Vector space architecture for emergent interoperability of systems by learning from demonstration

Blerim Emruli*, Fredrik Sandin, Jerker Delsing

EISLAB, Luleå University of Technology, S-97187 Luleå, Sweden

Abstract

The rapid integration of physical systems with cyberspace infrastructure, the so-called Internet of Things, is likely to have a significant effect on how people interact with the physical environment and design information and communication systems. Internet-connected systems are expected to vastly outnumber people on the planet in the near future, leading to grand challenges in software engineering and automation in application domains involving complex and evolving systems. Several decades of artificial intelligence research suggests that conventional approaches to making such systems interoperable using handcrafted "semantic" descriptions of services and information are difficult to apply. In this paper we outline a bioinspired learning approach to creating interoperable systems, which does not require handcrafted semantic descriptions and rules. Instead, the idea is that a functioning system (of systems) can emerge from an initial pseudorandom state through learning from examples, provided that each component conforms to a set of information coding rules. We combine a binary vector symbolic architecture (VSA) with an associative memory known as sparse distributed memory (SDM) to model context-dependent prediction by learning from examples. We present simulation results demonstrating that the proposed architecture can enable system interoperability by learning, for example by human demonstration.

Keywords: communications, interoperability, learning, vector symbolic architecture, sparse distributed memory, artificial intelligence, system of systems

1. Introduction

The increasing number of physical objects and devices connected to the Internet has created a demand for methodologies that enable vendor-independent communication and collaboration between systems (Papazoglou, 2003; Huhns

*Corresponding author

Email addresses: blerim.emruli@ltu.se (Blerim Emruli), fredrik.sandin@ltu.se (Fredrik Sandin), jerker.delsing@ltu.se (Jerker Delsing)

and Singh, 2005; Bohn et al., 2006; Souza et al., 2008; Karnouskos et al., 2010; Baresi et al., 2013) and within System of Systems (SoS), which are composed of parts that are complex functioning systems in their own right (Maier, 1998; Fisher, 2006). The adoption of standardized interfaces and protocols is a key aspect of research and development in this context, which enables communication of information in heterogeneous systems. For example, by using common Internet protocols and service-oriented interfaces any developer with the appropriate access credentials can use or reuse functionality developed by others. This software design philosophy is not new (Vinoski, 1997; Henning and Vinoski, 1999) and some ideas date back to the 1970s (Ackoff, 1971; McIlroy, 1976), but it tends to be transformed and applied to a broader range of systems today, for example Cyber–Physical Systems based on Web services and the Internet of Things (IoT) in general (Souza et al., 2008; Karnouskos et al., 2010; Baresi et al., 2013).

Interoperability is a central aspect of SoS and integrated systems, which can be defined as, "The ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to a shared operational semantics in order to achieve a specified purpose in a given context." (Carney et al., 2005). Presently, the development of interoperable systems is a matter of handcrafting such a *shared operational semantics*, e.g., in the form of metadata, standardized protocols, service descriptions and orchestrated system functionality. In particular, metadata play a key role in the design of components of heterogeneous systems (Sheth, 1999; Obrst, 2003; Baresi et al., 2013) and can for example take the form of a resource description framework (RDF) graph including semantic identifiers like keywords and references, see Figure 1 for an example. This is a significant improvement compared with non-standardized and proprietary approaches to communication, but this approach also has limitations.

In the conventional approach outlined above humans artificially create se-

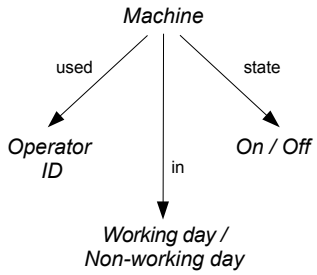

Figure 1: An example of an RDF graph representing a basic semantic model of a machine, e.g., an electric motor. Each arrow (edge) is an RDF statement, e.g, the name at the top is the statement's subject (*Machine*). The names at the end of each arrow are the statement's objects (*Operator ID, Working day/Non-working day* and *On/Off*) and the names that label the arrows are the predicates (used, in and state). Semantic descriptions of this type enable design and integration of heterogenous systems (Baresi et al., 2013), but are ambiguous and can be misinterpreted (Fisher, 2006; Ekbia, 2010; Guns, 2013).

mantic descriptions. Such definitions are complex and involve assumptions, for example, concerning the interpretation of words and sentences written in human language. Therefore, semantic descriptions can be misinterpreted (Fisher, 2006, Section 3.7). Several decades of artificial intelligence research suggests (Ekbia, 2010; Guns, 2013) that it is difficult to make machines that automatically and correctly make use of such non-grounded information (Harnad, 1990; Barsalou, 2008). In general, the conventional approach to designing large-scale heterogeneous systems appears to have the following limitations:

- The use of semantic descriptors is error-prone and difficult to automate.
- Experts need to define the interfaces of components and integrate systems in minute detail, which is costly.
- The software design effort increases with domain complexity, which limits scalability.
- The software maintenance effort increases when systems evolve, which limits applicability.

Therefore, alternative approaches to system design and communication, which can enable components in heterogenous systems to automatically exchange information in a useful manner are needed.

There are several remarkable examples of interoperability in biological systems and interconnected biological and technological systems, resulting from learning rather than pre-defined protocols. For example, brains can learn to interact with robotic and virtual arms via implanted electrode arrays (O'Doherty et al., 2011; Ifft et al., 2013) and to see after surgical rewiring of visual stimuli from the primary visual cortex to auditory cortex (von Melchner et al., 2000); Cochlear implants enable deaf and severely hearing-impaired persons to hear and have conversations (Wilson et al., 1991; Wilson, 2013; Mudry and Mills, 2013); Electrotactile and vibrotactile image projection enable blind and severely vision-impaired persons to see (Bach-y-Rita et al., 1969; Kaczmarek et al., 1991; Bach-y-Rita et al., 2003) and a vibrotactile magnetic compass can create a sense of direction (Tsukada and Yasumura, 2004). While it would be an interesting initiative to explore how first principles of neural circuits can drive self-organization and interoperability of systems, that approach is ambitious and beyond the scope of this work.

In this paper, we outline a bioinspired mathematical approach to the design of interoperable systems. The core idea is that interoperability should emerge by learning, starting from an initial pseudorandom state, rather than by handcrafted semantic descriptions and rules (Sheth, 1999; Obrst, 2003; Baresi et al., 2013). This way we can, at least in principle, avoid some of the difficulties facing conventional approaches. We use a vector space model known as a vector symbolic architecture (VSA) (Gayler, 2003) and an associative memory model known as sparse distributed memory (SDM) (Kanerva, 1988) for learning and predicting state transitions in interconneted systems. A VSA uses distributed representations (Hinton et al., 1986; Hinton, 1990) to approximate information,

which is motivated by information coding in sensory systems of animals, different from the localist representations used in most conventional technologies. Using a VSA it is possible to encode low-level features and categories of stimuli, as well as higher-order compositional structures needed to represent concepts (Zentall et al., 2008, 2014) and sensory–motor functions. A VSA makes it possible to approximate functions involving both structure and content (or syntax and semantics) in a systematic and fairly simple manner using operations in a high-dimensional space, which is not easily replicated by other models that are based on distributed representations.

There are several types of VSAs (Plate, 1995; Kanerva, 1996; Gayler, 1998; Kanerva, 2009; Rachkovskij and Kussul, 2001; Gallant and Okaywe, 2013) which differ in the mathematical details. Here we adopt an architecture known as Binary Spatter Codes (BSCs) (Kanerva, 1996). The motivation for that choice is threefold: (1) Information coded in BSCs can naturally be stored in and retrieved from an SDM, enabling transparent learning of multiple concepts, relationships between concepts, and analogies (Emruli and Sandin, 2014; Emruli et al., 2013); (2) Unlike digital representations of integers, floating point numbers, strings and so forth, the interpretation of a binary value in a distributed representation involves a minimum of assumptions (a bit is simply 0 or 1); (3) Spikes in neural systems have a stereotyped form, which for example are converted to binary vectors in brain–machine interfaces (O'Doherty et al., 2011; Ifft et al., 2013). We propose that semantic descriptors of services and information can be replaced with distributed representations of sensory–motor functions grounded in sensory projections (Harnad, 1990; Barsalou, 2008), and randomly initialized structures that become gradually associated in a system-wide memory through learning (Kanerva, 1988; Emruli and Sandin, 2014). For related discussions see Sutton and Whitehead (1993); Plate (1995); Eliasmith and Thagard (2001); Neumann (2002). In the following sections of this paper, we present the proposed communication architecture, the simulation results demonstrating that the approach enables interoperability in the form of learning context-dependent prediction of state transitions, and related work.

2. Model

The proposed communication architecture is illustrated in Figure 2. This architecture enables automatic prediction of learned context-dependent state transitions, which can be exploited by individual systems for automation purposes, anomaly detection and so forth. A *state transition* is defined as the transition from one *state* of the working memory (the source) to the next state (the target), and the context is defined by items in working memory that are constant during the state transition. The long-term memory learns to predict the target state from the source state by heteroassociation of distributed representations, including analogical mapping (Gentner and Smith, 2013; Emruli and Sandin, 2014) which enables generalization when source–target transitions are invariant with respect to some particular substructure of the source and target states. A *predicted state*, Φ_c, is broadcast to all systems, for example using a

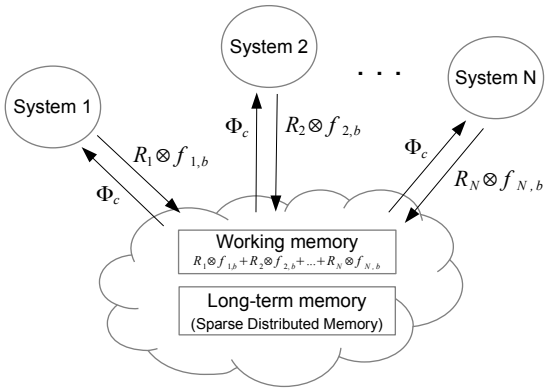

Figure 2: Communication architecture including a VSA working memory and an SDM-based long-term memory, which enable learning and prediction of context-dependent state transitions.

conventional digital communication system. The broadcast prediction can be decoded by individual systems using the VSA methods that are described below.

Each system, a, is given a unique VSA *role* in the form of a high-dimensional pseudorandom binary vector, R_a, $a \in [1,\ N]$, which identifies the system and is used to encode and decode information communicated using VSA binding and probing operators, respectively (see the following subsection for details). As the name suggests, R_a defines the role of each particular system, but not in a shared operational semantics fashion but an arbitrary and unique one. Roles can be created with a pseudorandom number generator when a particular system is manufactured, i.e., they could be hardcoded identifiers unique for each system. It is the purpose of the long-term memory to learn how the roles of different systems are implicitly related to each other so that meaningful predictions can be made. In addition to roles, systems can have some kind of sensory–motor function, which provides the examples that are learned. The *internal state* of an individual system is represented in the form of VSA *fillers*, $f_{a,b}$, which can be compositional and grounded structures (Harnad, 1990; Barsalou, 2008) that include sensor and/or motor information. Before transmitting information using the architecture, a system a must bind the filler, $f_{a,b}$, to its hardcoded role, R_a. The vector resulting from the binding process in each system, $R_a \otimes f_{a,b}$, are transmitted to the architecture, which superposes received vectors in the working memory, $R_1 \otimes f_{1,b} + R_2 \otimes f_{2,b} + ... + R_N \otimes f_{N,b}$ (this superposition is shown in Figure 2). When the working memory is updated, the state transition is written to the long-term memory, and the long-term memory predicts the subsequent state of the working memory, Φ_c, which is broadcast to all systems. Each system can then decode the broadcast predictions, Φ_c, received from the long-term memory using the role, R_a, and the probing operation. The resulting VSA vector will be nearly orthogonal to all vectors represented within system

a if Φ_c does not involve the role R_a, i.e., if Φ_c is not based on learned state transitions involving vectors communicated by system a. In that case system a finds no use of Φ_c and waits for the next prediction, Φ_{c+1}.

The mechanism outlined above is simple compared to the complex learning circuits that are at work in brain–machine interfaces, which are only partially known and understood. In particular, we do not address dynamical aspects in this work, only basic context-dependent prediction of state transitions. Nevertheless, the proposed architecture includes the basic concepts needed to enable system interoperability by learning, forming a novel conceptual basis for further research and development.

Here the term *system* is general and refers to any kind of system that conforms to the architecture, e.g., sensors, actuators, embedded systems, appliances, or mobile apps. In principle the approach taken here is relevant also for large-scale systems-of-systems (Maier, 1998) like energy systems or a national emergency system, where each component is a complex and independently functioning system by itself (e.g., a power plant or a police station). Here we consider moderately sized systems, like a factory assembly line or a smart home. The working memory is subject to a capacity limit (Kanerva, 1996; Gallant and Okaywe, 2013) that requires implementation of an attention-like mechanism when too many systems are involved (cf. Miller's law (Miller, 1956)). For simplicity, we assume that the number of systems that communicate simultaneously is low so that the working memory is not saturated. In the next section we demonstrate that this architecture enables learning and context-dependent prediction of state transitions without reference to semantic descriptions and manual system integration, provided that each system conforms to the general information coding rules of the VSA that are outlined below. Systems can otherwise be designed independently of each other. The architecture also supports autonomous relearning when a constituent system is replaced.

Next we present some details and characteristics of the working and long-term memories. Refer to some of the numerous papers and books in the area for further technical information, for instance Kanerva (1988); Plate (2003); Gayler (2003); Kanerva (2009); Rachkovskij and Kussul (2001); Gallant and Okaywe (2013); Emruli and Sandin (2014).

2.1. Working memory

The working memory is implemented in the form of a binary VSA, which is a connectionist model that uses high-dimensional vectors of the same, fixed dimensionality to represent and manipulate entities, whether atomic or composite (Plate, 1995; Kanerva, 1996; Gayler, 2003). Compositional structures representing objects, properties and relations are points in a high-dimensional space of dimensionality ≥ 1000. There are standard operations in a VSA that allow representations to be composed, decomposed, probed for similarity and transformed in various ways. These operations do not have to be learned as they are automatic consequences of the mathematical structure of the VSA. The absence of fields, the potential to combine structure and content, and the ability to operate on representations without first decomposing them means that a VSA

implements a form of holistic processing (Plate, 1995; Kanerva, 1997; Neumann, 2002).

Well-known examples of VSAs are the Binary Spatter Codes (BSCs) (Kanerva, 1996) and the Holographic Reduced Representations (HRRs) (Plate, 1995). All VSAs provide a product-like binding operator and a sum-like superposition operator and, optionally, other operators such as permutation of vector components for sequence coding. The type of vector elements and the mathematical definitions of the operators vary between different VSAs. For example, in HRRs the vector elements are real or complex numbers, binding is implemented as circular convolution, and superposition is implemented as element-wise addition. Whereas for BSCs, which are used here, the vector elements are binary, *binding* is defined as the element-wise binary XOR operation, and *superposition* of multiple vectors x_k is defined as an element-wise binary average $\langle \sum_{k=1}^{n} x_{k,i} \rangle = \Theta\,(1/n \sum_{k=1}^{n} x_{k,i})$, where $\Theta(x) = \{1 \text{ for } x > 0.5,\ 0 \text{ for } x < 0.5\,,$ random otherwise$\}$. For completeness, we mention that there is one more important VSA operation, called probing. For BSCs *probing* is defined in the same way as the binding operation, and it is used to extract known parts of a compositional structure (which may themselves be compositional structures). Probing can be compared to accessing a field in a record, or a slot in a frame, or a node in a graph.

When compositional structures are combined and manipulated using VSAs the results are approximate, in the sense that the output vectors are close to but not exactly the same as the vectors representing the original structures. The notion of distance is key for VSAs, since it reflects the similarity of both structure and content. In BSCs, similar structures have low Hamming distance, δ (normalized with respect to the dimensionality of the vectors), and high Pearson correlation coefficient, ρ, where $\rho = 1 - 2\delta$. For example, when creating a superposition of binary vectors, the expected similarity between a binary superposition of K vectors and one of the superposed vectors is given by the binomial function $\delta = 0.5 - 2^{-K}\mathrm{Bin}[K-1, (K-1)/2]$, which is approximately $\delta \simeq 0.5 - 0.4/\sqrt{K - 0.44}$ (Kanerva, 1997). The standard deviation of distance is $\sigma = \sqrt{\delta(1-\delta)/\mathcal{D}}$, where $\mathcal{D}$ is the dimensionality of the vectors. This means that the standard deviation becomes insignificant when the dimensionality of the representation is increased. We return to this issue below.

Defining similarity in this way is different compared to conventional approaches to data representation, where the structure and the content are separated. The comparison of structured representations, i.e., in the form of graphs, is an NP-complete problem (Gentner and Forbus, 2011). This is avoided in VSAs by the introduction of fixed-size approximate representations, in combination with some type of long-term memory used to "clean up" the approximate representations in short-term memory.

Figure 3 shows how a binary superposition correlates with its superposed parts. In Figure 3a the mean of the distance increases with the number of parts, but it is not affected by the dimensionality. The standard deviation of distance, σ, decreases with increasing dimensionality of the vectors. Figure 3b shows that the standard deviation of distance decreases with dimensionality,

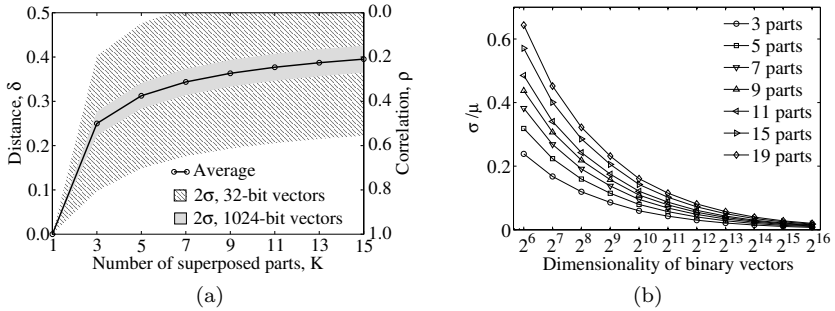

Figure 3: (a) Distance and correlation between a binary superposition of vectors and one of the superposed vectors, versus the number of superposed vectors for different dimensionalities. The solid line indicate the mean and the shaded areas denote two standard deviations, or 95% confidence intervals. (b) Relative standard deviation, σ/μ, of the correlation between a binary superposition of vectors and one of the superposed vectors versus the dimensionality of the vectors.

while the average distance, μ, decreases with the number of parts. Therefore, the maximum number of parts that can be superposed and processed as a whole increases with dimensionality but decreases with the number of items in memory. In addition, the maximum number of parts that can be superposed and distinguished from a fixed number of random vectors increases linearly (at first order) with the dimensionality (Gallant and Okaywe, 2013). This creates a fundamental link between resources and the number of superposed parts that can be simultaneously processed, which qualitatively mimics the capacity limit of human working memory and an empirical result called Miller's law[1] (Kanerva, 1997; Miller, 1956). The integration of a long-term memory is motivated also from this perspective, because it is needed to "clean up" noise in the working-memory representations. When a comparable number of superposed parts are processed with BSCs the standard deviation is negligible at a dimensionality of about 10^4 (Kanerva, 1997; Emruli and Sandin, 2014; Emruli et al., 2013), which coincidentally is comparable to the number of synapses on cortical and hippocampal pyramidal cells. For a more detailed analysis of noise see Gallant and Okaywe (2013).

The limitation to simultaneously process a certain number of superposed vectors applies both to prediction with spatial information and to prediction with temporal information (approximation of first- and higher-order Markov processes). Predictions beyond this limit can be made only with sequential queries of the long-term memory, or by the formation of more abstract composite structures and associated long-term memories for source–target mappings. In applications where the number of simultaneously active superpositions gener-

[1]Miller's "magic number seven" has been subject of much debate. It is not within the scope of this study to argue for the particular number of items that can be held and simultaneously processed in working memory.

ated by the sensory system exceeds this limit, attention and binding mechanisms are needed to select and bind objects of interest before they enter working memory. Attention is beyond the scope of this work but is actively studied in various fields, mostly in the perspective of biological and machine vision systems (Treisman, 1998; Reynolds and Desimone, 1999; Itti and Koch, 2001; Wichert, 2011; Allen et al., 2012).

2.2. Long-term memory

Sparse distributed memory (SDM) is a mathematical model of associative and episodic memory, which is thoroughly described in the seminal book by Kanerva (1988). Kanerva developed the SDM inspired by some characteristics of human long-term memory. The SDM is based on the same idea as VSAs that similar or related concepts are represented by nearby points in a high-dimensional space (Kanerva, 1988, 1993). The SDM have been successfully used in numerous applications since the 1990s, e.g., pattern recognition (Hely et al., 1997; Meng et al., 2009), predictive analytics (Rogers, 1989, 1990), robot navigation (Rao and Fuentes, 1998; Jockel et al., 2009), approximation of Bayesian inference in a fashion similar to Monte Carlo importance sampling (Anderson, 1989; Abbott et al., 2013), and biologically inspired cognitive architectures (Rachkovskij et al., 2013; Franklin et al., 2014).

Here we briefly summarize the SDM, the interested reader is referred to Emruli and Sandin (2014) for a more detailed description of how the SDM model is implemented. The SDM consists of two parts: a binary address matrix, A, and an integer content matrix, C. These two matrices are initialized as follows; The matrix A is populated randomly with zeros and ones with equal probability, and the matrix C is initialized with zeros. The rows of the matrix A are referred to as address vectors, and the rows of the matrix C are counter vectors. There is a one-to-one link between address vectors and counter vectors, so that an activated address vector corresponds to one specific activate counter vector. The address and counter vectors have dimensionality $\mathcal{D}$. The number of address vectors (or counter vectors) defines the size of the memory, S. The parameters of the long-term memory that is integrated in the communication architecture are listed in Table 1.

The combination of a VSA with the idea of *mapping vectors* (Plate, 1995; Kanerva, 2000; Neumann, 2002) enables learning of multiple concepts, relationships between concepts, and analogical mappings in a coherent way (Emruli and Sandin, 2014). To briefly outline the idea of mapping vectors, consider a

Table 1: Summary of exogenous parameters of the long-term memory, which have to be defined by the architect. These are the only parameters of the architecture.

Expression	Description
$\mathcal{D}$	Dimensionality of the binary vector symbols.
S	Size of the memory, number of address (or counter vectors).
χ	Average probability for activating an SDM location.

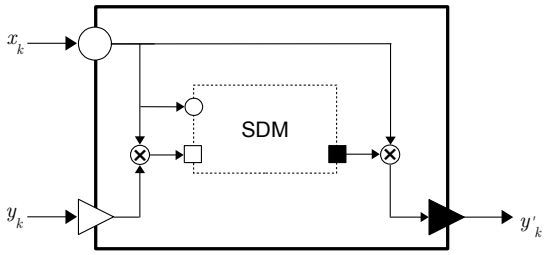

Figure 4: The analogical mapping unit (AMU) which learns state transitions of the type $x_k \rightarrow y_k$ from examples and enables approximate context-dependent prediction of y'_k for a novel input x_k (Emruli and Sandin, 2014).

mapping of a source state, x, to a target state, y. Kanerva (2000) suggests that a mapping vector, M, can be defined as $M = x \otimes y$, and that superposition of mapping vectors can be used to correctly map novel compositional structures. This is the core generalization mechanism that is implemented in the Analogical Mapping Unit (AMU), see Figure 4. The XOR-based binding operator commutes and is its own inverse, so it follows that $M \otimes x = y$ and $M \otimes y = x$, and $x \otimes x = \mathbb{I}$, which implies that mappings are bidirectional. The problem of bidirectionality of the mapping is important to enable learning and prediction of sequences. The AMU solves this problem by breaking the commutative property of the mapping vectors with an SDM.

Similar to the SDM, the AMU takes two binary input vectors and generates one binary output vector, but with a different result and interpretation compared to the SDM. Given a source state, x_k, and the target state, y_k, the AMU learns to predict y_k from x_k by adding the corresponding mapping vector, $x_k \otimes y_k$, to the activated SDM locations. This process resembles the concept of Hebbian learning and is a one-shot (single-cycle) learning process. When the source, x_k of a learned example, $x_k \rightarrow y_k$, is presented to the AMU an approximation of the learned target, y_k, is calculated. If a novel source, x'_k, that is structurally and content-wise similar to learned examples is presented, the AMU calculates an approximate analogical structure y'_k. The long-term memory defined in terms of an AMU has three important properties: 1) Mapping vectors of unrelated states, x_k, are encoded in different storage locations of the long-term memory, which makes it possible to automatically learn multiple unrelated mappings. 2) Similar mapping examples are automatically superposed in nearby storage locations, which is necessary for generalization. 3) The SDM activation mechanism breaks the commutative property of the mapping vectors, $x_k \otimes y_k$, so that the reversed mapping $y_k \rightarrow x_k$ is unlikely as long as a limited subset of the SDM locations are activated (Emruli et al., 2013). These are the key points allowing the communication architecture to enable system interoperability by learning, as shown by the simulation experiments below.

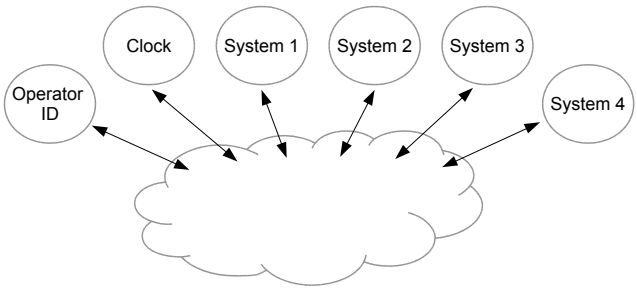

Figure 5: A hypothetical automation system composed of multiple constituent systems serving different purposes: (1) a system for identification of human operators, (2) a system that provides a timestamp, and (3) four systems that turn on or off depending on the instructions of the human operators. The communication architecture is outlined in Figure 2.

3. Simulation experiments

In the simulations presented below we investigate the interoperability learning rate and prediction accuracy of the communication architecture described in Section 2. We simulate the architecture in a hypothetical automation scenario, for example, a factory assembly line or smart home, which is composed of the six systems illustrated in Figure 5. Four systems have sensory–motor functions, which can be used by human operator to set the internal states of these systems to either "on" or "off", while the other two systems provide context information in the form of the current time and the ID of a human operator (e.g., through an RFID tag for localization). The systems considered here are analogous to the system illustrated in Figure 1. The specific functions of these systems emerge from the instructions given by two human operators, Alice and Bob, who interact with the sensory–motor functions of the four systems to achieve a particular goal. The instructions of Alice and Bob are the same, but they are different during working and non-working days, thereby making the behaviour of systems context dependent.

The systems which turn on or off are given different unique roles (not to be confused with the size of the long-term memory, S) S_1, S_2, S_3 and, S_4, and the systems for identification of human operators and timing also have unique roles, ID and T, respectively. See Table 2 for a definition of roles and fillers used in the simulation experiments. The importance of assigning different unique roles for each system is explained in Section 2. Note that the fillers define unique representations of the internal states of a system (e.g., On_k, Off_k), which are not shared with other systems — there is no shared operational semantics defining the interpretation of that information. T_w and T_{nw} represent the fillers for working and non-working days, respectively.

We use state diagrams to illustrate the transitions between states of the communication architecture, which result from the interaction between systems and human operators. States are illustrated as circles, and a mapping between states as a directed line that connects the circles. Figure 6 illustrates state tran-

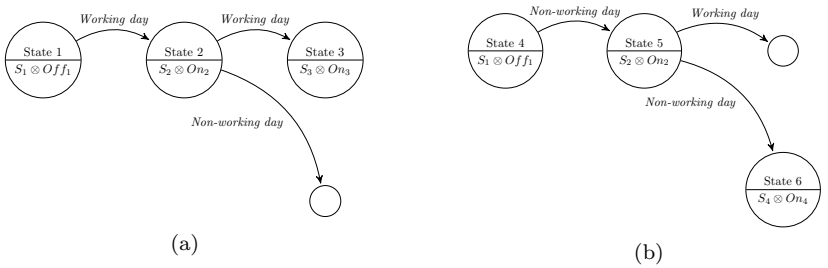

Figure 6: The communication architecture learns to predict the next state based on Alice's and Bob's instructions, which are compatible but vary between (a) working and (b) non-working days. See Table 3 and related text for more details concerning the encoding of states.

sitions when Alice is instructing the systems during working and non-working days. For example, State 1, 2 and 3 correspond to Alice turning off S_1, turning on S_2 and then turning on S_3 (working days) or S_4 (non-working days). See Table 3 for the encoding (role-filler bindings) of the corresponding states. States corresponding to instructions from Bob follow the same logic, but are excluded from the table for clarity. As Figure 6 illustrates, we chose to simulate a basic transitive relationship between states, in which State 1 is followed by State 2, and State 2 is followed by State 3, without State 3 being followed by State 2 (the same logic applies for Figure 6b). In principle, an interoperation relationship does not have to be transitive because in the composition of states we expect to see different kinds of interoperability (e.g., transitive, commutative, or reflexive). In a real-world implementation of the architecture, simultaneously active systems transmit their bound role–filler vectors to the architecture, which superposes all received vectors in the working memory.

In order to learn a generic mapping that will correctly predict state transitions of the communication architecture, for each transition between states the corresponding mapping vector is stored in the long-term memory. In the simulations we assume that Alice or Bob instruct the architecture on a daily basis, thereby providing one example of each state transition per day. Therefore, examples that involve working day states are presented 5 times per week, while mappings of non-working day states are presented 2 times per week.

The correlation between the prediction of the communication architecture and the correct target state varies from one simulation experiment to the next because all roles, fillers and memory components are randomly initialized. The

Table 2: Roles and fillers are uniquely defined for each system, meaning that there is no shared operational semantics defined in the implementation of the architecture.

Roles	Notation	Fillers
$Operator\ ID$	ID	$Alice, Bob$
$Systems$	S_1, S_2, S_3, S_4	On_k, Off_k
$Time$	T	T_w, T_{nw}

distribution functions for these variations have a non-trivial structure. It is difficult to translate average correlation coefficients and variances into tail probabilities of error. Therefore, we estimate the *probability of error* numerically from the error rate in the simulation experiments. An error occurs when the broadcast output of the communication architecture, Φ_c, is incorrectly interpreted by at least one of the systems. The communication architecture learns from few examples, however, we repeat the simulation experiments many times until the 95% confidence interval of the probability of error becomes less than one tenth of the average probability of error. We estimate the confidence interval for the probability of error with the Agresti–xCoull interval (Agresti and Coull, 1998), which is similar to the pragmatic rule "add 2 successes and 2 failures".

After learning the transition between different states of the architecture for one simulated week of transitions, we test the ability of the proposed communication architecture to predict the next state by presenting a learned state, $x_k = \langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_1 \otimes Off_1 \rangle$. The corresponding output vector, y'_k, is compared (in terms of the element-wise correlation coefficient) with four possible alternative solutions, see Table 4. The first alternative is the correct solution, whereas the other alternatives represent "meaningful" incorrect targets, which are incorrect compositions of relevant roles and fillers. It is possible to construct many more incorrect targets that do not involve the correct roles and fillers, but these vectors can be safely excluded from the test. The result of this simulation is presented in Figure 7. The parameters of the long-term memory are set to $\mathcal{D} = 8192$, $S = 100$ and $\chi = 0.1$; which means that the long-term memory has 100 locations out of which 10 are activated in each storage and retrieval operation. With more training examples, resulting from further use of the system, the accuracy of the communication architecture increases. This is demonstrated in the next simulation experiment.

A second simulation experiment is carried out to test the ability of the proposed communication architecture to relearn when one system, S_2, is replaced with a new system that has a unique role and set of fillers (e.g. S_2 is replaced because it is faulty). Initially the new system, S_x, is not correctly integrated, see Figure 8. In this experiment we investigate the accuracy and learning rate of the architecture when the source state stays the same and the target state is modified, which is more challenging than vice versa. The source state is the

Table 3: States of the architecture when Alice is instructing the systems during working and non-working days.

State	Encoding
State 1	$\langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_1 \otimes Off_1 \rangle$
State 2	$\langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_2 \otimes On_2 \rangle$
State 3	$\langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_3 \otimes On_3 \rangle$
State 4	$\langle ID \otimes Alice \ + \ T \otimes T_{nw} \ + \ S_1 \otimes Off_1 \rangle$
State 5	$\langle ID \otimes Alice \ + \ T \otimes T_{nw} \ + \ S_2 \otimes On_2 \rangle$
State 6	$\langle ID \otimes Alice \ + \ T \otimes T_{nw} \ + \ S_4 \otimes On_4 \rangle$

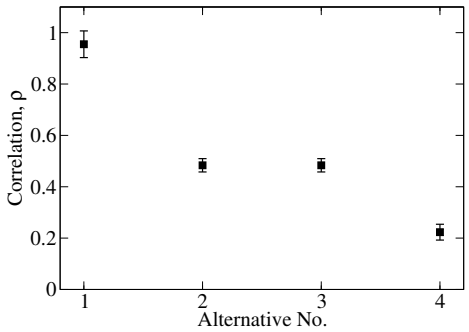

Figure 7: The correlation, ρ, between the state predicted by the communication architecture and four alternative states listed in Table 4. Error bars denote standard deviations. The parameters of the long-term memory are set to $\mathcal{D} = 8192$, $S = 100$ and $\chi = 0.1$. Alternative number one has the highest correlation, $\rho = 0.96 \pm 0.052$. The probability that the communication architecture predicts an incorrect alternative is $1 \cdot 10^{-5} \pm 1 \cdot 10^{-5}$ at 95% confidence level.

same as above, $x_k = \langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_1 \otimes Off_1 \rangle$, while the target is $\langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_x \otimes On_x \rangle$, instead of $\langle ID \otimes Alice + T \otimes T_w + S_2 \otimes On_2 \rangle$.

We simulate two cases, where the architecture is trained with the old system, S_2, for one and five weeks, respectively, and then we train the architecture for fifteen weeks with the new system, S_x. The parameters of the long-term memory are the same as in the previous experiment, $\mathcal{D} = 8192$, $S = 100$ and $\chi = 0.1$. The result is presented in Figure 9a, which illustrates that in this case it takes the communication architecture almost the same time as it has been operating in the old setup to adapt and correctly integrate the new system, S_x. This effect is natural, since it reflects a key property of a long-term memory, which is storing information for long periods of time. However, the linear scaling of learning and relearing time is valid only for short learning periods because variables of the long-term memory saturate after some time. Therefore, the maximum relearning time is set by the rate of training examples and long-term memory saturation. Another interesting result is that the probability of error converges and does not improve beyond a certain number of training examples. Next, we set $S = 200$, which implies that the size of the long-term

Table 4: Alternative states used when testing the ability of the communication architecture to predict the correct alternative, when a learned example, $x_k = \langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_1 \otimes Off_1 \rangle$, is presented.

Alternative	Encoding of corresponding states
Alternative 1	$\langle ID \otimes Alice \ + \ T \otimes T_w \ + \ S_2 \otimes On_2 \rangle$
Alternative 2	$\langle ID \otimes Alice \ + \ T \otimes T_{nw} \ + \ S_2 \otimes On_2 \rangle$
Alternative 3	$\langle ID \otimes Bob \ + \ T \otimes T_w \ + \ S_2 \otimes On_2 \rangle$
Alternative 4	$\langle ID \otimes Bob \ + \ T \otimes T_{nw} \ + \ S_2 \otimes On_2 \rangle$

memory is doubled. In Figure 9b it is shown that the the probability of error decreases with larger S and does not converge as quickly as in Figure 9a. We also investigate the effect of different values of χ and present results for $\chi = 0.1$ and $\chi = 0.3$. That is, when 10% and 30% of the long-term memory is activated in each prediction. Figure 9c shows that a higher χ leads to a higher learning rate. These simulation experiments show that the architecture can be fairly accurate using a small long-term memory, with a probability of error reaching approximately 1.5%. Furthermore, the accuracy is expected to improve when the communication architecture continues to operate and learn from additional examples.

4. Related work

In the literature, among many other approaches employed, service-oriented architectures (SOA), multi-agent systems and bioinspired computing seem to be the most common approaches to address interoperability and integration problems in heterogeneous devices and systems. Several projects focus on the SOA approach. SIRENA (Bohn et al., 2006), SOCRADES (Souza et al., 2008), and AESOP (Karnouskos et al., 2010) are a few prominent examples. Some advanced implementations in these projects feature event-driven messaging, and to some extent support processing of contextual information. However, as mentioned above a difficulty of this approach is that experts are needed for the design, deployment and maintenance, which is costly and limits scalability. In addition, this approach is error prone because standards and semantic descriptions can be misinterpreted.

Multi-agent systems is another approach that has been used to build middleware infrastructure for evolvable and dynamic environments. The core is an organizational meta-model based on multi-agent systems that aim to handle a hierarchical structure. This approach usually provides a step-by-step modeling guide and tools for all the software development phases ranging from requirements specification to implementation. Nevertheless, this process is different from building traditional software systems and services. In general, multi-agent

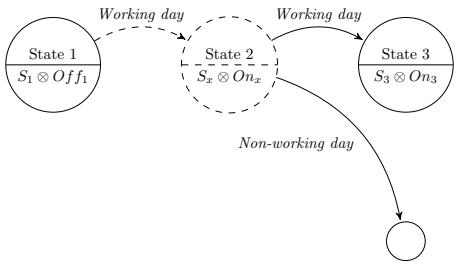

Figure 8: System S_2 is replaced with a novel system, S_x, in order to test the ability of the communication architecture to relearn how to interoperate with the new system. The dashed circle represents the state in which the new system participates.

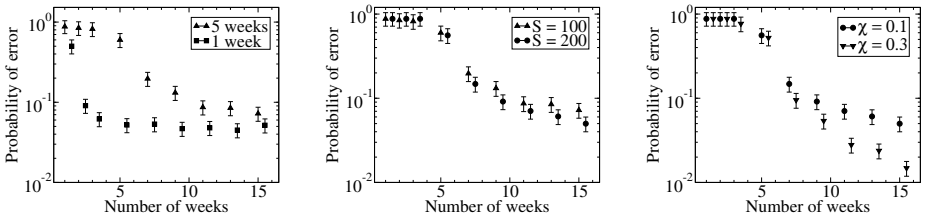

Figure 9: The probability that the communication architecture predicts an incorrect state at 95% confidence level versus the number of weeks that the new system has been connected (given that Alice or Bob provide one instruction for each state transition per day). Error bars denote standard deviations. The second dataset displayed in each panel is shifted forward in time by one half week for clarity, so that the error bars do not overlap. (a) The parameters of the long-term memory are $\mathcal{D} = 8192$, $S = 100$ and $\chi = 0.1$. (b) The effect of the memory size, S; the other parameters are $\mathcal{D} = 8192$ and $\chi = 0.1$. (c) The effect of the probability for activation, χ; the other parameters are $\mathcal{D} = 8192$ and $S = 200$. After the 15th week the probability of error is $1.5 \cdot 10^{-2} \pm 3 \cdot 10^{-3}$.

systems exhibit learning and adaptation characteristics, but they do not solve the general problem of interoperability between heterogeneous devices and systems (Wirsing et al., 2013). Furthermore, fault tolerant techniques are usually not employed at the low level but at the more abstract levels that involve planning and cooperation of agents, which sometime leads to unreliable states (Baresi et al., 2013). This being said, there are some interesting works, such as: INGENIAS (Pavon et al., 2005), ASPECS (Cossentino et al., 2010) and SeSaMe (Baresi et al., 2013) based on this approach.

Another approach is to build adaptive middleware infrastructure inspired by biological (eco-)systems. It is argued that conventional approaches cannot meet the requirements of the next generation pervasive computing infrastructure (Huhns and Singh, 2005; Barros and Dumas, 2006; Zambonelli and Viroli, 2011). The motivation for a nature-inspired approach is that nature is constantly changing, both in terms of scale and the level of dynamic interaction between different complex systems, yet nature is able to adapt without a central controlling authority (Fisher, 2006; Agha, 2008; Zambonelli and Viroli, 2011). Two interesting examples of this approach are the BIONETS (Biologically-inspired Autonomic Networks and Services) (Miorandi et al., 2008) and SAPERE (Self Aware Pervasive Service Ecosystems) (Zambonelli et al., 2011) projects. Overall, both projects take inspiration from living organisms, ecosystems, and ecological and evolutionary processes to enable interoperability and integration of heterogeneous devices and systems.

Many different approaches have been explored to enable interoperability, however, to our knowledge, none have focused on the importance of the representations of information. The challenge of making systems interoperate is leading some researchers to seek inspiration in the field of aritificial intelligence, in particular the semantic web. However, in doing so, it is important to consider the literature discussing when such approaches might be unfruit-

ful (Uschold, 2001; Haikonen, 2009; Ekbia, 2010; Guns, 2013). In the fields of artificial intelligence and cognitive science, the representations chosen are commonly considered to determine which types of tasks the system can handle (Churchland and Sejnowski, 1992; Valiant, 2000; Gardenfors, 2004; Stewart and Eliasmith, 2012; Doumas and Hummel, 2012). Neural networks typically use distributed representations, while symbolic models use localist representations. Studies based on neural networks have shown that distributed representations can facilitate learning, generalization and are robust to noise; while symbolic models based on localist representations can be used to integrate structured information, perform symbolic manipulation and enable logical reasoning. The communication architecture proposed herein is based on a VSA, which mimics both these approaches in one mathematical model.

5. Discussion

Conventional approaches to designing, simulating and developing interoperable systems using hand-crafted semantic descriptions of services and information, have some challenging limitations. Taking inspiration from biology and over two decades of vector space modeling of cognition, the aim of this study is to investigate whether a vector symbolic architecture can enable interoperability between heterogenous systems through learning from examples. We show that this is possible in principle and demonstrate the proposed communication architecture in a number of simulation experiments. Information is encoded in high-dimensional binary vectors, which combine the representation of structure and content (or syntax and semantics) in a distributed fashion. The architecture has system-wide working and long-term memories (see Figure 2). The core idea is that interoperability should emerge by learning from sensory–motor functions with the help of the long-term memory, starting from an initial pseudorandom state, rather than by hand-crafting artificial semantic descriptions and rules. This way we can, at least in principle, avoid some of the difficulties facing conventional approaches. We also demonstrate a basic scenario where the architecture is able to relearn the correct behavior when a constituent system is replaced with a new, initially incompatible system. In the simulation experiments Alice and Bob give compatible instructions, thereby introducing an invariance in the long-term memory with respect to the operator ID, which makes generalization to unknown operators possible. This mechanism is discussed by Emruli and Sandin (2014) and forms the basis for the analogical mapping capability of the memory.

We take a different approach compared with the mainstream literature, in which meaning is not seen as a label in an ontology (defined artificially by humans) but as interconnections between systems, that are the "owners" of the meaning, and can interpret information differently depending on implementation, context, goals and human–machine interactions. In a sense, the meaning is grounded (Harnad, 1990; Barsalou, 2008) in the interactions between the physical environment and the sensory–motor functions of the systems. The problem how to achieve this in a fully autonomous and optimal way is an open

question. The proposed communication architecture enables basic prediction of learned state transitions, which for example can be demonstrated by a human operator for automation or anomaly detection purposes. In total, the architecture has only three exogenous parameters that need to be defined by the architect, and the importance of each parameter is briefly discussed in the simulation experiments in Section 3. The nature of distributed representations in the working and long-term memory, and the simplicity of BSC operations make the architecture suitable for parallel and distributed implementation. Furthermore, a VSA approach was recently considered for collective communication in a dense sensor-network environment (Jakimovski et al., 2012). These are some of the features that make the architecture interesting for applications involving resource-constrained and wirelessly communicating devices.

This being said, we do not suggest that the proposed model can replace conventional approaches at this early stage of development, but it offers a new way of thinking about the challenging and important problems involved in making systems compatible and interoperable. In particular, the proposed architecture includes the basic concepts needed to enable system interoperability by learning, forming a novel conceptual basis for further research and development. Here we limit the simulation experiments to source–target transitions between two states, i.e., a first-order Markov chain, but the architecture as such can approximate more complex sequences (Kanerva, 1988; Flynn et al., 1989; Kanerva, 2009; Mendes and Coimbra, 2012; Emruli et al., 2013). One direction for further research is to investigate how an architecture of this type can be combined with a conventional architecture, e.g., on SOA, so that the complementary properties of these two approaches can be exploited in applications. Moreover, the vector symbolic representations and operations implemented in the working- and long-term memories can in principle be mapped to neural networks, which opens up the possibility for future developments of biologically more plausible implementations with spiking neurons and learning in a dynamic environment.

Acknowledgements

We thank Pentti Kanerva and Ross Gayler for questions that helped us improve the manuscript, and Sergio Martin-del-Campo, Denis Kleyko and Evgeny Osipov for proofreading. B.E. thanks Nordeas Norrlandsstiftelse and the Wallenberg Foundation for travel grants. This work was partially supported by the ARTEMIS Arrowhead project and the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), grant number IG2011-2025.

References

Abbott, J., Hamrick, J., and Griffiths, T. (2013). Approximating Bayesian inference with a sparse distributed memory system. In *Proceeding of The 35th Annual Conference of the Cognitive Science Society*, pages 1686–1691.

Ackoff, R. L. (1971). Towards a system of systems concepts. *Management Science*, 17(11):661–671.

Agha, G. (2008). Computing in pervasive cyberspace. *Commun. ACM*, 51(1):68–70.

Agresti, A. and Coull, B. A. (1998). Approximate is better than "Exact" for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126.

Allen, R. J., Hitch, G. J., Mate, J., and Baddeley, A. D. (2012). Feature binding and attention in working memory: a resolution of previous contradictory findings. *Quarterly journal of experimental psychology (2006)*, 65(12):2369–2383.

Anderson, C. (1989). A conditional probability interpretation of Kanerva's sparse distributed memory. In *The Proceeding of International the Joint Conference on Neural Networks, 1989*, pages 415–417 vol.1.

Bach-y-Rita, P., Collins, C. C., Saunders, F. A., White, B., and Scadden, L. (1969). Vision substitution by tactile image projection. *Nature*, 221(5184):963–964.

Bach-y-Rita, P., Tyler, M. E., and Kaczmarek, K. A. (2003). Seeing with the brain. *International Journal of Human-Computer Interaction*, 15(2):285–295.

Baresi, L., Guinea, S., and Shahzada, A. (2013). SeSaMe: towards a semantic self adaptive middleware for smart spaces. In Cossentino, M., Seghrouchni, A. E. F., and Winikoff, M., editors, *Engineering Multi-Agent Systems*, number 8245 in Lecture Notes in Computer Science, pages 1–18. Springer Berlin Heidelberg.

Barros, A. and Dumas, M. (2006). The rise of web service ecosystems. *IT Professional*, 8(5):31–37.

Barsalou, L. W. (2008). Grounded cognition. *Annual Review of Psychology*, 59:617–645.

Bohn, H., Bobek, A., and Golatowski, F. (2006). SIRENA - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains. In *The Proceeding of the International Conference on Networking and the International Conference on Systems ICN/ICONS/MCL*, page 43. IEEE Computer Society.

Carney, D., Fisher, D., and Place, P. (2005). Topics in interoperability: System-of-systems evolution. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Churchland, P. S. and Sejnowski, T. J. (1992). *The Computational Brain*. A Bradford Book.

Cossentino, M., Gaud, N., Hilaire, V., Galland, S., and Koukam, A. (2010). ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304.

Doumas, A. A. L. and Hummel, J. E. (2012). Computational models of higher cognition. In Holyoak, K. J. and Morrison, R. G., editors, *The Oxford Handbook of Thinking and Reasoning*, pages 52–66. Oxford University Press.

Ekbia, H. R. (2010). Fifty years of research in artificial intelligence. *Annual Review of Information Science and Technology*, 44(1):201–242.

Eliasmith, C. and Thagard, P. (2001). Integrating structure and meaning: a distributed model of analogical mapping. *Cognitive Science*, 25(2):245–286.

Emruli, B., Gayler, R. W., and Sandin, F. (2013). Analogical mapping and inference with binary spatter codes and sparse distributed memory. In *The Proceeding of the International Joint Conference on Neural Networks, 2013*, pages 1–8.

Emruli, B. and Sandin, F. (2014). Analogical mapping with sparse distributed memory: A simple model that learns to generalize from examples. *Cognitive Computation*, 6(1):74–88.

Fisher, D. (2006). An emergent perspective on interoperation in systems of systems. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Flynn, M. J., Kanerva, P., and Bhadkamkar, N. (1989). Sparse distributed memory: Principles and operation. Technical Report CSL-TR-89-400, Research Institute for Advanced Computer Science (RIACS).

Franklin, S., Madl, T., D'Mello, S., and Snaider, J. (2014). LIDA: a systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, 6(1):19–41.

Gallant, S. I. and Okaywe, T. W. (2013). Representing objects, relations, and sequences. *Neural Computation*, 25(8):2038–2078.

Gardenfors, P. (2004). *Conceptual spaces: The geometry of thought*. The MIT Press.

Gayler, R. W. (1998). Multiplicative binding, representation operators & analogy. In Gentner, D., Holyoak, K. J., and Kokinov, B. N., editors, *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*, page 405. New Bulgarian University, Sofia, Bulgaria.

Gayler, R. W. (2003). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. In *The Proceeding of Joint International Conference on Cognitive Science, 2003*, pages 133–138.

Gentner, D. and Forbus, K. D. (2011). Computational models of analogy. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(3):266–276.

Gentner, D. and Smith, L. A. (2013). Analogical learning and reasoning. In Reisberg, D., editor, *The Oxford Handbook of Cognitive Psychology*, pages 668–681. Oxford University Press.

Guns, R. (2013). Tracing the origins of the semantic web. *Journal of the American Society for Information Science and Technology*, 64(10):2173–2181.

Haikonen, P. O. A. (2009). The role of associative processing in cognitive computing. *Cognitive Computation*, 1(1):42–49.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1–3):335–346.

Hely, T., Willshaw, D., and Hayes, G. (1997). A new approach to Kanerva's sparse distributed memory. *IEEE Transactions on Neural Networks*, 8(3):791–794.

Henning, M. and Vinoski, S. (1999). *Advanced CORBA Programming with C++*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75.

Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 77–109. MIT Press, Cambridge, MA, USA.

Huhns, M. and Singh, M. (2005). Service-oriented computing: key concepts and principles. *IEEE Internet Computing*, 9(1):75–81.

Ifft, P. J., Shokur, S., Li, Z., Lebedev, M. A., and Nicolelis, M. A. L. (2013). A brain-machine interface enables bimanual arm movements in monkeys. *Science Translational Medicine*, 5(210):210ra154.

Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature Review Neuroscience*, 2(3):194–203.

Jakimovski, P., Schmidtke, H. R., Sigg, S., Chaves, L. W. F., and Beigl, M. (2012). Collective communication for dense sensing environments. *Journal of Ambient Intelligence and Smart Environments*, 4:123–134.

Jockel, S., Mendes, M., Zhang, J., Coimbra, A., and Crisostomo, M. (2009). Robot navigation and manipulation based on a predictive associative memory. In *The Proceeding of the 8th International Conference on Development and Learning, 2009*, pages 1–7.

Kaczmarek, K., Webster, J., Bach-y-Rita, P., and Tompkins, W. J. (1991). Electrotactile and vibrotactile displays for sensory substitution systems. *Biomedical Engineering, IEEE Transactions on*, 38(1):1–16.

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

Kanerva, P. (1993). Sparse distributed memory and related models. In *Associative Neural Memories: Theory and Implementation*, pages 50–76. Oxford University Press.

Kanerva, P. (1996). Binary spatter-coding of ordered k-tuples. In *The Proceeding of the International Conference on Artificial Neural Networks, 1994*, pages 869–873.

Kanerva, P. (1997). Fully distributed representation. In *Real World Computing Symposium*, volume 97, pages 358–365.

Kanerva, P. (2000). Large patterns make great symbols: An example of learning from example. In Wermter, Stefan; Sun, R., editor, *Hybrid Neural Systems*, volume 1778, pages 194–203. Springer Berlin Heidelberg.

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

Karnouskos, S., Colombo, A., Jammes, F., Delsing, J., and Bangemann, T. (2010). Towards an architecture for service-oriented process monitoring and control. In *The Proceeding of 36th Annual Conference on IEEE Industrial Electronics Society - IECON, 2010*, pages 1385–1391.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

McIlroy, M. D. (1976). Mass-produced software components. In *Software Engineering Concepts and Techniques*, pages 88–98. NATO Scientific Affairs Division.

Mendes, M. and Coimbra, P. A. (2012). Robot navigation based on view sequences stored in a sparse distributed memory. *Robotica*, 30(04):571–581.

Meng, H., Appiah, K., Hunter, A., Yue, S., Hobden, M., Priestley, N., Hobden, P., and Pettit, C. (2009). A modified sparse distributed memory model for extracting clean patterns from noisy inputs. In *The Proceedings of the International Joint Conference on Neural Networks, 2009*, pages 2084–2089.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, pages 81–97.

Miorandi, D., Carreras, I., Altman, E., Yamamoto, L., and Chlamtac, I. (2008). Bio-inspired approaches for autonomic pervasive computing systems. In Liò, P., Yoneki, E., Crowcroft, J., and Verma, D. C., editors, *Bio-Inspired Computing and Communication*, number 5151 in Lecture Notes in Computer Science, pages 217–228. Springer Berlin Heidelberg.

Mudry, A. and Mills, M. (2013). The early history of the cochlear implant: A retrospective. *JAMA Otolaryngology - Head and Neck Surgery*, 139(5):446–453.

Neumann, J. (2002). Learning the systematic transformation of holographic reduced representations. *Cognitive Systems Research*, 3(2):227–235.

Obrst, L. (2003). Ontologies for semantically interoperable systems. In *Proceeding of The Twelfth International Conference on Information and Knowledge Management, 2003*, pages 366–369. ACM.

O'Doherty, J. E., Lebedev, M. A., Ifft, P. J., Zhuang, K. Z., Shokur, S., Bleuler, H., and Nicolelis, M. A. L. (2011). Active tactile exploration using a brain-machine-brain interface. *Nature*, 479(7372):228–231.

Papazoglou, M. (2003). Service-oriented computing: concepts, characteristics and directions. In *Proceeding of The Fourth International Conference on Web Information Systems Engineering, 2003*, pages 3–12.

Pavon, J., Gomez-Sanz, J. J., and Fuentes, R. (2005). The ingenias methodology and tools. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 236–276. IGI Global.

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 for Cognitive Structures*. Center for the Study of Language and Information (CSLI).

Rachkovskij, D. A. and Kussul, E. M. (2001). Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13(2):411–452.

Rachkovskij, D. A., Kussul, E. M., and Baidyk, T. N. (2013). Building a world model with structure-sensitive sparse binary distributed representations. *Biologically Inspired Cognitive Architectures*, 3:64–86.

Rao, R. P. N. and Fuentes, O. (1998). Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Autonomous Robots*, 5(3-4):297–316.

Reynolds, J. H. and Desimone, R. (1999). The role of neural mechanisms of attention in solving the binding problem. *Neuron*, 24(1):19–29, 111–125.

Rogers, D. (1989). Statistical prediction with Kanerva's sparse distributed memory. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 586–593. Morgan-Kaufmann.

Rogers, D. (1990). Predicting weather using a genetic memory: A combination of Kanerva's sparse distributed memory with Holland's genetic algorithms. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 455–464. Morgan-Kaufmann.

Sheth, A. (1999). Changing focus on interoperability in information systems: From system, syntax, structure to semantics. In Goodchild, M., Egenhofer, M., Fegeas, R., and Kottman, C., editors, *Interoperating Geographic Information Systems*, pages 5–29. Kluwer Academic Publishers.

Souza, L. M. S. d., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., and Savio, D. (2008). SOCRADES: a web service based shop floor integration infrastructure. In Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F., and Sarma, S. E., editors, *The Internet of Things*, number 4952 in Lecture Notes in Computer Science, pages 50–67. Springer Berlin Heidelberg.

Stewart, T. and Eliasmith, C. (2012). Compositionality and biologically plausible models. In Werning, M., Hinzen, W., and Machery, E., editors, *The Oxford Handbook of Compositionality*, pages 596–615. Oxford University Press.

Sutton, R. S. and Whitehead, S. D. (1993). Online learning with random representations. In *Proceeding of The Tenth International Conference on Machine Learning, 1993*, pages 314–321. Morgan Kaufmann.

Treisman, A. (1998). Feature binding, attention and object perception. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 353(1373):1295–306.

Tsukada, K. and Yasumura, M. (2004). ActiveBelt: belt-type wearable tactile display for directional navigation. In Davies, N., Mynatt, E. D., and Siio, I., editors, *UbiComp 2004: Ubiquitous Computing*, number 3205 in Lecture Notes in Computer Science, pages 384–399. Springer Berlin Heidelberg.

Uschold, M. (2001). Where are the semantics in the semantic web. *AI Magazine*, 24:2003.

Valiant, L. G. (2000). *Circuits of the Mind*. Oxford University Press, USA.

Vinoski, S. (1997). CORBA: integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, 35(2):46–55.

von Melchner, L., Pallas, S. L., and Sur, M. (2000). Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404(6780):871–876.

Wichert, A. (2011). The role of attention in the context of associative memory. *Cognitive Computation*, 3(1):311–320.

Wilson, B. S. (2013). Toward better representations of sound with cochlear implants. *Nat Med*, 19(10):1245–1248.

Wilson, B. S., Finley, C. C., Lawson, D. T., Wolford, R. D., Eddington, D. K., and Rabinowitz, W. M. (1991). Better speech recognition with cochlear implants. *Nature*, 352(6332):236–238.

Wirsing, M., Hölzl, M., Tribastone, M., and Zambonelli, F. (2013). ASCENS: engineering autonomic service-component ensembles. In Beckert, B., Damiani, F., Boer, F. S. d., and Bonsangue, M. M., editors, *Formal Methods for Components and Objects*, number 7542 in Lecture Notes in Computer Science, pages 1–24. Springer Berlin Heidelberg.

Zambonelli, F., Castelli, G., Ferrari, L., Mamei, M., Rosi, A., Marzo, G. D., Risoldi, M., Tchao, A.-E., Dobson, S., Stevenson, G., Ye, J., Nardini, E., Omicini, A., Montagna, S., Viroli, M., Ferscha, A., Maschek, S., and Wally, B. (2011). Self-aware pervasive service ecosystems. *Procedia Computer Science*, 7:197–199.

Zambonelli, F. and Viroli, M. (2011). A survey on nature-inspired metaphors for pervasive service ecosystems. *International Journal of Pervasive Computing and Communications*, 7(3):186–204.

Zentall, T. R., Wasserman, E. A., Lazareva, O. F., Thompson, R. R. K., and Ratterman, M. J. (2008). Concept learning in animals. *Comparative Cognition & Behavior Reviews*, 3:13–45.

Zentall, T. R., Wasserman, E. A., and Urcuioli, P. J. (2014). Associative concept learning in animals. *Journal of the Experimental Analysis of Behavior*, 101(1):130–151.