×10^{-37}, which is almost 0. Kanerva (1988) defines this property as *tendency to orthogonality*, making these vectors good candidates for representing unrelated concepts.

These spaces offer an enormous number of possible activation patterns with which to represent individual items, and the necessity for compact representations becomes less critical. There is no need for a one to one correspondence between patterns and items. For example, in the same binary space described above, we can theoretically represent 2^{1000} different items, but this is highly unlikely. We can use just a fraction of the vectors in the space, say 2^{100} vectors uniformly distributed in the space, which still allows a gigantic number of possible representations. Even after adding some noise by introducing a few changes in one of these vectors, it can still represent the same item. In other words, *a region of the space, instead of just one point, represents an item in a distributed manner, creating a more noise robust representation that gracefully degrades as noise increases, and produces desirable properties such as pattern completion*.

One frequent criticism of distributed representations (or vector representations) is the difficulty they pose in the representation of complex structures. Performing high level cognitive tasks such as reasoning, planning, or action selection often involves structures with multiple components. Implementations of these tasks frequently utilize structures such as sequences or hierarchies, and require variable binding. Moreover, the components of these structures can in turn be complex structures themselves. Of course, we can create these structures and use vectors as components. But, in that case, the vectors become mere symbols, with a significant loss of expressive power. Hinton (1990) introduced the concept of *reduced description*, a method for encoding complex structures as single vectors. The main idea is to have a dual representation: the structure can be represented explicitly, with a vector for each component, or as a reduced description, where a single vector represents the whole structure. When the system focuses on a particular composite element, its constituent structure is represented in full, instantiating all the elements (vectors) that compose it. On the other hand, when the element participates in the structure of another element that has the current focus, it is represented with a single vector as a reduced description. See Figure 2.

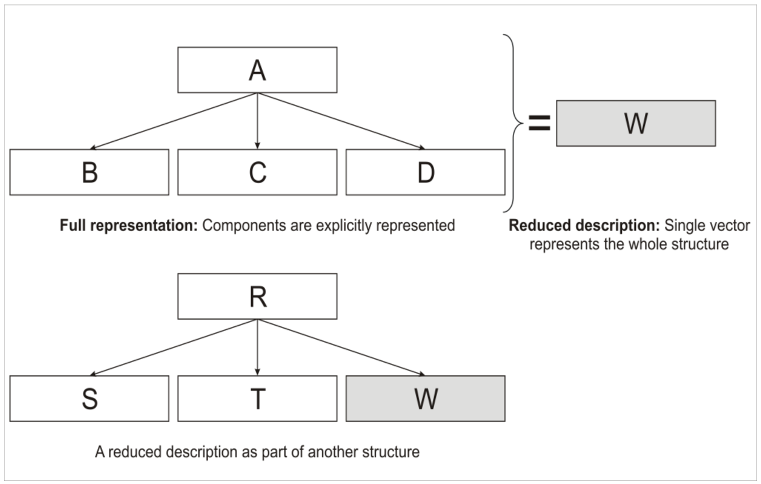
A
B
C
D
W
Full representation: Components are explicitly represented
Reduced description: Single vector represents the whole structure
R
S
T
W
A reduced description as part of another structure

Figure 2. Reduced description. A complex structure has a dual representation: a full representation with an explicit structure where each element is a vector, and a reduced description, where a single vector

Plate (2003, pp. 19-20) analyses reduced descriptions from four desirable characteristics: *representation adequacy* (full descriptions can be reconstructed from the reduced ones), *reduction* (the reduced descriptions have a size similar to their components), *systematicity* (the process of constructing the reduced description must be well known and deterministic), and *informativeness* (the reduced description encloses information about the whole it represents). He also discussed *explicit similarity*; that is, similar elements have similar representations.

Many complex structures apply pervasively to processes in cognitive architectures; examples include sequences, hierarchies, and predicates (i.e. rules with variable binding). These structures, and probably others, can be constructed out of even simpler primitives such as binding and grouping. *Binding* is the assignment of one element, which is called the filler, to a particular role or position in the structure. *Grouping* is forming a set (or collection) of elements. To create a reduced description model, we need to define binding and grouping operations, and a distance or similarity measure. Kanerva (2009) introduced more abstract names for these operations; he uses multiplication for binding, and sum for grouping, which simplifies the operations' notation. We will use this same convention here.

Modular Composite Representation

Modular Composite Representation (MCR)(Snaider & Franklin, in press) is a new reduced description model that employs long integer vectors. This representation paradigm has properties similar to Spatter Code (Kanerva, 1994), which uses binary vectors, and to Holographic Reduced Representations (HRR) (Plate, 1995, 2003), based on vectors of real or complex numbers. Spatter Code uses binary vectors, which limit the model's expressiveness. Some of its required operations, such as normalization, introduce excessive noise into the vectors that can diminish the performance of the model. Holographic Reduced Representation (HRR), on the other hand, has a rich representation capability, but requires complex operations such as circular convolution. Moreover, the vectors must follow a normal distribution for each dimension, which further complicates its use. MCR is an intermediate point between these two models, balancing representational expressiveness and implementational simplicity. Since MCR is a key component of the Vector LIDA project introduced in Section 5, we will describe here its main operations and characteristics. For a more detailed description and analysis see (Snaider & Franklin, in press).

Modular integer vectors have a defined integer range of possible values for each dimension. For example, the range of values can be {–8, 7} or {0, 15}. Although any range of values is possible, for simplicity in the notation, we will use ranges with 0 as the lower bound and $r - 1$ as the upper bound, and only even values of r. In more formal notation, MCR employs vectors within a multidimensional space, $v \in \mathbb{Z}_r^n$, where n is the number of dimensions of the space and r is the size of the range of values for each dimension.

Modular Composite Representation

Two basic operations, grouping and binding, constitute the heart of the reduced description models. Grouping (or sum) operation is used to create sets or groups of elements, and binding (or multiplication) creates representations for bonds among elements, such as in the role-filler case. Given that the required properties of these operations are responsible for the behavior and characteristics of the reduced description models, each model can define these operations according to the characteristics of its vector space. In this way, we can abstract the reduced description model ideas and its basic operations to explore problems and applications independently of the reduced description implementation. For example, consider the following expression that represents a red circle:

$$F = [circle \otimes Shape + red \otimes Color]$$

where *circle*, *Shape*, *red*, and *Color* are vectors, and the symbols ⊗ and + represent the binding and grouping operations respectively. This expression can work in any reduced description model with appropriate definitions for grouping and binding.

The distance measure employed by MCR is a variation of the Manhattan distance that complements the modular arithmetic characteristic of the dimensions of the space:

$$d(u,v) = \sum_i min\big(mod_r(u_i - v_i), mod_r(v_i - u_i)\big)$$

3.1.1 MCR Binding

The binding (or multiplication) of modular integer vectors is defined as the modular sum in each dimension. For example, the multiplication of two vectors A and $B \in \mathbb{Z}_{16}^n$, with values for dimension *i* equal 10 and 12 respectively, produces a new vector *C* with dimension *i* equals to 6.

$$C_i = mod_r(A_i + B_i)$$

This operation resembles the bitwise XOR used in Spatter Code.[§]

The unbinding operation is simply the modular subtraction in each dimension, or the modular sum of the first operand with the complement of the second operand in each dimension. This leads to the definition of the inverse vector in this model. The inverse of the vector *A* is another vector A^{-1} such that each dimension *i* of A^{-1} is the complement of the value of *A* in the same dimension:

$$A_i^{-1} = mod_r(r - A_i)$$

This multiplication operation is associative, commutative, distributive over the sum (see below), preserves distances

$$d(A,B) = d(A \otimes C, B \otimes C),$$

and produces vectors that tend to differ from the operands.

$$A \otimes B \not\approx A \text{ and } A \otimes B \not\approx B$$

where $\not\approx$ denotes dissimilarity.

3.2.1 MCR Grouping

The grouping (or sum) utilizes a mechanism similar to the sum operation defined for HRR in the frequency domain (Plate, 2003, p. 146). Let us consider each possible value as a vector of unit length in a plane, called an *equivalent vector*. The center of the circle in Figure 3 corresponds to the coordinate's origin in this plane. For example, the equivalent vector for the value zero is (0, 1) and the value seven corresponds to the equivalent vector $(\sqrt{2}, -\sqrt{2})$. This sum operation involves two steps to

[§] Actually, XOR is a special case of the modular sum when $r = 2$.

calculate the value of each dimension *i*: the equivalent vector sum and the normalization. The first step consists of calculating the rectangular sum (i.e. their vector sum) of the equivalent vectors corresponding to the values of each operand for dimension *i*. Second, the normalization process calculates the value of each dimension of the group vector as the closest value corresponding to the resultant vector normalized to length one. Since the dimensions have only *r* possible values, a table with the equivalent vectors' components and the tangent of their angles can speed up the calculation and normalization processes. Figure 3 shows the representation of the equivalent vectors and a couple of examples of grouping.

We can extend the definition of this sum for the case of more than two operands by simply summing, in each dimension, all the equivalent vectors of the operands for each dimension before normalizing. Grouping several operands in this way produces more consistent results than summing and normalizing in each individual group operation between two operands. Figure 3 depicts the result for combining three vectors that have values 0, 7, and 13 respectively for a given dimension.

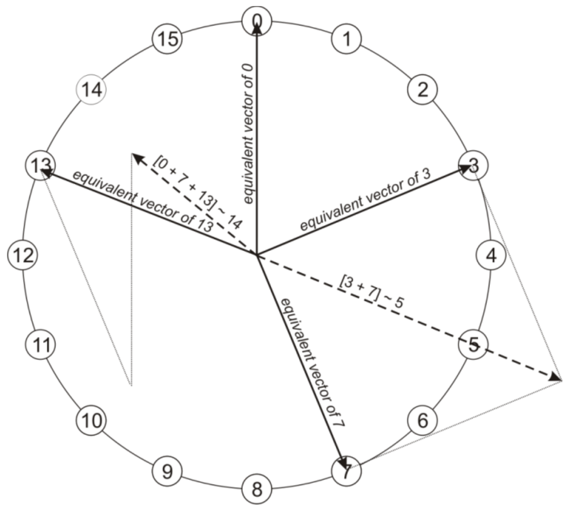

Figure 3. Equivalent vectors and examples of grouping.

The most important characteristic of the grouping operation is producing vectors similar to its operands. This similarity allows identifying a composed vector from some of its elements, and vice versa, a fundamental property of reduced description models. The grouping operation is also commutative. These properties (and the properties of the binding operation) are responsible for the behavior and characteristics of the reduced description model. (See Kanerva, 2009; Snaider & Franklin, in press for a detailed discussion of this subject.)

4 Sparse Distributed Memory and its Extensions

Some of the MCR operations produce approximate versions of the vectors on which they operate. An auto-associative *cleanup* memory that stores the vectors known by the system (i.e., all the vectors used in the representations) can retrieve the original (clean) vectors. Integer Sparse Distributed Memory, an extension of the original Sparse Distributed Memory (SDM) that works with modular integer vectors, is a good option for cleanup memory for the MCR model. Moreover, the innate characteristics of this memory make it an attractive option for implementing memory modules in Vector LIDA.

First proposed by Kanerva (1988, 1993), SDM is a biologically inspired mathematical model of human long term memory based on large binary vectors. This memory has several desirable properties. It is distributed, auto-associative, content addressable, and noise robust. Moreover, it exhibits one-shot learning, is resilient to damage, and its contents degrade gracefully. It also possesses interesting psychological characteristics as well (interference, knowing when it doesn't know, the tip of the tongue effect), that make it an good candidate with which to model episodic memory (Baddeley, Conway, & Aggleton, 2001; Franklin et al., 2005). Implementations of SDM are ongoing for various applications (Bose, Furber, & Shapiro, 2005; Furber, Bainbridge, Cumpstey, & Temple, 2004; Jockel, 2009; Mendes, Coimbra, & Crisóstomo, 2009; Meng et al., 2009). Several improvements and variations have been proposed for SDM; for example Ramamurthy and colleagues introduced forgetting as part of an unsupervised learning mechanism (Ramamurthy, D'Mello, & Franklin, 2006; Ramamurthy & Franklin, 2011). Furthermore, SDM's structure is ideal for parallel processing or hardware implementation.

Extended SDM increases the hetero-associativity feature of the memory. A novel mechanism using this extension is particularly effective for sequence learning (Snaider & Franklin, 2011; Snaider & Franklin, 2012a). The main idea of this new memory structure is the use of vectors with different lengths for the addresses and for the words. A word has a longer length than the address in which it is stored. Each address has n dimensions while each word has m dimensions with $n < m$. Moreover, the address vector is included in the word vector. Formally, a word of length m and an address with length n, the first n bits of the word compose the address.

Conceptually, this memory is a mix of auto-associative and hetero-associative memories. The address part of the word is auto-associative whereas the rest of the word is hetero-associative. This allows us to preserve, and even to improve, the desirable characteristics of the SDM. First, with an initial vector as address to cue the memory, it is possible to retrieve the corresponding word, even if the initial vector is a noisy version of the stored one. This means that ESDM maintains the noise robustness characteristic of SDM. Second, the data of each vector is stored in a number of hard locations in a distributed way. So, it is also robust in the case that some hard locations are corrupted or lost. Third, the previously discussed psychological characteristics in SDM are also present in ESDM. Finally, the hetero-associative part of the words in ESDM allows storing other data related with the address data, but without interfering with it. For a complete description of this memory ant its improved sequence learning mechanism see (Snaider & Franklin, 2012a).

The original SDM uses binary vectors for both addresses and data words. This usage results in several limitations. First, real data are not always Boolean, making representations using more than two values desirable. A possible solution for this limitation is to use several dimensions of the word vectors to represent one feature, but this approach does not fit very well with the structure of SDM. Mendes and colleagues (2009) studied this problem in detail. They evaluated several binary encodings to use with SDM in robot navigation tasks, and reported their difficulties and limitations. Another disadvantage of binary vectors is the loss of information due to the noise introduced into the representation by the normalization used in combining vectors. Vectors can be summed up dimension by dimension (for this operation, vectors belonging to $\{0; 1\}^n$ are replaced by vectors of $\{-1; +1\}^n$). This operation produces a vector belonging to $\mathbb{Z}^n$, the space of integer vectors with n dimensions. The normalization process reduces the resultant to a vector that is also in $\{-1; +1\}^n$ but with significant loss of information. See for example (Kanerva, 2009; Snaider & Franklin, 2011; Snaider & Franklin, 2012a).

A variation of SDM, Integer Sparse Distributed Memory (Integer SDM) (Snaider & Franklin, 2012b), is also based on large vectors, on the order of thousands of dimensions, where in this case each dimension has a range of possible *integer* values. This memory has properties similar to the original one, noise robustness, auto-associativity and being distributed. A further extension of Integer

SDM permits words and addresses of different lengths, which is particularly useful for the reliable storage of sequences and other data structures. In addition, this memory avoids the limitations imposed by binary representation, as described above, allowing a better encoding of non-binary data and alleviating the normalization problem when combining several vectors.

The benefits of both models are retained when merging this model with Extended SDM into a combined ESDM model that uses integer vectors, has better hetero-associativity support, and improves sequence learning. These models can be further expanded, for instance with the forgetting mechanism (Ramamurthy, D'Mello, et al., 2006), which can be expected to improve the unsupervised learning capabilities of the memory.

5 Vector LIDA

A promising change in the LIDA conceptual model and its implementation, called Vector LIDA, would intensively utilize the technologies presented in the previous sections. This new version implements the LIDA architecture (Franklin & Patterson, 2006; Snaider et al., 2011) using MCR vectors as its main representation for data structures, and the various extensions of SDM described here for its main memory mechanisms. Figure 1 depicts the structure of the LIDA model, including its modules and their interactions.

The previous version of the LIDA model utilizes nodes and links in a directed graph-like structure (node structure) as its main data structure. This implementation introduces several problems. First, comparing node structures can be computationally expensive. Moreover, some of LIDA's processes require approximate comparisons of the node structures, which can be even harder to compute. MCR vectors can represent information such as that contained in node structures, but unlike node structures, MCR vectors have an innate approximate comparison property, as explained in Section 3.

Second, some modules in the LIDA model, such as perceptual associative memory, episodic memory, and procedural memory, require the implementation of learning mechanisms. These mechanisms must be able to learn new node structures in an instructionalist learning mode, and reinforce previous ones via reinforcement learning. The previous model uses a value attached to each node and link, called base level activation, that helps to implement reinforcement learning. However, the model does not have a generic strategy for the learning of new elements, and the current implementations of several modules do not scale well. The SDM variations described here have both the required learning mechanisms (instructionalist and reinforcement) integrated into their basic functionality. Learning new vectors (instructionalist learning) simply consists of storing the vector in the memory using its standard storage procedure. When the same vector is stored several times (reinforcement), the hard locations' counters corresponding to the values of each dimension of this vector will have larger counts, making it resistant to interference by other vectors stored in the memory. This effect would improve implementing a forgetting mechanism.

Moreover, the original implementation of the episodic memory module in LIDA already employs a SDM memory as its base implementation. The problem of translating back and forth from node structures to vectors in episodic memory disappears when using MCR vectors as the main data structure of LIDA. Furthermore, the sequence storage mechanism of Extended ISDM enables the episodic memory module to store composite events, sequences of simpler events, improving the event-learning capability of the episodic memory module.

Third, MCR vectors have the potential of implementing directly Barsalou's perceptual symbol system (1999), which uses symbols grounded in sensory and motor information. Although the original LIDA model employs a version of perceptual symbols, it does not exploit their capability for expressiveness, and they have a limited impact on the functionality of the whole system. Nodes in LIDA are grounded in sensory data. The activation of a node depends on the activation of its child nodes, which ultimately are activated from sensory data. However, a node (without considering its children) does not represent any specific sensory or motor information by itself, so its grounded feature is seldom employed in the LIDA model processes. Moreover, the simulator idea, central to the perceptual symbol system theory, is hard to implement using nodes and node structures. On the other hand, constructing MCR vectors from sensory and motor information using hyperdimensional computing operations (Kanerva, 2009) would produce representations that have many of the perceptual symbols' characteristics described by Barsalou (1999). Similar sensory information would yield similar representations, and the holistic processing operations of MCR could facilitate the implementation of the simulators described in his model. Interestingly, MCR vectors with role-filler components for each modality have the potential to integrate several modalities in a single representation, addressing the so-called binding problem. For example, the MCR vector B

$$B = [birdImage \otimes Visual + birdSong \otimes Auditory]$$

represents the integration of the data from the visual and auditory modalities. Notice that the vectors *birdImage* and *birdSong* would be in turn reduced descriptions also.

Moreover, the homogeneous representation of different levels of abstraction in a hierarchy of concepts, or even in the representation of sequences, plans, and other cognitive entities, may simplify the implementation of high-level cognitive processes such as metacognition, theory of mind, and deliberation. For example, a single MCR vector a can represent an event, and another vector b can represent "thinking about that event". The vector a can be a component of vector b.

Fourth, in recent years, several models of the so-called *deep learning* systems, such as HMAX (Serre, Wolf, Bileschi, Riesenhuber, & Poggio, 2007), HTM (George, 2008), DeSTIN (Arel, Rose, & Coop, 2009), and deep belief nets (Hinton, 2007; Hinton, Osindero, & Teh, 2006), have emerged. These models, based on the hierarchical organization of the neocortex and particularly of the visual cortex, focus on learning and recognition of spatial and temporal patterns. They detect pattern invariances in space and (in some models) in time in each level of the hierarchy. The output of a lower layer provides the input for the higher ones. The higher the layer, the more abstract are the features they capture of the data. These hierarchical networks provide biologically plausible mechanisms with which to perceive both spatial and temporal patterns from low level sensory data, making them attractive for modeling low level perception between sensory memory and perceptual associative memory (PAM) in LIDA (see Figure 1). Since these models in general produce high dimensional vectors as output, interfacing them with Extended Integer SDM memories for implementing PAM would be simpler, more scalable, and more noise robust than with the current implementation. HMAX (Serre et al., 2007) is probably the most biologically realistic hierarchical model for this function, since their authors designed it following the biological data as accurately as possible, but other models such as HTM (George, 2008) or DeSTIN (Arel et al., 2009) are also possible options. Furthermore, these hierarchical models have the potential of detecting spatial-temporal patterns in other modules, such as the workspace or perceptual memory, and they would seamlessly integrate with MCR vectors. For example, attention and structure-building codelets (see Figure 1) can be implemented with these hierarchical networks so as to detect patterns in the workspace, and build coalitions and complex structures, respectively, with these patterns. A similar implementation for procedural memory, using hierarchical networks, could improve the detection and learning of temporal patterns that eventually became sequences of actions or *behavior streams* (D'Mello, Ramamurthy, Negatu, & Franklin, 2006). These hierarchical network models, combined with MCR vectors and Extended Integer SDM, have the potential to provide a primary biological plausible and efficient detection algorithm in LIDA. McCall implemented a variation of HTM's Cortical Learning Algorithm aimed at integrating it into the LIDA architecture (McCall, 2014). Fifth, using MCR vectors would produce a more biologically plausible model through its synergy with other models, such as the hierarchical networks mentioned above, Barsalou's perceptual symbols, Fuster's *cognits* (2006), and several neurodynamical theories (Franklin et al., 2012). We have already described (see above) how to implement perceptual symbols, and how their construction addresses the binding problem. A discussion follows of the relationship between MCR vectors and both cognits and neurodynamical theories.

Fuster described a cognit as an abstraction of a network of neurons. Its representation power comes from the activity of the neurons that compose it and specially the relationship between its component neurons. He extensively describes how different memory types (e.g., episodic, perceptual, motor, etc.) can be interpreted as hierarchies of cognits. He pointed out that cognits in one level of this hierarchy can be a composition of other cognits from several levels in the hierarchy. MCR vectors may be used as an abstraction of the cognit model. They are also distributed, can combine elements of various levels of the memory hierarchy in a single vector, and their hyperdimensional operations can combine and associate cognits represented as vectors.

Franklin and colleagues (2012) have compared several neurodynamical theories with the LIDA model. By interpreting the brain as a dynamical system, the representations would be trajectories in the phase space (pattern of activation space) of one or several cell assemblies. These trajectories can in turn interfere with and influence the trajectory in the phase space of other cell assemblies. A MCR vector would model not only a pattern of activation of a cell assembly, but also a trajectory of these patterns. For example, if a single neuron in a cell assemble has a sequence of activations in a trajectory of 4 steps (e.g., 1011, where one and zero mean high firing rate and low firing rate respectively), we may code this sequence as a single value (11 in the example) and assign this value to one dimension in our MCR vector. Employing the same procedure for each neuron in the cell assembly, produces a MCR vector that represent the trajectory of the pattern. Using an Integer SDM as a cleanup memory can produce a previously stored vector from a partial vector, which would model the oscillatory and self-organizing properties of the dynamical system interpretation. Using random permutation or multiplication produces a new vector that would model the influence from one cell assembly on another. Although these ideas are still under development, using MCR vectors and the memories proposed herein are good candidates to model representations and cognitive processes in a more biologically plausible way.

Moreover, the vector nature of these models makes them ideal for being implemented in new parallel technologies such as multicore processors and GPUs, which have the potential to speed up these algorithms by several orders of magnitude. For example, our first experiments with GPU's implementations of Extended SDM showed a speedup of 49 times the performance of the mono-processor implementation (Snaider, 2012).

Several pieces of this new implementation have already been implemented, including the MCR vector operations, the Extended ISDM, and the basic data structure of the model (which replaces the previous node-structure implementation). Various experiments involving these modules were reported in previous papers (Snaider & Franklin, 2012a, 2012b, in press). Integration of vector representation into the different modules of the LIDA model is currently in progress.

6 Conclusions

The representation paradigm employed by a system greatly influences the kind of problems that it can efficiently solve. The features of high-dimensional vectors and reduced description models make them good candidates for a representation paradigm for cognitive architectures. In particular, they allow a noise-robust representation model where the distance between vectors measures the similarity (or dissimilarity) of the concepts they represent, using reduced description models. This includes complex structures such as hierarchies and sequences. The approximate similarity comparisons and the pattern matching intrinsic operations of these vectors are clear advantages over other more brittle representations such as nodes and links.

The changes in the LIDA cognitive architecture aim to exploit the characteristics of high-dimensional vector representations. In particular, the MCR model, a reduced description model instantiation, balances representational expressiveness and implementation simplicity. The Integer SDM (and its combination with Extended SDM) described here, which employs the same vectors as MCR, can be used as cleanup memory for this model. Moreover, this memory exhibits useful properties, such as noise robustness and one shot learning, that make it attractive for implementation technology of several memory modules in LIDA.

Some of the advantages of Vector LIDA over the current implementation include a more realistic and biologically plausible model, including a representation better grounded in sensory and motor data, better integration with its episodic memory, better integration with deep learning architectures such as the variations of CLA proposed by McCall for LIDA (2014) or HMAX (Serre et al., 2007), better scalability, and easier learning mechanisms.

References

Arel, I., Rose, D., & Coop, R. (2009). DeSTIN: A Scalable Deep Learning Architecture with Application to High-Dimensional Robust Pattern Recognition. *Proc. of the AAAI 2009 Fall Symposium on Biologically Inspired Cognitive Architectures (BICA)*.

Baars, B. J. (1988). *A Cognitive Theory of Consciousness*. Cambridge, UK: Cambridge University Press.

Baars, B. J. (2002). The conscious access hypothesis: origins and recent evidence. *Trends in Cognitive Science, 6*, 47–52.

Baars, B. J., & Franklin, S. (2003). How conscious experience and working memory interact. *Trends in Cognitive Science, 7*, 166–172.

Baars, B. J., & Franklin, S. (2009). Consciousness is computational: The LIDA model of Global Workspace Theory. *International Journal of Machine Consciousness, 1*(1), 23-32.

Baddeley, A. D., Conway, M., & Aggleton, J. P. (2001). *Episodic Memory*. Oxford, UK: Oxford University Press.

Barsalou, L. W. (1999). Perceptual symbol systems. *BEHAVIORAL AND BRAIN SCIENCES, 22*, 577–609.

Bose, J., Furber, S. B., & Shapiro, J. L. (2005). Spiking neural sparse distributed memory implementation for learning and predicting temporal sequences. *Lecture Notes in Computer Science, 3696/2005*, 115 - 120.

Cohen, T., & Widdows, D. (2009). Empirical distributional semantics: Methods and biomedical applications. *Journal of Biomedical Informatics, 42*(2), 390-405. doi: 10.1016/j.jbi.2009.02.002

D'Mello, S. K., Ramamurthy, U., Negatu, A., & Franklin, S. (2006). A Procedural Learning Mechanism for Novel Skill Acquisition. In T. Kovacs & James A. R. Marshall (Eds.), *Proceeding of Adaptation in Artificial and Biological Systems, AISB'06* (Vol. 1, pp. 184–185). Bristol, England: Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

Deerwester, S. C., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science, 41*(6), 391-407.

Franklin, S. (1995). *Artificial Minds*. Cambridge MA: MIT Press.

Franklin, S. (2003). IDA: A Conscious Artifact? *Journal of Consciousness Studies, 10*, 47–66.

Franklin, S., Baars, B. J., Ramamurthy, U., & Ventura, M. (2005). The Role of Consciousness in Memory. *Brains, Minds and Media, 1*, 1–38.

Franklin, S., & Graesser, A. C. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents *Intelligent Agents III* (pp. 21–35). Berlin: Springer Verlag.

Franklin, S., & Patterson, F. G. J. (2006). The LIDA Architecture: Adding New Modes of Learning to an Intelligent, Autonomous, Software Agent *IDPT-2006 Proceedings (Integrated Design and Process Technology)*: Society for Design and Process Science.

Franklin, S., Strain, S., Snaider, J., McCall, R., & Faghihi, U. (2012). Global Workspace Theory, its LIDA Model and the Underlying Neuroscience. *Biologically Inspired Cognitive Architectures, 1*, 32-43.

Furber, S. B., Bainbridge, W. J., Cumpstey, J. M., & Temple, S. (2004). A Sparse Distributed Memory based upon N-of-M Codes. *Neural Networks, 17*(10), 1437-1451.

Fuster, J. M. (2006). The cognit: A network model of cortical representation. *International Journal of Psychophysiology, 60*(2), 125-132. doi: 10.1016/j.ijpsycho.2005.12.015

George, D. (2008). *How the brain might work: A hierarchical and temporal model for learning and recognition. PhD Thesis.* (PhD), Stanford University.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence,* (46), 47-75.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences, 11*(10), 428-434.

Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*, 1527-1554.

Jockel, S. (2009). *Crossmodal Learning and Prediction of Autobiographical Episodic Experiences using a Sparse Distributed Memory. PhD Thesis.* (PhD), University of Hamburg, Hamburg.

Johnson, T. R. (1997). Control in Act-R and Soar. In M. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 343-348). Hillsdale, NJ: Lawrence Erlbaum Associates.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review, 114*, 1-37.

Kanerva, P. (1988). *Sparse Distributed Memory*. Cambridge MA: The MIT Press.

Kanerva, P. (1993). Sparse Distributed Memory and related models. In M. H. Hassoun (Ed.), *Associative Neural Memories: Theory and Implementation* (pp. 50-76). New York, NY: Oxford University Press.

Kanerva, P. (1994). The binary spatter code for encoding concepts at many levels. In M. Marinaro & P. Morasso (Eds.), *ICANN '94: Proceedings of International Conference on Artificial Neural Networks* (Vol. 1, pp. 226–229). London, UK: Springer-Verlag.

Kanerva, P. (2009). Hyperdimensional Computing: An Introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation, 1*(2), 139-159.

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. In P. Wang, B. Goertzel & S. Franklin (Eds.), *Artificial General Intelligence 2008*. Amsterdam: IOS Press.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence, 33*, 1–64.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research, 10*(2), 141-160. doi: doi: 10.1016/j.cogsys.2006.07.004

Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., & Thompson, K. (1991). A design for the ICARUS architecture. *ACM SIGART Bulletin, 2*, 104–109.

Lebiere, C., & Anderson, J. R. (1993). A Connectionist Implementation of the ACT-R Production System *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society* (pp. 635–640). Hillsdalle NJ: Erlbaum.

McCall, R. (2014). *Fundamental Motivation and Perception for a Systems-Level Cognitive Architecture.* (PhD), University of Memphis, Memphis, TN USA.

Mendes, M., Coimbra, A., & Crisóstomo, M. (2009). Assessing a Sparse Distributed Memory Using Different Encoding Methods. *Proceedings of the World Congress on Engineering, 1*, 1-6.

Meng, H., Appiah, K., Hunter, A., Yue, S., Hobden, M., Priestley, N., . . . Pettit, C. (2009). *A modified sparse distributed memory model for extracting clean patterns from noisy inputs.* Paper presented at the International Joint Conference on Neural Networks (IJCNN), Atlanta, GA, USA.

Plate, T. A. (1995). Holographic Reduced Representations. *IEEE Transactions on Neural Networks, 6*(3), 623-641.

Plate, T. A. (2003). *Holographic Reduced Representation: distributed representation of cognitive structure*. Stanford: CSLI.

Ramamurthy, U., Baars, B. J., D'Mello, S. K., & Franklin, S. (2006). *LIDA: A Working Model of Cognition.* Paper presented at the 7th International Conference on Cognitive Modeling, Trieste, Italy.

Ramamurthy, U., D'Mello, S. K., & Franklin, S. (2006). Realizing Forgetting in a Modified Sparse Distributed Memory System. In C. Schunn & S. Lane (Eds.), *Proceedings of the 28th Annual Conference of the Cognitive Science Society* (pp. 1992–1997). Mahwah, NJ: Lawrence Erlbaum Associates.

Ramamurthy, U., & Franklin, S. (2011). *Memory Systems for Cognitive Agents*. Paper presented at the Proceedings of Human Memory for Artificial Agents Symposium at the Artificial Intelligence and Simulation of Behavior Convention (AISB'11), University of York, UK.

Sahlgren, M. (2005). *An Introduction to Random Indexing.* Paper presented at the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005, Copenhagen, Denmark.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007). Robust Object Recognition with Cortex-Like Mechanisms. *IEEE Transations on Pattern Analysis and Machine Intelligence, 29*(3), 411-426.

Snaider, J. (2012). *Integer Sparse Distributed Memory and Modular Composite Representation. PhD Thesis.* (PhD), University of Memphis, Memphis, TN.

Snaider, J., & Franklin, S. (2011). *Extended Sparse Distributed Memory.* Paper presented at the Biological Inspired Cognitive Architectures 2011, Washington DC.

Snaider, J., & Franklin, S. (2012a). Extended Sparse Distributed Memory and Sequence Storage. *Cognitive Computation, 4*(2), 172-180. doi: 10.1007/s12559-012-9125-8

Snaider, J., & Franklin, S. (2012b). *Integer Sparse Distributed Memory.* Paper presented at the The 25th Florida Artificial Intelligence Research Society Conference FLAIRS-25, Marco Island, FL.

Snaider, J., & Franklin, S. (in press). Modular Composite Representation. *Cognitive Computation*. doi: 10.1007/s12559-013-9243-y

Snaider, J., McCall, R., & Franklin, S. (2011). *The LIDA Framework as a General Tool for AGI.* Paper presented at the The Fourth Conference on Artificial General Intelligence, Mountain View, CA.

Snaider, J., McCall, R., & Franklin, S. (2012). Time Production and Representation in a Conceptual and Computational Cognitive Model. *Cognitive Systems Research, 13*(1), 59-71.

Stewart, T. C., & Eliasmith, C. (2011). *Neural Planning and Reasoning Using the Synaptic Connections of the Basal Ganglia and Thalamus.* Paper presented at the Biologically Inspired Cognitive Architectures 2011, Whashington, DC.

Sun, R. (1997). An agent architecture for on-line learning of procedural and declarative knowledge *Proceedings of the International Conference on Neural Information Processing (ICONIP'97): Progress in Connectionist-Based Information Systems* (pp. 766–769). Singapore: Springer Verlag.

Turney, P. D., & Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research, 37*, 141-188.

Ustun, V., Rosenbloom, P. S., Sagae, K., & Demski, A. (2014). *Distributed vector representations of words in the Sigma cognitive architecture.* Paper presented at the 7th Annual Conference on Artificial General Intelligence, Quebec City, Canada.

Winston, P. H. (1992). *Artificial Intelligence* (3rd ed.). Boston, MA: Addison Wesley.