Algorithmic ersatz for VSA: Macroscopic simulation of Vector Symbolic Architecture

Chloé Mercier [a,*] and Thierry Viéville [a,**]
[a] *Mnemosyne Team, Inria Bordeaux, U. Bordeaux, LaBRI and IMN, France*
E-mails: chloe.mercier@inria.fr, thierry.vieville@inria.fr

Submitted to Neurosymbolic Artificial Intelligence

Abstract.
Spiking neuronal networks are biologically plausible implementations of brain circuit computations, meaning that they can provide a way to manipulate symbols embedded as numeric vectors that carry semantic information. More precisely, the Neural Engineering Framework (NEF), relying on a vector symbolic architecture (VSA) formalism, bridges the gap between tightly interleaved numerical and symbolic (including formal) computations. Determining how the brain can implement such processing is an important issue.

In the present work, following this track, we consider such a VSA-based formalism, and propose an implementation at a macroscopic level, thus a higher order of magnitude of scale than usual mesoscopic implementations. We also attempt to provide a better natural representation of usual human symbolic operations, considering an implementation of modal logic.

Beyond usual VSA data structures such as associative memories we also introduce the notion of "relation maps" corresponding to relational memories, as observed in the brain.

An open-source implementation is provided with a benchmark and experimental observation of the method's performance and limitations.

Keywords: Vector Symbolic Architecture, Semantic Pointer Architecture, Modal Logic, Neuro-symbolism

1. Introduction

1.1. Biologically plausible neurosymbolic representations

As a possible entry point to considering a biologically plausible implementation at a symbolic level, vector symbolic architectures (VSAs) were introduced as a way to manipulate symbolic information represented as numeric vectors (see, e.g., [22] for an introduction). VSAs have been proven to help model high-level cognition and account for multiple biological features [13, 16]. More specifically, the semantic pointer architecture (SPA) [13] instantiates so-called semantic pointers (i.e., vectors that carry semantic information) and makes it possible to manipulate them in networks of spiking neurons. This approach takes a significant step towards the unification of symbolic and sub-symbolic processing, providing a way to translate the former into the latter. Consequently, complex knowledge

*Supported by https://team.inria.fr/mnemosyne/en/aide.E-mails: chloe.mercier@inria.fr, thierry.vieville@inria.fr.
**Corresponding author. E-mail: thierry.vieville@inria.fr.

representations in the form of compositional structures that are traditionally restricted to symbolic approaches can now be distilled in numerical and even neural[1] systems [8].

How can we represent a symbol in a neuronal assembly? A localist representation (one neuron or neuron group represented by a symbol) does not correspond to what is observed in the brain, and the basic idea is that a symbol corresponds to a pattern of activity distributed over the whole assembly. Let us consider a spiking neuron network and quantify its activity using, e.g., the neuron rates or higher-order statistics (see, e.g., [5] for a discussion). As developed by [14], this includes timing codes and population codes (i.e., relative timing codes between neurons). In the Neural Engineering Framework (NEF) [14], this high-dimensional set of bounded quantitative values can be collected and normalized, as a unitary stochastic vector in a high-dimensional space (with more than hundred of thousand dimensions for a biological neuronal map and often a few hundred dimensions at the simulation level), thus defining a SPA (building upon a particular case of VSA). The NEF provides a set of principles for implementing such an architecture through synaptic connections, including a time representation in spiking neuron systems (rather than, e.g., other representations based on synchrony within the neural assembly; (see [14] for technical details). This framework is a rather scalable alternative for a biologically plausible implementation of VSA, and it has already been implemented into a simulator called Nengo [1].

In the present study, we consider these developments as prerequisites and will simply consider that neural assembly activity is represented by a high-dimensional unary stochastic vector. We also need to specify transformations and define them at this abstract algebraic level. Mainly, following [24], we will consider the auto-association mechanism, as developed in [36], and functional transformations, as detailed in [14].

1.2. What is this paper about?

We first revisit how to encode symbols within the VSA approach based on the framework introduced in [13], targeting a macroscopic level of modeling. We describe how to generalize symbol encoding considering a related degree of belief, beyond binary information, and following [24], we explain the semantic interest of such a generalization. We consider the Vector-Derived Transformation Binding (VTB) operator [17], for which we recall its algebraic properties in Appendix C.

We then consider hierarchical knowledge structures in the sense of, e.g., [12], as a complement to associative and sequential memorization, and we revisit how to implement such a memory structure using the VSA formalism. To this end, we review VSA data structures and demonstrate that they are related to cognitive memory classification according to [12]. To better understand their computational properties, we also illustrate how such data structures compare to programming language containers, in Appendix B. We introduce a new data structure implementing "relational maps" expressing semantic knowledge [28]. This data structure is important for a class of symbolic derivations and exemplifies the benefits of our macroscopic implementation.

We then illustrate the proposed mechanism comparing to an existing simulation at the mesoscopic level, utilizing Nengo simulator [1], with a simulation at the macroscopic scale. We show that such computations may be, up to a certain point, approximated without explicitly performing mesoscopic computations at the vector component level; instead, an algorithmic ersatz can be used.

We finally discuss the applications and limit of this alternative approach.

Notations and Layout. We write vectors and matrices in bold letters (bold capital letters for matrices), and scalars in italic. The dual quantity of a vector $\mathbf{x}$ is represented as its transpose $\mathbf{x}^T$. The dot product between two vectors $\mathbf{x} \cdot \mathbf{y}$ can thus also be written $\mathbf{x}^T\,\mathbf{y}$. Components of vectors and matrices are represented here using subscripts.

We use the Kronecker notation $\delta_{\mathcal{P}} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \mathcal{P} \text{ is true} \\ 0 & \text{if } \mathcal{P} \text{ is false} \end{cases}$

In this paper, we consider normal distributions $\mathcal{N}(0,\sigma)$, i.e., Gaussian distribution with a null mean and a standard deviation σ, and we write the related random variable $\nu(\sigma)$.

Notice: In order to the paper easily readable, we provide verbal evidences in the text, while demonstrations of statements are given in footnotes, and non straightforward derivations are done within a symbolic computing environment (our open-source program is available at https://gitlab.inria.fr/line/aide-group/macrovsa).

[1] The term "neural" refers to any type of nerve cell, whereas "neuronal" is specifically related to neurons.

2. Symbolic information encoding

Let us first revisit how VSA approaches implement symbolic computation, providing complementary details to be used for the macroscopic simulation of mesoscopic mechanisms.

2.1. Symbol encoding

At the numerical level, each symbol is implemented as a randomly drawn fixed unit d-dimensional vector $\mathbf{x} \in \mathcal{R}^d$. Typically, $d \simeq 100 \cdots 1000$, and we expect to manipulate $k \simeq 100 \cdots 10000$ symbols at the simulation level. Here, our macroscopic implementation uses a dimension of 256, also typically found in related studies such as [24]. In a cortical or brain map, the order of magnitude is higher since the vector corresponds to the neuronal map activity (close to $10^{5\cdots6}$) and the number of encoded symbols depends on which map is considered, bute could be rather high (about $10^{3\cdots4}$).

The vector components are drawn from a normal distribution $\mathcal{N}(0, 1/\sqrt{d})$, to have an average magnitude[2] of 1.

A similarity measure is now introduced to semantically compare two vectors. Classically, the cosine similarity (i.e., normalized dot product, denoted $\cdot$) is used to compute the semantic similarity between two unit vectors[3]:

$$\mathbf{x} \cdot \mathbf{y} \stackrel{\text{def}}{=} \mathbf{x}^\top \mathbf{y} = \cos\left(\widehat{\mathbf{x}, \mathbf{y}}\right),$$

where $\mathbf{x}^\top$ denotes the transpose of $\mathbf{x}$. This measure also corresponds to the angular distance between the vectors.

The key property is that, provided that the space dimension d is large enough, two randomly chosen different vectors will be approximately orthogonal. More precisely[4],

$$\mathbf{x}^T \mathbf{y} \sim \delta_{\mathbf{x}=\mathbf{y}} + \nu(1/d),$$

i.e., it is almost 1 if equal and 0 otherwise, plus centered normal noise [33]. At the numerical level, using basic mean and standard-deviation calculi[5], drawing vectors from such a distribution, and measuring the magnitude, orthogonality, and standard-deviation average values, we have verified for $d \simeq 100 \cdots 1000$ that we generate unary vectors with a relative precision on the magnitude below 0.3%, while orthogonality is verified with a relative precision below 0.4%; the noise standard deviation prediction relative precision is below 0.3%.

This allows us to define a hypothesis to decide whether the $\mathcal{H}_0$ hypothesis $\mathbf{x} \cdot \mathbf{y} = 0$ can be rejected, as detailed in Appendix A.

2.2. Modality encoding

2.2.1. The notion of belief

Most VSA approaches consider that two vectors $\mathbf{x}$ and $\mathbf{y}$ contain equivalent information, when the similarity τ equals 1. They also can contain other information. There are different ways to interpret this result. Here, we enrich the notion of something being either false or true using a numeric representation of, e.g., partial knowledge, as illustrated in Fig. 1. The true value corresponds to 1 (fully possible and fully necessary), the false value to -1 (neither possible nor necessary, i.e., impossible), and the unknown value to 0, which corresponds to a fully possible but not necessary value.

---

[2] Given independent normal samples of zero mean and standard-deviation σ, the square of the magnitude is the sum of d independent normal samples product, thus a chi-square distribution of standard-deviation $1/d$. Its sum is thus of standard-deviation 1 and so is the magnitude.

[3] Let us consider two vectors $v_1 = \mathbf{u} + \mathbf{w}_1$ and $v_2 = \mathbf{u} + \mathbf{w}_2$, carrying the same semantic information encoded in $\mathbf{u}$, plus some additional information $\mathbf{w}_1$ and $\mathbf{w}_2$ independent from $\mathbf{u}$ and from each other. Since independent vectors are orthogonal, $v_1^T v_2 = u^T u = 1$: this is the meaning of semantic similarity.

[4] It is known, that the product of these two zero mean random variables of standard-deviation $1/\sqrt{d}$ is a random variable of standard-deviation $1/\sqrt{d}^2$.
The product of these two normal random variables is not a normal variable but a linear combination of two independent Chi-square random variables, but we approximate then by a normal distribution, which is a conservative choice as detailed in Appendix A.
The dot-product can be considered as the d times the average value of this chi-square combination distribution over d samples, thus of the same variance, since an average value over d samples divides the variance by d, which is multiplied by d in this case.

[5] See https://raw.githubusercontent.com/vthierry/onto2spa/main/figures/z_score.mpl for the open-source code used in this subsection.

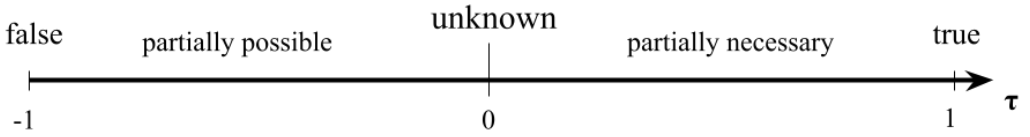

Fig. 1. Representation of partial truth $\tau \in [-1, 1]$ in relation to necessity and possibility, as defined in the possibility theory. The interpretation is that something partially possible but not necessary is unlikely, whereas what is likely is entirely possible but only partially necessary. Such a formulation corresponds qualitatively to the human appreciation of the degree of belief as proposed by, e.g., [34].

Our representation is in one-to-one correspondence with the dual notions of necessity and possibility representation in the standard possibility theory. Information is always related to a certain degree of what is called "belief" in this formalism. While almost all partially known information is related to probability, Piaget proposed that the human "level of truth" is more subtle and related to possibility and necessity [34], as formalized in modal logic. These notions are further developed in the possibility theory discussed in [10] and [11].

In other words, the possibility theory is devoted to the modeling of incomplete information, which is related to an observer's belief regarding a potential event and surprise after the event's occurrence. This is considered representative of what is modeled in educational science and philosophy [32]. Furthermore, in symbolic artificial intelligence, i.e., knowledge representation and logical inference, a link has been drawn between this necessity/possibility dual representation and ontology [38]. This must be understood as a deterministic theory, in the sense that partial knowledge is not represented by randomness[6]. This modal notion of partial belief has several semantic interpretations depending on the context [15] (i.e. not only epistemic[7] or doxastic[8] but also deontic[9] and so on). This representation has also been designed to be compatible with the ternary Kleene logic, in addition to being coherent with respect to the possibility theory, as discussed in detail in [42], where this deterministic representation of partial knowledge is generalized to include a probabilistic representation (using a 2D representation).

2.2.2. Implementing partial similarity knowledge

Let us now propose a design choice to apply this quantification to VSA symbols. A symbol representing a piece of information with a partial degree of belief $\tau \in [-1, 1]$ could be defined as:

$$\hat{\mathbf{x}} \stackrel{\text{def}}{=} \tau\,\mathbf{x},$$

where $\mathbf{x}$ corresponds to the numerical grounding of a symbol, and $\hat{\mathbf{x}}$ corresponds to the numerical grounding of a symbol, given its degree of belief τ.

Interestingly enough, this representation is coherent with the semantic similarity in the following sense: Are two vectors containing similar information? Considering $\mathbf{x} \cdot \mathbf{y}$, if this value is close to 1, then it is considered true, and the modal representation and semantic similarity are coherent. If it is almost equal to 0, then the modal representation is *not true*. Since our design choice is to consider being in an open world in which all that is not true is not necessarily false, but that we simply can not claim is its true, say it is unknown. To take this a step further, if this value is negative (down to -1), the modal representation considers that it is false, i.e., that the contrary is true, which is coherent with the semantic similarity, although negative values are not explicitly used, to the best of our knowledge, in the literature quoted in this paper.

Given these atomic ingredients, let us now study how they can be stored and manipulated in different data structure.

---

[6] This deterministic representation of partial knowledge can be generalized to also include a representation of the randomness belief. In the vanilla possibility theory, the possibility can be seen as an upper probability: Any possibility distribution defines a set of admissible probability distributions, i.e., a consonant plausibility measure in the Dempster–Shafer theory of evidence [2]. In [40, 42], it is proposed to bound the approximate probability, reconsidering the original notion of necessity, in order to also consider a lower bound of probability. This could be an interesting extension of the present work.

[7] Epistemic modal logic is concerned with reasoning about knowledge.

[8] Doxastic logic is a type of logic concerned with reasoning about beliefs.

[9] Deontic logic is the field of logic that is concerned with obligation, permission, and related concepts.

3. Knowledge structure encoding

3.0.1. Using bundling and binding to store information

As developed in detail in Appendix B and summarized in Table 1, the VSA formalism makes it possible to implement different data structures corresponding to different memory architectures, as defined by [12]. This corresponds to different programming containers. Further details on a scalable biologically plausible knowledge representation can be found in [8]. A key point of the present work is to verify that these VSA mechanisms generalize to modal symbol encoding (see Appendix B for an exhaustive development, after [24] who introduce such a design choice).

All data structures rely only on the two following operations:

- The bundling operation of N symbolic vectors $\mathbf{s}_i$ corresponds to a simple addition $\mathbf{s} \stackrel{\text{def}}{=} \sum_{k=1}^{N} \mathbf{s}_i$ while the similarity operator $\mathbf{s} \cdot \mathbf{s}_j$ allows to detect if the symbol related to $\mathbf{s}_j$ belongs to the bundling $\mathbf{s}$.
- The binding operation of a symbol $\mathbf{s}_1$ with another symbol $\mathbf{s}_2$ writes $\mathbf{s} \stackrel{\text{def}}{=} B_{\mathbf{s}_1}\, \mathbf{s}_2$ and enjoys the property that the corresponding unbinding operator $B_{\mathbf{s}_1^\sim}$ allows to retrieve $\mathbf{s}_2$, i.e., $\mathbf{s}_2 \simeq B_{\mathbf{s}_1^\sim}\, \mathbf{s}$.

A reader not familiar with the related VSA formalism will find in Appendix B a didactic introduction, while the choice of the binding operator from among several available binding operators [33] is discussed in Appendix C. For this section to be self-contained, one may simply consider that binding makes it possible to create a key-value symbol pair, while the unbinding operation makes it possible to retrieve the value from the key, as discussed below.

Container	VSA mechanism	Cognitive usage	Main available operations
Set	Bundling or superposition	Self-associative memory	+ Element insertion/modification + Membership check - No enumeration*.
Map or dictionary	Binding superposition	Auto-associative and hetero-associative memory**	+ Element insertion/modification + Value·s check from key + Key·s check from value + Symbol check from approximate input - No enumeration*.
Indexed and chained list	Ordinal binding superposition	Sequential memory	+ Element insertion/modification + Value enumeration.
Relational map	(see next subsection)	Hierarchical memory	+ Element insertion/modification - No enumeration*.

(*) Through, using imperative programming, such data structures do have enumeration capabilities, for VSA implementations, symbol enumeration is also easily implementable, but using an external mechanism.

(**) It is worth noticing that, for instance, associative maps are not necessarily defined combining bundling/binding operations, but we make the choice to restrain here to such algebraic definition in order to have an homogeneous setup to specify it at the macroscopic level.

Table 1

Biologically plausible data containers, using usual VSA implementations; see Appendix B for details..

3.0.2. Relational maps

From early artificial intelligence knowledge representation to modern web semantic data structures, one[10] basic idea of symbolic representation is to express knowledge through relationships, i.e., triple statements of the form

[10] Of course, other expressive frameworks, such as logical representations, frame-based semantics, and hyper-graphs, offer alternative models, which could be richer and more nuanced, than the ontology framework targeted here:
- On one hand, the link between the ontology framework and logical representations, namely description logic, is well established and developed, the key point being to propose a more expressive framework than propositional logic, but less expressive than first-order logic, in order to guaranty that the core reasoning problems for description logic are (as much as possible) decidable. Several level of specification allow the underlying description logic features of an ontology language level to exhibit different balance between expressive power and reasoning complexity. See,

`($subject, $predicate, $object)`, as schematized in Fig. 2, [4, 20]. We consider this structure here to demonstrate that beyond existing data structures implemented with VSA, we can also easily define more complex structures at a macroscopic level.

Here we target to define a set of relations between symbols, each predicate relating subjects to objects, thus defining a relation between subjects and objects. We propose[11] to call this a "relational map".

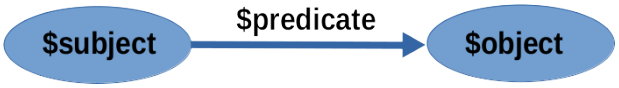

Fig. 2. Atomic representation of knowledge: To express some knowledge regarding a symbol, the subject, we define a feature with a predicate that has an object as an attribute (i.e., a quantitative data value or a qualitative symbol).

Such information can be implemented through a distributed representation using bundling and binding operations, i.e., associative maps. Following and generalizing [24], we propose to start with an architecture of combined associative memories as represented in Fig. 3. Each associative map stands for a predicate, as proposed and developed by, e.g., [35]. It integrates a demultiplexer which is nothing but another associative map, allowing to index the previous `subject` $\rightarrow$ `object` associative maps. At the algebraic level, this writes:

$$\mathbf{t}_{\mathbf{pso}} \stackrel{\text{def}}{=} \sum_i \mathbf{B}_{\mathbf{p}_i}\, \mathbf{t}_{\mathbf{p}_i}, \text{ with } \mathbf{t}_{\mathbf{p}_i} \stackrel{\text{def}}{=} \sum_{j,\mathbf{p}_i=\mathbf{p}_j} \mathbf{B}_{\mathbf{s}_j}\, \mathbf{o}_j,$$

where:

- $\mathbf{s}_i$, $\mathbf{p}_i$, $\mathbf{o}_i$ are vectors encoding the subject, predicate and object symbols;
- each $\mathbf{B_y}\,\mathbf{x}$ is a binding which corresponds to a key-value pair (the key is $\mathbf{y}$ and the value is $\mathbf{x}$);
- combining these with bundling (i.e., a simple sum) in $\mathbf{t}_{\mathbf{p}_i}$ allows to define an associative map, which unbinding operation allows to retrieve the object of a given subject and predicate:
$$\mathbf{B}_{\mathbf{s}_i^\sim}\, \mathbf{t}_{\mathbf{p}_j} \simeq \mathbf{o}_{ij} + \mathbf{unknown};$$
where "**unknown**´´ stands for a vector which is almost orthogonal, thus not similar, to any other vectors, as made explicit in Appendix B.
- selecting the associative map, given a predicate, is also done through the $\mathbf{t}_{\mathbf{pso}}$, implementing the demultiplexer:
$$\mathbf{B}_{\mathbf{p}_i^\sim} \mathbf{t}_{\mathbf{pso}} = \mathbf{t}_{\mathbf{p}_i} + \mathbf{unknown}.$$

To obtain such properties the choice of a non-commutative binding operator from among several available binding operators [33] is important, in order not to mix predicates, subjects, and objects.

Given a triple $(\mathbf{s}_0, \mathbf{p}_0, \mathbf{o}_0.)$, it is straightforward to verify to what extent it is stored in the relational map through unbinding:

$$(B_{\mathbf{s}_o^\sim}\, B_{\mathbf{p}_o \sim}\, \mathbf{t}_{\mathbf{pso}} \cdot \mathbf{o}_0).$$

---

for instance [18] for a development.

- On the other hand, frame-based semantics is a very interesting alternative to ontology frameworks to ease data and reasoning specifications, while it directly maps onto ontology frameworks, as discussed for instance in [21]. This mapping does not make this approach useless: On the contrary it allows another way to express knowledge. However, at our implementation level, we can consider that the mapping of such representation on ontology will allow one to open the possibility to extend the present work to this class of representation.
- A step further, hyper-graph representations intrinsically offers a richer representation than the previous ones, standing on labeled graph representations. Despite the fact that there is a trivial one to one mapping for an hyper-graph onto a bipartite graph, more precisely a Levy graph, it appears of interest to discuss specific adapted VSA implementation, beyond the scope of this work. This is briefly discussed in the conclusion.

[11] The choice of the term "relational map" is closely related to relational data models, which is exactly the idea of an ontology. Furthermore, "relational mapping" allows one to map an object model into a relational data model, usually a data-base, but not exclusively. Reciprocally, "object–relational mapping" allows to convert data between a relational database and a data-structure of an object-oriented representation. What we propose here is not a "mapping" (i.e. a transformation) but a data-structure, thus a "map". A step further, the term "relationship mapping" represents the connections between different entities (e.g. friendship between people), but in quite different contexts. Finally, "relation maps" is a mathematical term that define maps from, to or between relations.

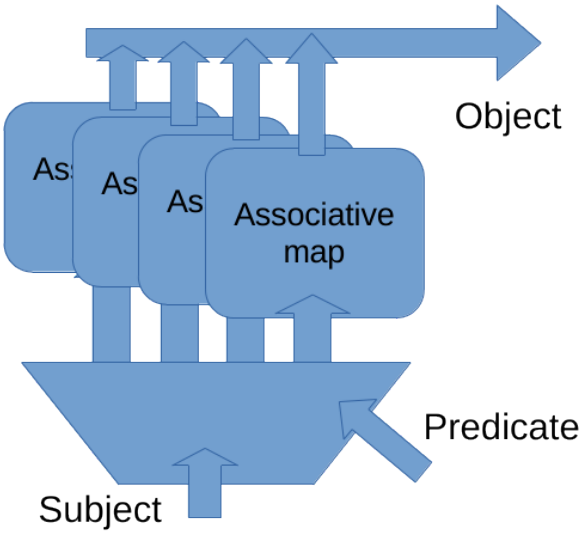

Fig. 3. A relational map as a row of associative memory. For each predicate, an associative memory stores the hetero-associations between subject and object. An input associative memory acts as a demultiplexor allowing to choose for a given predicate which associative memory to choose. At the algebraic level, this corresponds to a simple combination of bundling and binding operations.

This obviously generalizes to a triple multiplied by a modal τ value, while the related τ value is simply the product of the element's τ value.

We can also further obtain all objects of a given subject for a given property,

$$\mathbf{t}_{\mathbf{p}_j,\mathbf{s}_j} \stackrel{\text{def}}{=} \sum_{\mathbf{p}_j=\mathbf{p}_i,\mathbf{s}_j=\mathbf{s}_i} \mathbf{o}_i \simeq B_{\mathbf{s}_j^\sim}\, B_{\mathbf{p}_j^\sim}\, \mathbf{t}_{\mathbf{pso}} + \mathbf{unknown},$$

using the notation of Appendix C. We can also easily define:

$$\mathbf{t}_{\mathbf{p}_j,\mathbf{o}_j} \stackrel{\text{def}}{=} \sum_{\mathbf{p}_j=\mathbf{p}_i,\mathbf{o}_j=\mathbf{o}_i} \mathbf{s}_i \simeq B_{\mathbf{s}_j^\sim}\, \mathbf{B}_\leftrightarrow\, B_{\mathbf{p}_j^\sim}\, \mathbf{t}_{\mathbf{pso}} + \mathbf{unknown}.$$

However such a construction, up to the best of our knowledge, cannot allows to retrieve, for instance each predicate of a given subject. More precisely, there is no operation to recover $\mathbf{t}_{\mathbf{s}_j}$ or $\mathbf{t}_{\mathbf{s}_j,\mathbf{o}_j}$ from $\mathbf{t}_{\mathbf{pso}}$, and no operation to recover $\mathbf{t}_{\mathbf{p}_j}$ or $\mathbf{t}_{\mathbf{p}_j,\mathbf{o}_j}$ from $\mathbf{t}_{\mathbf{spo}}$. nevertheless, to this end, a dual construction, $\mathbf{t}_{\mathbf{spo}} \stackrel{\text{def}}{=} \sum_i \mathbf{B}_{\mathbf{s}_i}\, \mathbf{B}_{\mathbf{p}_i}\, \mathbf{o}_i$, with similar decoding formulae makes it possible to further access the properties of a given subject $\mathbf{t}_{\mathbf{s}_j} \stackrel{\text{def}}{=} \sum_{i,\mathbf{s}_j=\mathbf{s}_i} \mathbf{B}_{\mathbf{p}_i}\, \mathbf{o}_i$ or the properties of a given subject-object couple $\mathbf{t}_{\mathbf{s}_j,\mathbf{o}_j} \stackrel{\text{def}}{=} \sum_{i,\mathbf{s}_j=\mathbf{s}_i,\mathbf{o}_j=\mathbf{o}_i} \mathbf{p}_i$ using similar formulae. We thus shave now two relation maps $\mathbf{t}_{\mathbf{pso}}$ and $\mathbf{t}_{\mathbf{spo}}$. This is an important constraint, and it would be interesting to verify if such a constraint is observed at the level of the brain's semantic memory.

Again, in order to enumerate the different elements of these maps $\mathbf{t}_\bullet$, we need the corresponding indexing mechanisms discussed previously. If the basic operation is to enumerate all triples, with order constraints, then the choice of the storage architecture is not crucial; this is going to be the case later in this paper.

To take this a step further, we can also consider an additional symbol "something," and each time a triplet $(\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i.)$ is added, we can also add $(\sigma, \mathbf{p}_I, \mathbf{o}_i.)$, $(\mathbf{s}_i, \sigma, \mathbf{o}_i.)$, and $(\mathbf{s}_i, \mathbf{p}_I, \sigma.)$. This makes it possible to retrieve the fact that there is a link between the predicate and object, subject and object, and subject and predicate, without requiring the enumeration of the different elements[12].

At the cognitive level, this corresponds to cognitive maps interacting with each other and is a proposal to formalize the notion of hierarchical memory organization, as discussed in, e.g., [12].

[12] This has been proposed by Gabriel Doriath Döhler (unpublished research report).

At the computer programming level, this corresponds to a "triple store" used in ontology reasoners and is in fact a distributed representation of an oriented graph, in the form of an adjacency set for $\mathbf{t}_{\mathbf{spo}}$ construction and a hierarchical edge set for $\mathbf{t}_{\mathbf{pso}}$ construction.

However, comparing to existing development in the literature, defining and manipulating such relational maps, especially when considering rather large data ensembles, might becomes intractable to simulate at the microscopic level, or even at the mesoscopic level, and it might be useful for large scale application to study to what extent this could also be implemented at a macroscopic level, as developed now.

4. Implementation at the macroscopic scale

The VSA, when implemented using the NEF, allows a microscopic simulation of the neuronal processes, at the spiking neuronal network level, for memorization and processing operations. At a higher scale, when the VSA is implemented as described in Appendix C using linear algebra and permutation operations, we are at a mesoscopic scale, allowing us to perform the same operations without explicitizing the neuronal state value and evolution, but only random vectors linear algebra. This is one major advantage of this class of approaches.

To take this a step further, at a higher macroscopic scale, we could directly consider the previous operations, predicting the results of the different algebraic operations without explicitly working at the vector component level. Let us describe how this approach can be designed and implemented using what could be called an "algorithmic ersatz".

It is important to describe the implementation up to the data structure choice, in order to precisely exppalin what is taken into account and what is not.

4.1. Symbol indexing and specification

In the VSA, each symbol of the vocabulary is associated with a d-dimensional random vector. At the macroscopic scale, we only need to register each unary vector $\mathbf{u}_k$ using an integer number k, incremented for each new symbol. At the input/output level, the human-readable string (s_k) representation of the symbol is utilized, but it is not considered further here.

Weighted symbols of index i, correspond to a unary vector number k_i, with also a "belief" value $\tau_i \in [-1,1]$, as discussed in subsection 2.2, that is equal to 1 by default. They are also estimated up to a certain normal centered additive noise $\nu(\sigma)$ of standard deviation $\sigma_i \geqslant 0$, which is equal to 0 by default when no approximate operation has been applied to the symbol.

Two symbols may thus have the same unary vector number but different belief levels or different noise levels. The associative table of symbols is thus a simple associative array data structure of (k_i, τ_i, σ_i), corresponding to

$$\mathbf{x}_{k_i} = \tau_i \, \mathbf{u}_{k_i} + \nu(\sigma_i),$$

as illustrated in a more programmatic format in Fig. 4.

If and only if comparison with mesoscopic computation is required, the explicitization of the unary vector value with $\|\mathbf{u}_{k_i}\| = 1$ is added to the data structure.

With this representation

- Two symbols $\mathbf{x}$ and $\mathbf{x}'$ are approximately colinear if and only if they have the same unary vector index.
- Two colinear symbols $\mathbf{x}$ and $\mathbf{x}'$ are indistinguishable if and only if:

$$0 \simeq \|\mathbf{x} - \mathbf{x}'\| \simeq (\tau - \tau') \, \mathbf{u}_k + \nu(\sigma + \sigma'),$$

up to the first order, with $|\tau - \tau'| < \sigma + \sigma'$ as developed in Appendix A.

- The similarity between two symbols $\mathbf{x}$ and $\mathbf{x}'$ writes:

$$\mathbf{x}^T \, \mathbf{x}' \simeq \tau \, \tau' \, \delta_{k=k'} + \nu(\sigma + \sigma'),$$

up to the first order.

At this stage we thus have specified atomic symbol, or the dual of such a symbol.

```
Symbol : {
 unsigned int key;                          // index
 double       tau;                          // belief level
 double       sigma;                        // noise  level
 enum         type;                         // symbol type
 string       name;                         // human readable name
 double       mesoscopic_value[dimension]; // optional vector
}
```

Fig. 4. Programmatic implementation of a symbol at the macroscopic scale, adding an optional mesoscopic numeric vectorial value, to compare macroscopic and mesoscopic calculations.
The symbol type can be *atomic*, i.e. scalar, such a symbol is only defined by its name, without including binding or bundling, or either *bundling* or *binding* with a complementary data structure, as developed below.

4.2. Symbolic derivation of compounded symbols

Given atomic symbols that are randomly drawn, using the enumerating operations given in Appendix C, we have to compute compounded symbols composed through bundling and binding (while unbinding is a simple binding with the dual symbol, thus taken into account with binding), while in order to compute entailment rules, we need to be able to compute the similarity between any of these compounded symbols.

At the macroscopic level, it is straightforward to define an "oracle" that can calculate the result of all operations as follows:

Bundling canonical representation. A symbol corresponding to the *bundling* of other symbols is fully defined by the symbol set and their corresponding τ and σ values. The key point is to have chosen an implementation that allows to maintain a normalized representation of the bundling elements, as follows.

The programmatic implementation of a bundling is an associative map:

$$\texttt{symbol-id} \longrightarrow \texttt{Symbol}$$

the `Symbol` data structure being defined in Fig.4.
- When a new symbol is added,
 if the symbol index is already present in the bundling, the τ and σ values are updated,
 otherwise a new map entry is created, making use of commutativity to group all symbol with same index.
- If the added symbol is itself a bundling, its components are directly expanded, making use of associativity, i.e., the fact that a bundling of bundling is a bundling.
- Deleting a symbol is not explicitly defined in usual VSA implementation. At the mesoscopic level, this corresponds to subtract a vector to the bundling, i.e., add it with a negative τ value, while the σ value is updated to take into account the fact this numeric operation is performed with some additive noise.

Usual operation over a bundling (such as binding or computing similarity) mainly results from applying the operation to each element.

The representation is canonical in the sense where, if equality is well-defined on their components, two bundlings are semantically equal, if and only if they are syntactically equal, i.e., if they have the same symbol map, and each corresponding symbols are equal, more precisely indistinguishable, when considering noise.

Associative map canonical representation. A symbol corresponding to an *associative map* is a binding of bundling, thus implementable using the two other mechanisms. However, to obtain the best performances, we consider a specific canonical data structure corresponding to an associative multi-map, i.e., a map mapping a key symbol to a set of symbol values. We use:

$$\texttt{symbol-id} \longrightarrow \texttt{(Symbol, (symbol-id} \longrightarrow \texttt{Symbol))}$$

in words a map mapping key IDs (i) to the related symbol and (ii) to the map of all symbol values.

The symbol addition mechanism is entirely similar to the bundling mechanism, since we reuse here the $\texttt{symbol-id} \longrightarrow \texttt{Symbol}$ representation. Conversion to the binding of bundling form is obvious to implement.

Binding canonical representation. A symbol corresponding to the *binding* of one symbol onto another is fully defined by the pair of symbols and yields either a reduction if it is the corresponding unbinding operation or a binding combination.

Then, to reduce such an expression we thus simply have to recursively[13]:
+ Expands $\mathbf{B_y}\ \mathbf{x}$ binding on the $\mathbf{y}$ argument if it is a bundling.
+ Expands $\mathbf{B_y}\ \mathbf{x}$ binding on the $\mathbf{x}$ argument if it is a bundling.
+ Reduces dual $\mathbf{B_y}\ \mathbf{B_y}{\sim}\ \mathbf{x}$ or $\mathbf{B_y}{\sim}\ \mathbf{B_y}\ \mathbf{x}$ binding/unbinding operation, on elementary or compounded symbols.

Let us call "atomic binding case", when for every $\mathbf{B_y}$, $\mathbf{y}$ is an atomic symbol. When considering all binding operations regarding used data structures, as discussed in detail in Appendix B, or literature quoted in this paper, we are in this atomic binding case. In such a case, the iterative application of these three operations allows to obtain all expressions in a canonical form, written:

$$\textstyle\sum_k \prod_{l=1}^{l_k} \mathbf{B}_{\mathbf{y}_{kl}}\ \mathbf{x}_k$$

where $\mathbf{x}_k$ and $\mathbf{y}_{kl}$ are atomic symbols and $\mathbf{y}_{kl} \neq \mathbf{y}^{\sim}_{k(l+1)}$, while we may have $l_k = 0$ thus omitting the binding operation.

At the programmatic level, binding and unbinding are simply defined by the same operator, with a flag to specify binding or unbinding.

In order to obtain a canonical representation with respect to (τ, σ) values, we use the 1st order relation discussed in the sequel, which is directly obtained from the linearity of the binding operator:

$$B_{\tau\,\mathbf{y}+\nu(\sigma)}\ (\tau'\,\mathbf{x} + \nu(\sigma')) = B_{\mathbf{y}}\ (\tau\,\tau'\,\mathbf{x} + \nu(\tau\,\sigma' + \tau'\,\sigma) + O(\sigma\,\sigma')$$

where $O(\sigma\,\sigma')$ contains second order terms with respect to the 1st order noises. As a consequence the $\mathbf{y}$ operator of a binding is always defined with $\tau = 1$ and $\sigma = 0$.

For the sake of generality, let us also discuss canonical forms beyond the "atomic binding case". Interestingly enough, there are no more reductions in terms of a flatter or reduced expression in this general case[14]. In other

---

[13] More formally, we consider the following rules:

$$\begin{array}{ll} e_y & \mathbf{B}_{\sum_k \mathbf{y}_k}\ \mathbf{x} \rightarrow \sum_k \mathbf{B}_{\mathbf{y}_k}\ \mathbf{x} \\ e_x & \mathbf{B_y}\ \sum_k \mathbf{x}_k \rightarrow \sum_k \mathbf{B_y}\ \mathbf{x}_k \\ r_1 & \mathbf{B_y}\ \mathbf{B_y}{\sim}\ \mathbf{x} \rightarrow \mathbf{x} \\ r_2 & \mathbf{B_y}{\sim}\ \mathbf{B_y}\ \mathbf{x} \rightarrow \mathbf{x} \end{array}$$

where
- e_y and e_x correspond to bundling expansion over binding,
- r_1 and r_2 correspond to the binding/unbinding reductions.

Applying recursively e_y and e_x guaranty that there is no bundling in the $\mathbf{y}$ and $\mathbf{x}$ arguments of the binding, thus yields to an expression of the form

$$\textstyle\sum_k \prod_l \mathbf{B}_{\mathbf{y}_{kl}}\ \mathbf{x}_k$$

where $\mathbf{x}_k$ are scalar symbols. This simply corresponds to a symbolic expansion of an expression, and it is well established (see, e.g., [3]) that the recursive application leads to a fixed point, and a canonical form, as for the expansion of, e.g., a polynomial.

Note that $\mathbf{y}_{kl}$ is either a scalar symbol or a compounded symbol of the form $\prod_h \mathbf{B}_{\mathbf{y}_{hkl}}\mathbf{x}_{kl}$, because the binding is associative with respect to the $\mathbf{x}$ argument, since it corresponds to a matrix multiplication, but not with respect to the $\mathbf{y}$ argument, as further discussed in the sequel.

Reducible binding couples are of the form $\mathbf{B_y}\ \mathbf{B_y}{\sim}$ or $\mathbf{B_y}{\sim}\ \mathbf{B_y}$, where $\mathbf{y}$ is either a scalar or a compounded symbol involving bindings, as made explicit with rules r_1 and r_2. These rules allows the reduction of such binding or unbinding couples. After such application, there is no guaranty that that reducible binding couples do not remain. Therefore, we have to apply r_1 and r_2 recursively on the resulting expression until no more reducible binding couples left. This again leads to a fixed point since, for instance, the expression positive size reduces at each step.

[14] Let us consider the idempotent mirroring matrix $\mathbf{B}_{\leftrightarrow}$, and the related dual operator ${\sim}$, also well-defined and detailed in appendix C.:

$$\mathbf{B}_{\leftrightarrow}\ \mathbf{x} = \mathbf{x}^{\sim} \text{ and } \mathbf{B}_{\leftrightarrow}\ \mathbf{B_y}\ \mathbf{x} = \mathbf{B_x}\ \mathbf{y}$$

We now have to consider expressions where the binding left parameter is also binding, i.e., of the form $\mathbf{B}_{\mathbf{B_y}\ \mathbf{z}}\ \mathbf{x}$. Thanks to the dual operator, it expands as:

$$e_b\ \mathbf{B}_{\mathbf{B_y}\ \mathbf{z}}\ \mathbf{x} \rightarrow \mathbf{B}_{\leftrightarrow}\ \mathbf{B_x}\ \mathbf{B_y}\ \mathbf{z} = [\mathbf{B_x}\ \mathbf{B_y}\ \mathbf{z}]^{\sim}\ .$$

However, as discussed in Appendix C, beyond idempotence, the dual operator ${\sim}$ neither expands nor simplifies over binding operations. It

words, we still derive a canonical form of an expression with binding.

A tsep further, the notion of approximate colinearity or equality (i.e., indistinguishably) between two binding symbols is easily defined, as it is simply induced by the same notion on the two left and right symbols.

Limit of the implementation. This symbolic derivation assumes that no symbol with two different names has a hidden similarity, i.e.:

$$\forall \mathbf{x}, \mathbf{y},\ \mathbf{x} \cdot \mathbf{y} \simeq 0 \text{ and } \mathbf{x} \cdot \mathbf{y}^{\sim} \simeq 0.$$

These symbolic derivation rules are in fact nothing but a subset of usual algebraic normalization rules, where the bundling stands for the sum, what it is here, and the bundling is a non-commutative product, with a left-inversion mechanism. We refer to the symbolic computation textbook for further details (see, e.g., [3]).

4.3. Symbol noise derivation

At the mesoscopic scale, calculations are made up to the floating-point machine precision, which is not taken into account here. The operations rely on the fact that we consider random vectors in a high-dimensional space, and thus they are approximately orthogonal up to, up to the first order, a normal centered additive noise. The main operations are the dot product used to calculate the similarity, as detailed in subsection 2.1, and the approximation of the matrix inverse using its transpose for unbinding, as detailed in Appendix C.

We must thus consider a level of noise for each symbol and update this level of noise after each calculation; this noise, up to the first order, can still be represented by a centered normal distribution. This cannot be neglected, because we also introduce a level of belief value that can be small and thus is not negligible with respect to the noise level. We denote by $\sigma_\bullet \stackrel{\text{def}}{=} O(1/d)$ the order of magnitude added by an approximate operation, as discussed previously in this paper.

On the one hand, considering the similarity operation between two symbols, we obtain[15] for the dot product

---

only expands over bundling. This means that we can not further reduce or normalize binding expressions for which the left argument, named **y** here, is not atomic.

In other words, the three form on the left-hand and right-hand size of e_b are simply equivalent and do not lead to further derivation, except trivial useless ones (e.g., adding a zero). Therefore, up to our best understanding, no other expression with **x**, **y**, and **z**, can be semantically equal to one of the three equivalent expressions.

It means that two expressions for which the right parameter is in canonical form as discussed previously and the left parameter is a binding itself in canonical form, are semantically equal if and only if they are syntactically equal.

[15] The derivation is written as follows:

$$\begin{aligned}&((\tau_i\,\mathbf{u}_{k_i} + \nu(\sigma_i)) \cdot (\tau_j\,\mathbf{u}_{k_j} + \nu(\sigma_j)) = \\ &\quad \tau_i\,\tau_j\,\mathbf{u}_{k_i} \cdot \mathbf{u}_{k_j} + \tau_i\,\mathbf{u}_{k_i} \cdot \nu(\sigma_j) + \tau_j\,\mathbf{u}_{k_j} \cdot \nu(\sigma_j) + \nu(\sigma_i)) \cdot \nu(\sigma_j).\end{aligned}$$

If $k_i \neq k_j$, then $\mathbf{u}_{k_i} \cdot \mathbf{u}_{k_j} = \nu(\sigma_\bullet)$ since these random vectors are approximately orthogonal up to normal noise with a standard deviation with an order of magnitude of $\sigma_\bullet$ [43], whereas if $k_i = k_j$, then $\mathbf{u}_{k_i} \cdot \mathbf{u}_{k_j} = 1$ since these are unary vectors.

Then, $\mathbf{u}_{k_i} \cdot \nu(\sigma_j)$ is the dot product, and it is a random variable of mean $\mathbb{E}\left[\mathbf{u}_{k_i} \cdot \nu(\sigma_j)\right] = 0$, since vectors are assumed to be independent of other sources of noise up to the first order, and variance $\mathbb{E}\left[\mathbf{u}_{k_i}^T\,\nu(\sigma_j)\,\nu(\sigma_j)^T\,\mathbf{u}_{k_i}\right] = \sigma_j^2$ since the covariance $\nu(\sigma_j) \cdot \nu(\sigma_j)^T = \sigma_j^2\,\mathbf{I}$ because the noise is isotropic; meanwhile, $\mathbf{u}_{k_i}^T\,\mathbf{u}_{k_i} = 1$ since it is a unary vector. Note that here, $\mathbf{u}_{k_i}$ is not a random variable; it stands for the mean of the random vector drawn.

The derivation for $\mathbf{u}_{k_j} \cdot \nu(\sigma_i)$ is identical.

Assuming that $\nu(\sigma_i)$ and $\nu(\sigma_j)$ are independent and of zero mean, as hypothesized, its related variance is known to be $\sigma_i^2\,\sigma_j^2$. The product of these two normal distributions is not a normal distribution; instead, it is a linear combination of chi-square distributions in the general case. However, here, as it is a second-order term, with respect to the expected small values of σ_i and σ_j, it is negligible. Collecting these results, we obtain that up to the first order,

$$(\tau_i\,\mathbf{u}_{k_i} + \nu(\sigma_i)) \cdot (\tau_j\,\mathbf{u}_{k_j} + \nu(\sigma_j)) = \tau_i\,\tau_j\,\delta_{k_i=k_j} + \nu\left(\underbrace{|\tau_j|\,\sigma_i + |\tau_i|\,\sigma_j + |\tau_i\,\tau_j|\,\sigma_\bullet}_{\stackrel{\text{def}}{=}\sigma_{ij}}\right),$$

yielding the expected result.

$$(\tau_i \mathbf{u}_{k_i} + \nu(\sigma_i)) \cdot (\tau_j \mathbf{u}_{k_j} + \nu(\sigma_j)) = \tau_i \tau_j \delta_{k_i=k_j} + \nu(\sigma_{ij}), \sigma_{ij} < \sigma_i + \sigma_j + \sigma_\bullet,$$

up to the first order, considering that the noise is independent of the vector values up to the first order. We can thus perform this operation without explicitly computing the dot product.

Here, we show a conservative choice by proposing an upper bound for the noise, while the exact value, up to the first order, of σ_{ij} can also easily be used, and is implemented. This design choice is also conservative with respect to a mesoscopic implementation, because it increases the noise at each operation, whereas at the mesoscopic level, each numerical random vector is drawn once; thus, depending on the combination of operations, the noise may not increase. We consider here that noise must be added at each step, and we wonder if this is not more realistic than a frozen noise value. However, it would have been possible (but quite computationally heavy) to consider freezing noise values and caching them in some tables.

On the other hand, considering a symbol of index j binded by a symbol of index i and unbinded by a symbol of index i' so that $k = k_i = k_{i'}$, in order to ensure a valid binding/unbinding operation, we obtain, up to the first order[16],

$$\mathbf{B}_{(\tau_i \mathbf{u}_i + \nu(\sigma_i))} (\tau_j \mathbf{u}_j + \nu(\sigma_j)) = \tau_i \tau_j \bar{\mathbf{u}}_{ij} + \nu(\sigma'_{ij}), \sigma'_{ij} \leqslant (1 + \sigma_i + \sigma_j)\, \sigma_\bullet^{\frac{1}{4}},$$

where $\bar{\mathbf{u}}_{ij} \stackrel{\text{def}}{=} \mathbb{E}[\mathbf{B}_{\mathbf{u}_i} \mathbf{u}_j]$ is a new vector orthogonal to $\mathbf{u}_i$ and $\mathbf{u}_j$, while the noise related to the binding operation is integrated into σ'_{ij}.

Since an unbinding operation is simply a binding operation with a dual vector, the noise calculation is the same.

Finally, since bundling is a simple summation, additive noise standard-deviations simply add. The tricky point is the approximation of τ, as summarized here[17]:

$$\textstyle\sum_{i=1}^{I} \mathbf{x}_i = \tau_\bullet \mathbf{u}_\bullet + \nu(\sum_i \sigma_i), \tau_\bullet \stackrel{\text{def}}{=} \sqrt{\sum_k (\sum_{i,k_i=k} \tau_i)^2},$$

where $\mathbf{u}_\bullet$ is a new unary vector approximately orthogonal to the others, while $\tau_\bullet$ is the related magnitude. We thus will use $\tau_\bullet$ as the approximate value of a bundling belief, which is rather arbitrary at this stage but could easily be improved.

4.4. Other possible features

Similar considerations would easily allow us to implement the same approach at a macroscopic for the dual operator $^\sim$, related to the commutator operator $\mathbf{B}_\leftrightarrow$ and the composition operator $\oslash$ given in Appendix C. However,

---

[16] As made explicit in Appendix C, the binding of two independent vectors $\mathbf{y}$ and $\mathbf{x}$ is a random vector and we can write

$$\begin{aligned}
&\mathbf{B}_{(\tau_i \mathbf{u}_i + \nu(\sigma_i))} (\tau_j \mathbf{u}_j + \nu(\sigma_j)) && = \\
&\tau_i \tau_j \mathbf{B}_{\mathbf{u}_i} \mathbf{u}_j + \tau_i \mathbf{B}_{\mathbf{u}_i} \nu(\sigma_j) + \tau_j \mathbf{B}_{\nu(\sigma_i)} \mathbf{u}_j + \mathbf{B}_{\nu(\sigma_i)} \nu(\sigma_j) && = \\
&\tau_i \tau_j \mathbb{E}[\mathbf{B}_{\mathbf{u}_i} \mathbf{u}_j] + \tau_i \tau_j \nu(1/d^{1/4}) + \tau_i \sigma_j \nu(1/d^{1/4}) + \tau_j \sigma_i \nu(1/d^{1/4}) + \sigma_j \sigma_i \nu(1/d^{1/4}) && \simeq \\
&\tau_i \tau_j \bar{\mathbf{u}}_{ij} + (\tau_i \tau_j + \tau_j \sigma_i + \sigma_j \sigma_i) \nu(1/d^{1/4}) && \simeq \\
&\tau_i \tau_j \bar{\mathbf{u}}_{ij} + \nu\left(\underbrace{(|\sigma_i \sigma_j| + |\tau_i| \sigma_j + |\tau_j| \sigma_i)\, \sigma_\bullet^{1/4}}_{\sigma'_{ij}} \right), &&
\end{aligned}$$

up to the first order, since $\nu(\sigma)$ is a random vector of magnitude σ, while

$$\mathbb{E}[\mathbf{B}_{\mathbf{u}_i} \nu(\sigma_j)] = \mathbb{E}\left[\mathbf{B}_{\nu(\sigma_i)} \mathbf{u}_j\right] = \mathbb{E}\left[\mathbf{B}_{\nu(\sigma_i)} \nu(\sigma_j)\right] = 0,$$

because these random vectors correspond to centered random vectors.

[17]

$$\begin{aligned}
\textstyle\sum_i \mathbf{x}_i &= \textstyle\sum_i \tau_i \mathbf{u}_{k_i} + \nu(\sigma_i) \\
&= \textstyle\sum_k \tau_k \mathbf{u}_k + \nu(\sum_i \sigma_i)\ \tau_k \stackrel{\text{def}}{=} \sum_{i,k_i=k} \tau_i \\
&= \tau_\bullet \mathbf{u}_\bullet + \nu(\textstyle\sum_i \sigma_i)
\end{aligned}$$

with:

$$\tau_\bullet \simeq \sqrt{\textstyle\sum_k \tau_k^2}, \|\mathbf{u}_\bullet\| = 1,$$

since $\mathbf{u}_k$ are approximately orthogonal.

as explored in the previous section, unless explicitly used to define expressions, they a not useful in our case. This is also the case for the **i** identity element.

Interesting enough, deriving a macroscopic ersatz of complex instead of real VSA specification, seems straightforward, since algebraic relations easily generalizes to these case, as reviewed in Appendix C.

Furthermore, we consider binding/unbinding operations using the VTB algebra, but the same reasoning could also be performed for other binding/unbinding operations, including and especially for heavily calculated operators, including if a commutative binding operation is considered, yielding obvious additional reduction rules.

This is another argument to consider macroscopic algorithmic ersatz: while VTB-algebra or almost one order of magnitude heavier than, for instance, convolution operators ($O(d^{3/2})$ instead of $O(d\ \log(d))$), and this is also the case for other non-commutative binding operations (see Appendix C), the macroscopic simulation of both have similar computational costs.

4.5. Available implementation

We are thus in a position to propose an algorithmic ersatz of the usual VSA mesoscopic linear algebra calculations involving high-dimensional random vectors. This has been implemented and made available as public documented open-source code[18]. For the generation of a symbol, at a given level of belief τ and for a given level of first-order random normal noise with a standard deviation σ, the usual similarity, bundling, and binding operations are made available, and the data-structures made explicit here also[19].

5. Experimental results

5.1. Calibrating macroscopic simulation

Symbolic expression syntax. In order to input or output expression we use the usual JSON[20] syntax in a weak form[21], namely:
- Bundlings are represented by lists:

```
[symbol_1 ...].
```

- Bindings are represented by the construct:

```
{b y: symbol x: symbol },
```

where `b` stands for binding and is replaced by `u` for unbinding.
- Atomic symbols are represented by the construct:

```
{ name: symbol-name tau: tau-value sigma: tau-value },
```

`tau` and `sigma` being optional, while atomic symbols with `tau=1` and `sigma=0` are also represented by strings. At the implementation level, this requires no more than a few lines of code.

Symbolic derivation examples. In Table 2, four illustrative examples of symbolic reduction are given:
-The 1st one illustrates the canonical representation of bundling where, in this case, $[a, b]$ and $[b, a]$ have the same internal representation, while expansion of binding over bundling is performed for the reduced form.
-The 2nd one illustrates that bundling of bundling is flattened while empty bundling or singleton bundling is reduced, it also shows an example of *tau* and *sigma* computation.
-The 3rd and 4th ones illustrate binding/unbinding reduction and related belief computation including in complex expressions.

[18] https://line.gitlabpages.inria.fr/aide-group/macrovsa.html.
[19] Additional mechanisms of rule derivations are also present, beyond the present work.
[20] See https://www.json.org.
[21] See https://line.gitlabpages.inria.fr/aide-group/wjson.

[I]: `[ {b y: c x: [a b]} {b y: c x: [b a]}]`
[P]: `[ B_(c)([ a b ]_<1.4±0>)_<1.4±0.044> B_(c)([ a b ]_<1.4±0>)_<1.4±0.044> ]_<2±0.088>`
[R]: `[ B_(c)(a)_<2±0.062> B_(c)(b)_<2±0.062> ]_<2.8±0.12>`
[I]: `[ [{name: a tau: 0.5 sigma: 0.1} [ ]] {name: a tau: 0.5 sigma: 0.1}]`
[P]: `[ a_<1±0.2> ]_<1±0.2>`
[R]: `a_<1±0.2>`
[I]: `{b y: a x: {u y: a x: [ {name: c tau: 2 sigma: 0.1}]}`
[P]: `B_(a)(B_(a~)([ c_<2±0.1> ]_<2±0.1>)_<2±0.066>)_<2±0.065>`
[R]: `c_<8±0.23>`
[I]: `{b y: c x: {b y: c x: {b y: c x: {u y: c x: {u y: c x: {u y: c x: a}}}}}}`
[P]: `B_(c)(B_(c)(B_(c)(B_(c~)(B_(c~)(B_(c~)(a)_<1±0.031>)_<1±0.032>)_<1±0.032>)_<1±0.032>)_<1±0.032>)_<1±0.032>`
[R]: `a_<1±0.19>`

Table 2

Four symbolic reduction examples.
- The [I] line corresponds to the Input of the symbolic expression in weak JSON syntax.
- The [P] line corresponds to the Parsed expression.
- The [R] line corresponds to the Reduced expression.

Bundlings are represented as a list between `[ ]`, Binding using the usual syntax, while the $< \tau \pm \sigma >$ construct allows to specify the belief value, when not equal to the default $< 1 \pm 0 >$ value.

Binding magnitude verification In Table 3, we have verified another aspect of our formal developments: The VTB binding magnitudes. This is important, contrary to previous studies because we have introduced the notion of belief via the τ magnitude parameter. As formally derived $\|\mathbf{B_y}\,\mathbf{x}\| \simeq 1$ and $\|\mathbf{B_y}\,\mathbf{y}\| \simeq \sqrt{2}$, while other magnitudes have not been derived a-priory only observed numerically. If simulation is run at the mesoscopic level, these values matter, and computations must be re-normalized. If the simulation is run at the macroscopic level, the normalization assumption is always considered.

Dimension:	100	400	1024	2500	4096	10000
$\|\mathbf{B_y x}\|^2$	1±0.14	1±0.073	1±0.043	1±0.027	1±0.023	1±0.014
$\|\mathbf{B_y} \sim \mathbf{B_y x}\|^2$	2.1±0.51	2±0.25	2±0.15	2±0.095	2±0.081	2±0.049
$\|\mathbf{B_y y}\|^2$	2.1±0.2	2±0.1	2±0.059	2±0.039	2±0.033	2±0.02
$\|\mathbf{B_y} \sim \mathbf{B_y y}\|^2$	5.3±1.2	5.2±0.64	5.1±0.36	5.1±0.25	5.1±0.2	5.1±0.13

Table 3

Mesoscopic computation of VTB binding magnitudes, over $N = 1000$ samples: mean $\pm$ standard-deviation is shown. This allows to verify what has been derived in Appendix C, regarding this aspect.

Numerical noise estimation. In Table 4, the comparison between mesoscopic similarities measures of elementary expressions, and the related macroscopic prediction is reported. Noise of order of magnitude $O(1/d)$ for similarities and $O(1/d^{1/4})$ for binding has been considered. We observe that with respect to our simple 1st order and conservative derivations, noise is overestimated at the macroscopic level, and this overestimation increases with the dimension.

This overestimation increases almost logarithmically with the space dimension, and may partially be compensated by a rather simple rule of thumb, which could easily be improved in the future. This is especially important since macroscopic simulation is mainly useful when the space dimension is to be increased.

After numerical adjustment, it appears that an experimental rule of thumb of:

$$\sigma_\bullet = O(1/d) \simeq \underbrace{\frac{1}{1024\,d}}_{\sigma^0_\bullet} \simeq \underbrace{\frac{1}{1024\,(6\,log_{10}(d) - 11)\,d}}_{\sigma^1_\bullet}$$

leads to reasonable results: using $\sigma^0_\bullet$ allows one to maintain the standard-deviation over-estimation above almost 1 and below 5, for $d < 10^5$, while using rule of the thumb defined by $\sigma^1_\bullet$ allows one to maintain the standard-deviation over-estimation 0.75 and below 2, for $d < 10^5$, which corresponds to the result in Table. 4.

This allows us to conclude from these numbers that our rather simple, 1st order, estimation of mesoscopic noise, allowing to predict it at the macroscopic level, being conservative, we observe an expected overestimation of the noise, which increases with the space dimension. Obtaining such an overestimation is a conservative choice, in the sense that some deductions may be missed, whereas false deductions will be avoided, as discussed in Appendix A.

A step further, we have investigated to what extents the usual approximation, here and in the literature, of chi-square distribution by a normal distribution, regarding the dot-product similarity operation is appropriate. This is shown in Fig. 5. The main result is that this difference is below 1 bit (i.e., we miss less than 1 bit of information, namely about half a bit, to consider it as a normal distribution instead of a combination of chi-square ones). This might decrease with the dimension, although the result is not obvious.

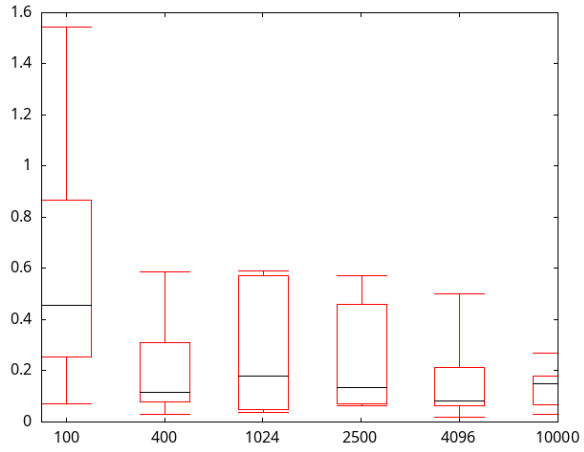

Fig. 5. Average divergence in bits (i.e., using log_2 in the formula) between the observed mesoscopic noise distribution and a normal distribution with the same standard-deviation, as a function of the space dimension.

Another interesting aspect is that our macroscopic model is consistent with that of [33], which obtains the following (from Fig. 4 of that paper) for the VTB representation:

$$d \gtrapprox 32\,(s + 0.575)$$

represents the minimal dimension d needed to obtain a 99% accuracy with a bundling of size s, using a similarity calculation to extract vectors from the bundling. Our model does not take the negligible bias 0.575 into account but allows us to calibrate the level of noise to $\sigma \simeq \frac{0.016}{d}$, in order to perform a simple z-score test under the normal hypothesis

$$\tau > 2\,\sigma$$

to decide if the related τ value of the similarity is distinguishable from the noise. On the other hand, we do not consider two vectors to be similar if the similarity is below the standard deviation of the noise, preferring a more conservative threshold.

To take this a step further, we also implemented the $(1/d^{1/4})$ noise dependency for unbinding, with the same calibration.

Studying, at the mesoscopic level, the numerical precision of unbinding on associative maps is to the best of our knowledge, this was not studied in the papers quoted in this one.

5.2. Benchmarking macroscopic simulation

In order to benchmark this implementation beyond simple tests, let us consider two rather large VSA existing experiments and test to what extents this can be simulated at the macroscopic level. We have taken into account

Dimension:	100	400	1024	2500	4096	10000
Mesoscopic bias and standard-deviation estimation over 1000 samples						
$\mathbf{B_y x} \cdot \mathbf{x}$	0.00041±0.1	-0.002±0.052	-7.9e-05±0.031	-0.00014±0.02	-0.00062±0.015	0.00061±0.0097
$\mathbf{B_y} \sim \mathbf{B_y x} \cdot \mathbf{x}$	-0.0007±0.24	-0.0068±0.11	0.004±0.071	0.0016±0.042	-0.0002±0.035	0.0013±0.022
$\mathbf{B_y y} \cdot \mathbf{y}$	0.0005±0.14	0.0014±0.072	0.0011±0.045	0.00016±0.028	0.00025±0.023	0.0006±0.014
$\mathbf{B_y} \sim \mathbf{B_y y} \cdot \mathbf{y}$	-0.0007±0.24	-0.0068±0.11	0.004±0.071	0.0016±0.042	-0.0002±0.035	0.0013±0.022
Macroscopic prediction of $\tau \pm \sigma$ (i.e. standard-deviation)						
$\mathbf{B_y x} \cdot \mathbf{x}$	1±0.11	1±0.081	1±0.063	1±0.051	1±0.045	1±0.036
$\mathbf{B_y} \sim \mathbf{B_y x} \cdot \mathbf{x}$	1±0.23	1±0.16	1±0.13	1±0.1	1±0.09	1±0.072
$\mathbf{B_y y} \cdot \mathbf{y}$	1±0.11	1±0.081	1±0.063	1±0.051	1±0.045	1±0.036
$\mathbf{B_y} \sim \mathbf{B_y y} \cdot \mathbf{y}$	1±0.23	1±0.16	1±0.13	1±0.1	1±0.09	1±0.072
Standard-deviation over-prediction						
$\mathbf{B_y x} \cdot \mathbf{x}$	1.1	1.5	2	2.6	2.9	3.7
$\mathbf{B_y} \sim \mathbf{B_y x} \cdot \mathbf{x}$	0.98	1.5	1.8	2.4	2.6	3.2
$\mathbf{B_y y} \cdot \mathbf{y}$	0.82	1.1	1.4	1.8	2	2.5
$\mathbf{B_y} \sim \mathbf{B_y y} \cdot \mathbf{y}$	0.98	1.5	1.8	2.4	2.6	3.2

Table 4: Mesoscopic versus macroscopic noise estimation.
- The 1st bloc corresponds to mesoscopic bias and standard-deviation estimation over 1000 samples, of basic expression similarities, at different vector space dimensions. The mesoscopic noise standard-deviation is estimated by computing dot-products over $N = 1000$ samples.
- The 2nd bloc corresponds to the macroscopic prediction of the mesoscopic standard-deviation estimation, as developed in this paper.
- The 3rd bloc shows the ratio between macroscopic/mesoscopic standard-deviation, showing the overestimation, while the order of magnitude is preserved.

the [44] King John Bible (KJB) data set[22]. This yields to about 10^6 input tokens for about 10^4 terms and about 10^3 documents. We did not reproduce the whole experiments, but simply have benchmarked our implementation storage and retrieval with such a rather large scale data set. Numerical results are not expected to be similar, since the [44] KJB data set pre-processing differs from the present one and since, while the [29] considers another data set.

The [44] experiment makes use of a search engine that needs to be able to assess similarity between terms and documents.

Regarding document similarity, thanks to the introduction of the τ parameter, now used in an extended manner and representing a level of activity, one implementation is to simply consider or each document a weighted binding and compute similarities as the binding vectors similarity:

$$v_{document} = \sum_{words} \tau_{word\ count}\, s_{word}$$

This is obviously implemented by adding each word of the document sequence to the document vector $v_{document}$, and this precisely corresponds to the [44] weight function, namely the column marginal value of the term-document matrix. The term-document matrix could also be represented with VSA mechanisms considering an associative-network, as reviewed in subsection B.3, but there is no need at the present stage.

Regarding word similarity, the dual obvious mechanism:

$$v_{word} = \sum_{documents} \tau_{count\ in\ document}\, s_{document}$$

namely the row marginal value of the term-document matrix. Then [44] proposes the following algorithmic ersatz : "term vectors can be compared with one another, and the space can be searched for the nearest neighbors of any given term´´. It is an "ersatz" in the sense that it is not implemented by VSA structures and connections. However, there exists a biologically plausible mechanisms at the microscopic level as reviewed in section B.1.2, which is made available in our implementation. Formally we obtain:

$$v_{word\ neighborhood} = \sum_{words} (v_{word}^T\, v_{word})\, B_{\nu_i}\, \mathbf{s}_{word},$$

where ν_i is a known ordinal symbol, this indexed list being sorted in decreasing $(v_{word}^T\, v_{other\ word})$ values.

Then, given a word, e.g. `fire` or `water` as calculated in Table 2 of [44], we can compute its neighborhood and obtain in our case the results reported in Table 5. Normalized values are presented, which is easily at a biologically plausible level in neural network computations using gain control, for instance using impedance adaptation [25].

word	τ	σ	**word**	τ	σ
`fire`	1.00	$2.90\,10^{-05}$	`water`	1.00	$1.63\,10^{-05}$
`burn`	0.40	$1.56\,10^{-05}$	`wash`	0.44	$7.26\,10^{-06}$
`chapter`	0.38	$7.10\,10^{-05}$	`toucheth`	0.44	$4.48\,10^{-06}$
`out`	0.38	$4.83\,10^{-05}$	`bathe`	0.42	$2.27\,10^{-06}$
`offer`	0.37	$1.52\,10^{-05}$	`clothes`	0.40	$9.94\,10^{-06}$
`savour`	0.36	$9.21\,10^{-06}$	`issue`	0.37	$2.43\,10^{-06}$
`day`	0.34	$4.01\,10^{-05}$	`unclean`	0.36	$5.70\,10^{-06}$
`among`	0.34	$4.00\,10^{-05}$	`uncleanness`	0.36	$6.52\,10^{-06}$
`even`	0.34	$4.41\,10^{-05}$	`copulation`	0.35	$1.48\,10^{-06}$
`burnt`	0.33	$1.72\,10^{-05}$	`until`	0.35	$1.84\,10^{-05}$

Table 5

Similarity between words as obtained using the macroscopic VSA implementation on the KJB data set.

This result differs from Table 2 of [44] because different "non-significant´´ words have been eliminated during pre-processing. We however, find expected associations such as `fire - burn` or `water - wash`.

[22] We have considered from https://www.kingjamesbibleonline.org the open PDF document, have segmented each chapter from the table of contents (treating each chapter as a document), and apply normalization and tokenization rules as detailed in the pre-processing available documentation. We obtained 1363 documents, 15085 different words and 2701846 words total, to be compared with the 1189 documents and 12818 different words reported by [44] using a different pre-processing: The order of magnitudes are similar, while the pre-processing used in the former case is more selective.

prefix	tail	τ^2
out land	egypt	91
spake moses	saying	77
if any	man	77
thus saith	hosts	76
therefore thus	saith	63
thus saith	behold	59
our jesus	christ	57
word came	saying	55
say thus	saith	52
saying thus	saith	51

$\tau = 1.1 \pm 0.8 \in [1, 91] \approx \Gamma(degree = 2, rate = 0.5, mode = 0.5)$

$\#\{\tau, \tau = 1\} \simeq 93\%, \quad \#\{\tau, \tau \leqslant 4\} > 99\%$

prefix	tail	τ
word came saying	man	34
written book chronicles	kings	34
years old began	reign	33
old began reign	reigned	32
praise exalt above	ever	31
bless praise exalt	above	31
spake moses saying	speak	30
chapter spake moses	saying	28
brought out land	egypt	24
forth out land	egypt	23

$\tau = 1.03 \pm 0.36 \in [1, 34] \approx \Gamma(degree = 8, rate = 0.1, mode = 0.7)$

$\#\{\tau, \tau = 1\} \simeq 97\%, \quad \#\{\tau, \tau \leqslant 2\} > 99\%$

Table 6

Prefix tail occurrence count for prefixes of length 2 (on the left) and 3 (on the right). The ten highest values are shown in both cases. The τ distribution mean, standard-deviation, Gamma-distribution approximation, and indications of value counts are reported.

	Macroscopic ersatz for any d (observed)	Mesoscopic calculation for $d = 1024$ (observed)	Mesoscopic calculation for $d \simeq 10^5$ (interpolated)
Similarity	310.00	3.00	$\simeq$ 300.00
Bundling add	0.01	0.03	$\simeq$ 3.00
Associative map add	8.00	360.00	$\simeq$ 180000.00

Table 7

Average unary computation time of one operation in μs (micro-second) using a standard Intel©Core^{TM} i5-8265U CPU @ 1.6GHz x 8 processor. Macroscopic ersatz computation and mesoscopic calculation for $d = 1024$ times are experimentally observed, while mesoscopic calculation times for $d \simeq 10^5$ is interpolated (computing similarity or bundling at the mesoscopic level is a simple dot-product or sum which linearly scales with the dimension, while other computation requiring binding computation that scale at about $O(d^{1.35})$, as obtained in Appendix C.1). The $d \simeq 10^3$ corresponds to usual dimensions considered when computing VSA at the mesoscopic level with standard methods. The $d \simeq 10^5$ value correspond to biologically plausible dimensions of a neural map.

A step further, we follow [29] addressing the question of sequence encoding. In this work, the VSA representation is implemented using binary values, while we use real value here. The key work is to measure, given a short sequence of words as prefix, the related tails occurrence. Formally, we can consider the following associative map data structure:

$$p_l = \sum_{i+l}^{L} B_{\sum_{j=1}^{l} B_{\nu_j} w_{i-j}} w_i$$

where ν_j stands for independent symbols representing ordinal numbers, as developed in Appendix B.4, while w_i is the i-th word in a document. This structure is a simple combination of bindings and bundlings, thus easy implementable in the VSA framework. Then, if a for a given prefix with $\tau = 1$ the tail occurs several times in the related bundling, say N times, considering the macroscopic bundling τ combination rule we obtain $N = \tau^2$, allowing to directly estimate the prefix $\rightarrow$ tail occurrence statistics, reported in Table 6.

Such count mechanism is the basic mechanism of, for instance next word prediction in large language models (LLM), whereas although about $2.5\,10^5$ prefixes has been taken into account throughout the book, we are far from the data size of what is used from LLM.

These results are not of real interest by themselves, they just allow us to experimentally verify the usage of this macroscopic implementation, with a relatively large scale experiment. Regarding computation time we have observed the following experimental results on a standard laptop computer, reported in Table 7.

We observe that similarity is definitely faster at the mesoscopic level, which is expected because it is a simple dot-product, instead of requiring symbolic derivations. Then, for $d \simeq 10^3$ bundling calculation time have of the same order of magnitude at both mesoscopic and macroscopic scales, but in favor of macroscopic symbolic derivation, especially at higher dimensions. A step further, as soon as binding is involved, even at rather low VSA dimension

of $d \simeq 10^3$ macroscopic computation is faster (about 45 times), while mesoscopic calculations become heavily tractable (computation time of the results presented in Table 6 would require about 1 hour for $d \simeq 10^4$ and more than 1 day for $d \simeq 10^5$).

This provides a rather precise evaluation of when using VSA computations at a macroscopic scale is of interest.

5.3. A tiny illustrative application

To illustrate the use of macroscopic mechanism beyond basic formulae, we reconsider the example proposed in [24] which has been simulated at the mesoscopic level and is based on a minimal ontology in Fig. 6.

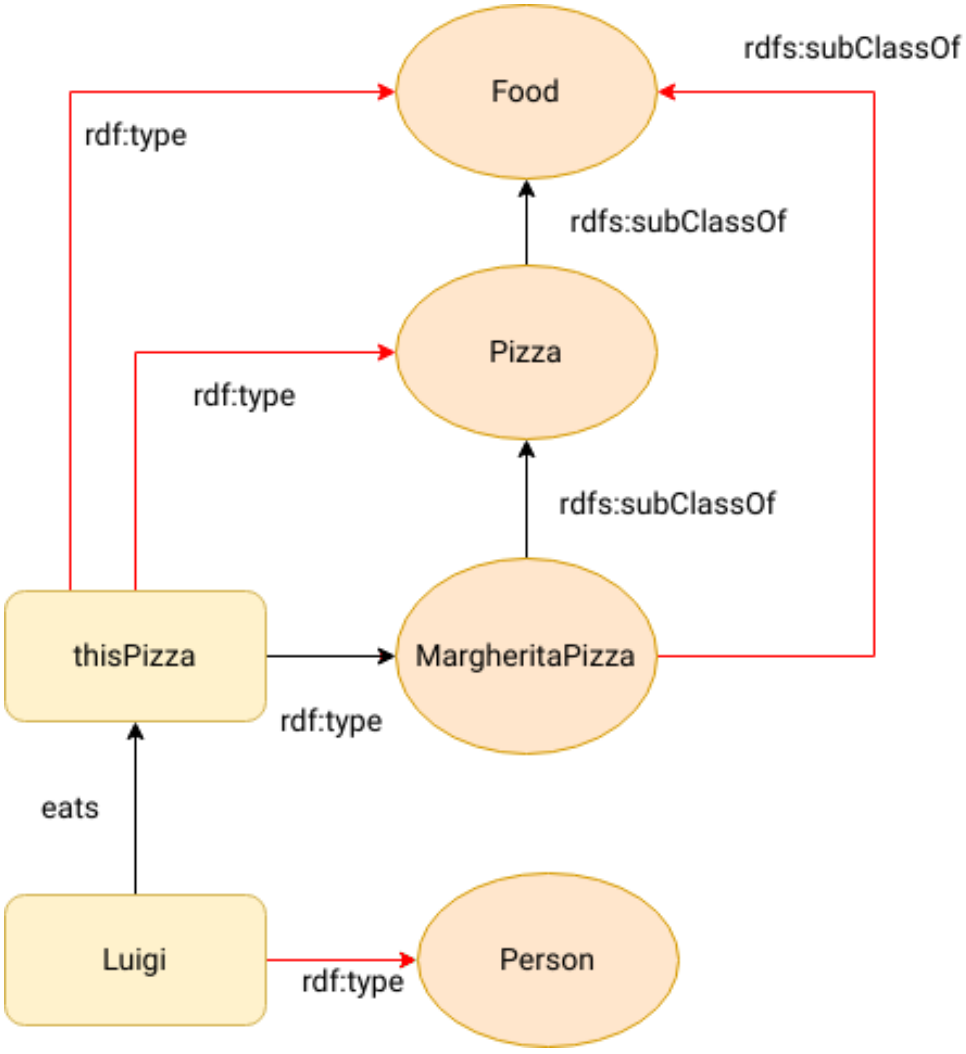

Fig. 6. An example of a simple ontology with three individuals. The black arrows correspond to factual statements input into the database and the green arrows correspond to inferred statements. Rectangular boxes stand for individuals, round boxes stand for classes, and properties are used to label arrows. Here, from the fact that a subject eats an object, we deduce that this subject is a Person, and the object is Food. This is illustrated by a red arrow. From the fact that the object is a Margherita pizza, which is a Pizza, which is a Food, according to the class hierarchy, we deduce that the object is a Pizza, and re-deduce that it is a Food. Furthermore, because Luigi (among other activities, since it is an open world) eats a pizza, we deduce that Luigi is a Person. Because of property heritage, meaning that here a Topping is an Ingredient, we also deduce from the fact that this pizza has mozzarella as a topping that it also has mozzarella as an ingredient. In the macroscopic implementation, this property is deduced from the fact that Margherita Pizza always has mozzarella as a topping, allowing us to generate compounded inferences. From [24].

We have considered a symbol encoding dimension of $d = 256$ to be consistent with previous mesoscopic experiments, such as those in [24]. This is tested with an associative map and relational map, as described in the previous section, and we have implemented the tiny Pizza experiment[23], obtaining, in the simplest case, the expected closure, as given in Fig. 7.

5.4. Macroscopic implementation of the VSA system

Input triples	*Inferred triples*
`(Luigi eats thisPizza)`	`(Luigi rdf:type Person)`
`(thisPizza rdf:type MargheritaPizza)`	`(thisPizza rdf:type Pizza)`
`(MargheritaPizza rdfs:subClassOf Pizza)`	`(thisPizza rdf:type Food)`
`(Pizza rdfs:subClassOf Food)`	`(MargheritaPizza rdfs:subClassOf Food)`

Fig. 7. The expected inferences using the proposed RDFS subset of entailment rules obtained by the macroscopic algorithmic ersatz of the VSA implementation.

More interesting is what happens when modality is considered, e.g.,

(Luigi 0.5 eats thisPizza).

In other words, it is possible but not completely necessary that Luigi eats the given pizza. In that case,
- it is still possible but no longer entirely true that Luigi is a person;
- it is still entirely true that this pizza is some food, even if Luigi did not eat it, because it is true that it is a pizza, which is food.

This is what is obtained by the implementation, as shown by the open-source tiny experiment output[23].

Although it is far from being complete, this macroscopic implementation of an algorithmic ersatz of VSA mesoscopic operations seems sound and it is consistent with previous results. It has a final non-negligible advantage: It is quite "simple" in the sense that it does not require very complicated or twisted mechanisms. It requires a bit more than 500 lines of formatted C++ code, including formal symbolic operations on the algebraic operators.

6. Discussion and conclusion

6.1. Contributions

In this paper, we have been able to propose, up to the implementation level, a reformulation of the powerful VSA approach with a few additions:
- We explicitized a degree of belief for each knowledge item that is linked to the possibility theory related to modal logic, and we revisited the main proposed abstraction of biologically plausible data structures to verify their compatibility with this generalization while comparing them with usual programming data structures and discussing how to efficiently scan (i.e., enumerate) such data structures.
- We proposed an implementation of hierarchical or relational semantic data structures within the VSA formalism in relation to hierarchical cognitive memory, allowing us to introduce symbolic derivations.
- We introduced the idea of simulating such a mechanism at a macroscopic, more symbolic level in order to obtain

[23] The source code is available at
https://gitlab.inria.fr/line/aide-group/macrovsa/-/blob/master/src/pizza_experiments.cpp
and it is noticeable that the C/C++ implementation of such rules is straightforward to write, as documented in the source code. This piece of code output is available at
https://gitlab.inria.fr/line/aide-group/macrovsa/-/raw/master/src/pizza_experiments.out.txt,
per Fig. 7, and it also shows the intermediate inference steps.

computations independent of the VSA dimension space, thus making it possible to scale up such mechanisms. This idea has been applied to VTB algebra but is also obviously reusable with other VSA algebras, though the macroscopic implementation is even more interesting for binding operations requiring more operations.

6.2. On biological plausibility and numerical versus semantic grounding

We thus propose an anchoring, i.e., a numerical grounding of semantic information. This, indeed, does not mean that the brain performs such operations as it, but this anchoring is biologically plausible in the sense that algorithms can be implemented in a VSA, as developed in this paper, which itself is a model of spiking neuron assembly activity, as developed by [13] with the NEF approach reviewed in this paper.

Numerical grounding, or anchoring, fundamentally differs from the semantic symbol grounding problem, as reviewed and discussed in [37], where symbols are linked to their meanings and anchored in sensorimotor features, which involves the capacity to pick referents of concepts and a notion of consciousness. In a nutshell, this is still an open problem that we are not going to address here. However, proposals of methods to link abstract symbols to neuronal reality enrich the issue of how mental states can be meaningful. Furthermore, the fact that our abstract representation is anchored in sensorimotor features means that it is also a link between symbols and their potential referents. To take this a step further, when we represent concepts, the chosen design choice associates prototypes, allowing us to anchor an abstract element to a concrete example.

Another aspect not targeted by the present study is the emergence of symbols, i.e., the fact that a symbolic representation emerges from a biological or any physical system in interaction with its environment. This issue corresponds to the ungrounding of concrete signs[24], as discussed in, e.g., [30], in relation to the emergence of symbolic thinking (see, e.g., [41] for a detailed discussion). At the computational neuroscience level, the issue is addressed in [31] for a toy experiment; that paper emphasizes that to address such an issue, we must avoid explicitly embedding any symbol anywhere in the model, a priory or a posteriory. Here, we do not address the emergence issue, but in a sense, we do address a *feasibility* issue: To what extent can sophisticated symbolic processing be anchored in numerical processing, not just rudimentary operators? We also address an *interpretation* issue, i.e., we consider to what extent sub-symbolic sensorimotor anchored processing corresponds to symbolic processing, as discussed later in this paper.

6.3. Approach limitations and perspectives

The macroscopic simulator is operational but still limited to basic VSA operations of bundling and binding, and their applications to set, associative maps, and other data structures detailed in Appendix B. The symbolic mechanisms are implemented directly at a procedural level, because relatively simple. More complex symbolic algebra could require specific software such as `Maple`[25] or equivalent ones, but at the cost of relying on close software or middle-ware and at the cost of a decrease of performances, because using very general mechanisms instead of tuned dedicated coding. Interestingly enough, the present implementation is easily generalized to other binding operators or other VSA mechanisms, and also to VSA in with complex numbers, as detailed in appendix C. As a next step, the present macroscopic package will be used to study biologically plausible inference mechanisms implemented with VSA.

At another level, we have targeted a representation related to ontology, which logical model is well established and the link with other frameworks based on relational representations, i.e. labeled graph at the geometric level, is well studied, including for frame-based semantics as discussed in this paper. The extension to hyper-graphs is less obvious. Claiming that an hyper-graph can always be represented as a bipartite graph between nodes and edges

[24] In the semiotic hierarchical meaning of an "icon" built only from sensorimotor features, structures at an "index" level built by concrete relationships between given objects give rise to a "symbol" in the semiotic sense, which corresponds to abstract general relationships between concrete concepts or sensorimotor features.

[25] See https://www.maplesoft.com.

is technically true and usable for some applications, but reductive[26] at the general level. The point is that hyper-graphs represent higher order knowledge, such inference rules for computing ontology justifications [45] or query processing [23] and allows to integrate real data bundles as studied by the same authors. We thus consider this as a limit of the present work.

At a more theoretical level, following the usual VSA approaches, the symbolic information is embedded in a compact Riemannian manifold with a very simple topology, a hyper-sphere, and we have made explicit the fact that finally, the number of encodable symbols is rather limited. Other geometries may offer better performances, and the particular hyperbolic embedding of hierarchical representations benefits from the fact that due to the hyperbolic negative curvature of the space, even an exponentially growing data structure can be parsimoniously represented [26] because of the expanding geometry (to make a long story short). The idea to embed the data representation in non-Euclidean spaces and especially hyperbolic spaces has already been explored in detail, for instance, in [9], showing that the satisfiability and algorithmic complexity can be drastically different[27]. This might be an interesting extension of typical VSA approaches that makes it possible to consider the symbol's numerical embedding in such a Riemannian differential manifold. This could be a fruitful perspective of such work.

Acknowledgments Terrence C. Stewart is gratefully acknowledged for his inspiring advice that helped us with some aspects of this work. Gabriel Doriath Döhler is thanked for his work on clarifying the use of VTB algebra and introducing interesting ideas during his undergraduate internship. Frédéric Alexandre and Hugo Chateau-Laurent are gratefully acknowledged for their valuable advice and their contributions to previous works on this subject. We are especially thankful for the reviewing work on this article, which deeply helped us to make the draft valuable. This work is supported by the https://team.inria.fr/mnemosyne/en/aide exploratory action. The NAI journal reviewers are gratefully acknowledged for precious advises allowing us to improve this paper.

Conflict of interest: This work is not subject to any conflict of interest.

References

[1] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T.C. Stewart, D. Rasmussen, X. Choo, A.R. Voelker and C. Eliasmith, Nengo: A Python tool for building large-scale functional brain models, *Frontiers in Neuroinformatics* **7** (2014).

[2] M. Beynon, B. Curry and P. Morgan, The Dempster–Shafer theory of evidence: An alternative approach to multicriteria decision modelling, *Omega* **28**(1) (2000), 37–50. https://www.sciencedirect.com/science/article/pii/S030504839900033X.

[3] B. Buchberger, G.E. Collins, R. Loos and R. Albrecht (eds), *Computer Algebra*, Computing Supplementa, Vol. 4, Springer, Vienna, 1983. ISBN 978-3-211-81776-6 978-3-7091-7551-4.

[4] S.T. Cao, L.A. Nguyen and A. Szałas, The Web Ontology Rule Language OWL 2 RL + and Its Extensions, in: *Transactions on Computational Intelligence XIII*, N.-T. Nguyen and H.A. Le-Thi, eds, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2014, pp. 152–175. ISBN 978-3-642-54455-2.

[5] B. Cessac and T. Viéville, On dynamics of integrate-and-fire neural networks with adaptive conductances, *Frontiers in Neuroscience* **2**(2) (2008). https://hal.inria.fr/inria-00338369.

[6] B. Cessac, H. Paugam-Moisy and T. Viéville, Overview of facts and issues about neural coding by spikes, *Journal of Physiology-Paris* **104**(1) (2010), 5–18. https://www.sciencedirect.com/science/article/pii/S0928425709000849.

[7] B. Cessac, H. Rostro-González, J.-C. Vasquez and T. Viéville, To which extend is the "neural code" a metric ?, arXiv, 2008, arXiv:0810.3990 [physics]. http://arxiv.org/abs/0810.3990.

[8] E. Crawford, M. Gingerich and C. Eliasmith, Biologically plausible, human-scale knowledge representation, *Cognitive Science* **40**(4) (2016), 782–821.

[26] Several hyper-graph VSA representations could be considered. In a nutshell, hyper-graph are defined by a set of nodes and labeled hyper-edges that are subset of nodes (in the non oriented case). It is thus defined as an associative map of label targeting set of nodes, all tools being available in the VSA framework. Formally, this could be written:

$$h \stackrel{\text{def}}{=} \sum_{hyperedges} B_{hyperedge\ symbol} \sum_{hyperedge\ nodes} s_{node\ symbol} \text{ and } h' \stackrel{\text{def}}{=} \sum_{nodes} B_{node\ symbol} \sum_{node\ edges} s_{hyperedge\ symbol},$$

allowing one to access each edge nodes, with a dual construction to access the edges of a symbol, and it is to verify that $h = B_{\leftrightarrow}\, h'$ with the notations of Appendix C.
However, this is the emerging part of the iceberg, because, as we did for the relational map representation, the hard part is to specify the expected operations on such data-structure

[27] A version of these elements intended for a wider audience is available in a science popularization journal: ..

[9] J.-P. Delahaye, *Complexités : Aux limites des mathématiques et de l'informatique*, HAL, 2006, Number: hal-00731936. https://ideas.repec.org/p/hal/journl/hal-00731936.html.

[10] T. Denœux, D. Dubois and H. Prade, Representations of uncertainty in AI: Beyond probability and possibility, in: *A Guided Tour of Artificial Intelligence Research: Volume I: Knowledge Representation, Reasoning and Learning*, P. Marquis, O. Papini and H. Prade, eds, Springer International Publishing, Cham, 2020, pp. 119–150. ISBN 978-3-030-06164-7.

[11] T. Denœux, D. Dubois and H. Prade, Representations of Uncertainty in AI: Probability and Possibility, in: *A Guided Tour of Artificial Intelligence Research: Volume I: Knowledge Representation, Reasoning and Learning*, P. Marquis, O. Papini and H. Prade, eds, Springer International Publishing, Cham, 2020, pp. 69–117. ISBN 978-3-030-06164-7.

[12] H. Eichenbaum, Memory: Organization and control, *Annual Review of Psychology* **68**(1) (2017), 19–45.

[13] C. Eliasmith, *How to Build a Brain: A Neural Architecture for Biological Cognition*, OUP, USA, 2013, Google-Books-ID: BK0YRJPmuzgC. ISBN 978-0-19-979454-6.

[14] C. Eliasmith and C.H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*, A Bradford Book, The MIT Press, 2002. https://mitpress.mit.edu/books/neural-engineering.

[15] B. Fischer, Modal Epistemology: Knowledge of Possibility & Necessity, 2018. https://1000wordphilosophy.com/2018/02/13/modal-epistemology/.

[16] R. Gayler, Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience, in: *Frontiers in Artificial Intelligence and Applications*, 2003, ICCS/ASCS International Conference on Cognitive Science.

[17] J. Gosmann and C. Eliasmith, Vector-Derived Transformation Binding: An Improved Binding Operation for Deep Symbol-Like Processing in Neural Networks, *Neural Computation* **31**(5) (2019), 849–869.

[18] B.C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider and U. Sattler, OWL 2: The next step for OWL, *Journal of Web Semantics* **6**(4) (2008), 309–322. https://www.sciencedirect.com/science/article/pii/S1570826808000413.

[19] B. Komer, T.C. Stewart, A.R. Voelker and C. Eliasmith, A neural representation of continuous space using fractional binding, in: *41st Annual Meeting of the Cognitive Science Society*, Cognitive Science Society, Montreal, Canada, 2019, p. 6. http://compneuro.uwaterloo.ca/publications/komer2019.html.

[20] H.J. Levesque, Knowledge Representation and Reasoning, *Annual Review of Computer Science* **1**(1) (1986), 255–287, Publisher: Annual Reviews.

[21] H.J. Levesque and R.J. Brachman, Expressiveness and tractability in knowledge representation and reasoning, *Computational Intelligence* **3**(1) (1987), 78–93, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8640.1987.tb00176.x.

[22] S.D. Levy and R. Gayler, Vector Symbolic Architectures: A New Building Material for Artificial General Intelligence, in: *Frontiers in Artificial Intelligence and Applications*, 2008, p. 6.

[23] M. Masmoudi, S.B.A.B. Lamine, H.B. Zghal, B. Archimede and M.H. Karray, Knowledge hypergraph-based approach for data integration and querying: Application to Earth Observation, *Future Generation Computer Systems* **115** (2021), 720–740, Publisher: Elsevier. https://hal.science/hal-04456331.

[24] C. Mercier, H. Chateau-Laurent, F. Alexandre and T. Viéville, Ontology as neuronal-space manifold: Towards symbolic and numerical artificial embedding, in: *KRHCAI-21@KR2021*, 2021.

[25] M.E. Nelson, A Mechanism for Neuronal Gain Control by Descending Pathways, *Neural Computation* **6**(2) (1994), 242–254.

[26] M. Nickel and D. Kiela, Poincaré Embeddings for Learning Hierarchical Representations, in: *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc., 2017. https://papers.nips.cc/paper/2017/hash/59dfa2df42d9e3d41f5b02bfc32229dd-Abstract.html.

[27] A. Nieder, Representation of Numerical Information in the Brain, in: *Representation and Brain*, S. Funahashi, ed., Springer Japan, Tokyo, 2007, pp. 271–283. ISBN 978-4-431-73021-7.

[28] F. Pulvermüller, How neurons make meaning: brain mechanisms for embodied and abstract-symbolic semantics, *Trends in Cognitive Sciences* **17**(9) (2013), 458–470. http://www.sciencedirect.com/science/article/pii/S1364661313001228.

[29] J.I. Quiroz Mercado, R. Barrón Fernandez and M.A. Ramírez Salinas, Sequence Prediction with Hyperdimensional Computing, *Research in Computing Science* (2025). https://www.academia.edu/54349431/Sequence_Prediction_with_Hyperdimensional_Computing.

[30] J. Raczaszek-Leonardi and T. Deacon, Ungrounding symbols in language development: implications for modeling emergent symbolic communication in artificial systems, in: *Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics*, 2018, p. 237.

[31] N.P. Rougier, Implicit and Explicit Representations, *Neural Networks* **22**(2) (2009), 155–160. https://hal.inria.fr/inria-00336167.

[32] A.L. Rusawuk, Possibility and Necessity: An Introduction to Modality, 2018. https://1000wordphilosophy.com/2018/12/08/possibility-and-necessity-an-introduction-to-modality/.

[33] K. Schlegel, P. Neubert and P. Protzel, A comparison of vector symbolic architectures, *arXiv:2001.11797 [cs]* **55** (2020). http://arxiv.org/abs/2001.11797.

[34] L. Smith, The development of modal understanding: Piaget's possibility and necessity, *New Ideas in Psychology* **12**(1) (1994), 73–87. https://www.sciencedirect.com/science/article/pii/0732118X94900590.

[35] J. Steinberg and H. Sompolinsky, Associative memory of structured knowledge, *Scientific Reports* **12**(1) (2022), 1–15, Number: 1 Publisher: Nature Publishing Group. https://www.nature.com/articles/s41598-022-25708-y.

[36] T.C. Stewart, Y. Tang and C. Eliasmith, A biologically realistic cleanup memory: Autoassociation in spiking neurons, *Cognitive Systems Research* **12**(2) (2011), 84–92. https://linkinghub.elsevier.com/retrieve/pii/S1389041710000525.

[37] M. Taddeo and L. Floridi, Solving the Symbol Grounding Problem: A Critical Review of Fifteen Years of Research, *Journal of Experimental and Theoretical Artificial Intelligence* **17** (2005).

[38] A. Tettamanzi, C.F. Zucker and F. Gandon, Possibilistic testing of OWL axioms against RDF data, *International Journal of Approximate Reasoning* **91** (2017). https://hal.inria.fr/hal-01591001.
[39] P.V. Tymoshchuk and D.C. Wunsch, Design of a K-Winners-Take-All Model With a Binary Spike Train, *IEEE Transactions on Cybernetics* **49**(8) (2019), 3131–3140, Conference Name: IEEE Transactions on Cybernetics. https://ieeexplore.ieee.org/abstract/document/8417947?casa_token=vK30j2BWdHYAAAAA:RcZjyJ1BCa6QE53oMhLBbVuLzetpGSh4FaFtSqu1RXsuqa82umuuPCzEm6KODKTGDBzIfUhP9_I.
[40] T. Vallaeys, Généraliser les possibilités-nécessités pour l'apprentissage profond, Report, Inria, 2021. https://hal.inria.fr/hal-03338721.
[41] T.D. Villiers, Why Peirce Matters: The Symbol in Deacon's Symbolic Species, *Language Sciences* **29**(1) (2007), 88–101. https://philarchive.org/rec/DEVWPM-4.
[42] T. Viéville and C. Mercier, Representation of belief in relation to randomness, Research Report, RR-9493, Inria & Labri, Univ. Bordeaux, 2022. https://hal.inria.fr/hal-03886219.
[43] A. Voelker, E. Crawford and C. Eliasmith, Learning large-scale heteroassociative memories in spiking neurons, in: *Unconventional Computation and Natural Computation*, 2014.
[44] D. Widdows and T. Cohen, Reasoning with Vectors: A Continuous Model for Fast Robust Inference, *Logic Journal of IGPL* (2014).
[45] H. Yang, Y. Ma and N. Bidoit, Hypergraph-Based Inference Rules for Computing $$\mathcal{EL}\mathcal{}^+$$-Ontology Justifications, in: *Automated Reasoning*, J. Blanchette, L. Kovács and D. Pattinson, eds, Springer International Publishing, Cham, 2022, pp. 310–328. ISBN 978-3-031-10769-6.

Appendix A. Hypothesis testing regarding symbol similarity

The design choice of a symbol implementation as a random normal unary vector, as reviewed in subsection 2.1 allows us to define a hypothesis to decide whether the $\mathcal{H}_0$ hypothesis $\mathbf{x} \cdot \mathbf{y} = 0$ can be rejected.

In our case, we approximate the chi-square distribution average of $\mathbf{x} \cdot \mathbf{y}$ by a normal distribution of the same standard-deviation, which is a conservative choice as shown in Fig. 8.

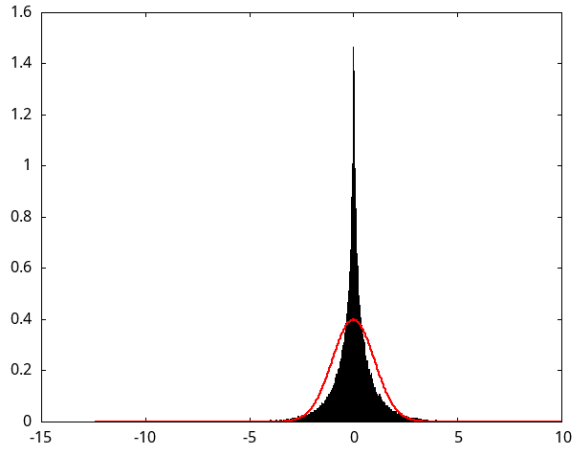

Fig. 8. Comparison between a chi-square distribution in black (numerically drawn from 10^5 samples) and the normal distribution of same mean and standard-deviation, in red: The choice is conservative because much more samples have values close to zero in the former case. More precisely the kurtosis is of about 6 (i.e. the sharpness estimation with respect to a normal distribution using 4-th order momenta).

We can consider a two-tailed "z-test" with the alternative hypothesis $\mathcal{H}_0$, which states that $\mathbf{x} \cdot \mathbf{y} \neq 0$. Here, the z-score[28], with d samples and a known standard deviation with an order of magnitude $O(1/d)$, is the following:

[28] Given a distribution, the z-score for d samples is defined as

$$z \stackrel{\text{def}}{=} \frac{\bar{X}-\mu}{\sigma/\sqrt{d}},$$

where the expected mean is $\mu = 0$, the a priory standard deviation is $\sigma = O(1/d)$, and the experimental mean $\bar{X} = (\mathbf{x} \cdot \mathbf{y})$ is obtained from the dot product.

$$z \equiv \sqrt{d}\,(\mathbf{x} \cdot \mathbf{y}).$$

It follows an almost distribution, which can be easily verified numerically, as shown in Fig. 9 (left column). For two vectors that are not independent but have an angular dependency, we can numerically observe, in Fig. 9 (right column), the similarity dependency as a function of the vector's relative orientation. This obvious fact is quite important, allowing us to develop a macroscopic simulation of VSA operations.

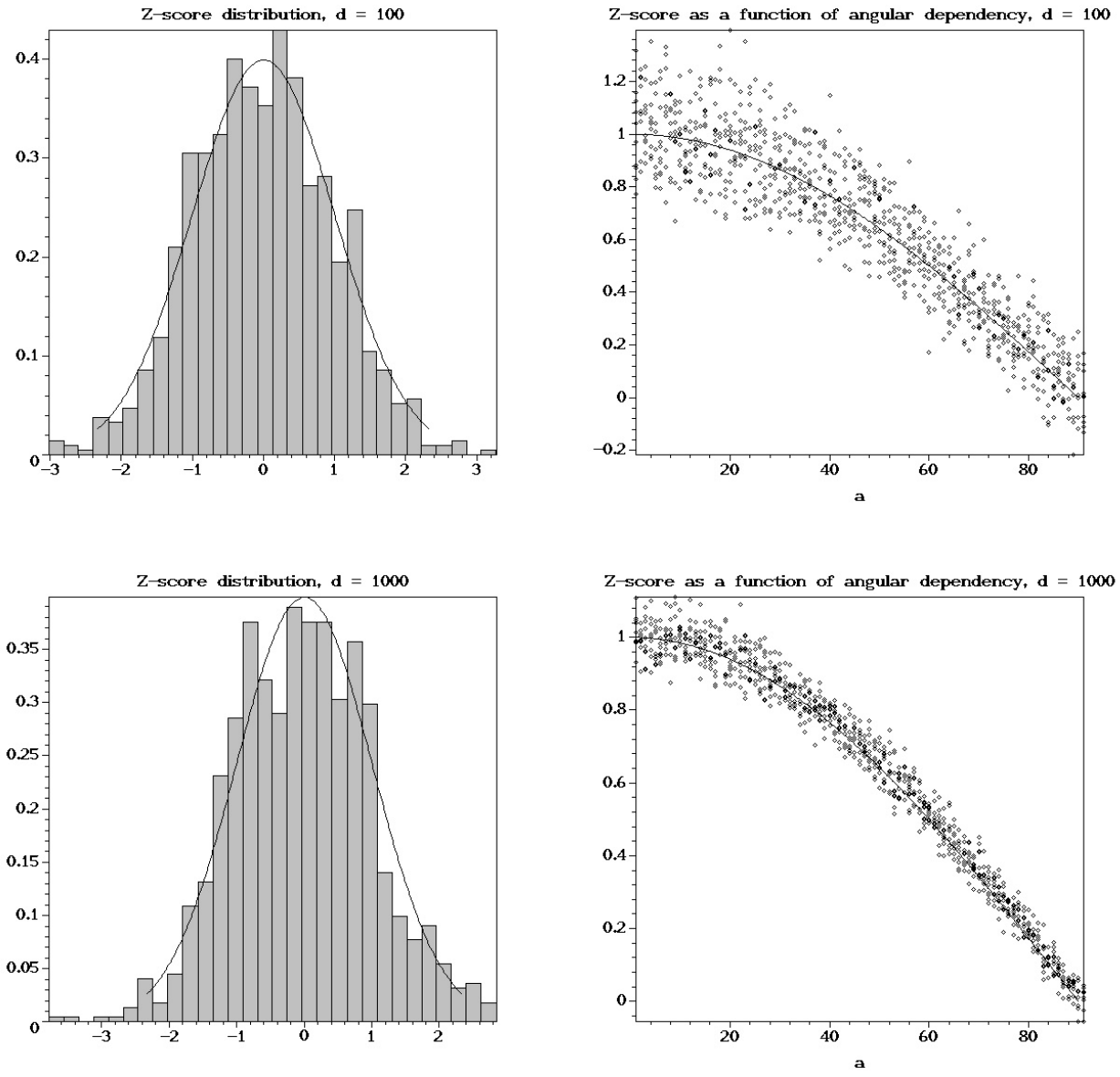

Fig. 9. Numerical observations of the similarity defined by the dot product of two random vectors for $d = 100$ in the upper row and $d = 1000$ in the lower row. The left column shows the histogram of the z-score ($\sqrt{d}\,(\mathbf{x} \cdot \mathbf{y})$) for two normal vectors, in comparison with a normal distribution. These experimental distributions have a kurtosis of about 10; this is lower than the kurtosis of a normal distribution, which is expected to have a kurtosis of 3. The right column shows the z-score as a function of the angle $a \stackrel{\text{def}}{=} \widehat{\mathbf{x},\mathbf{y}} = \arccos(\mathbf{x} \cdot \mathbf{y})$, making it possible to visualize the dispersion with respect to the expected cosine profile.

This makes it possible, on the one hand, to consider, for instance, a threshold:

$$\theta \stackrel{\text{def}}{=} \pm 2\,\sigma,$$

along with considering this z-score to have a confidence interval better than 99%, and to relate the similarity estimation to an angular dependence between two vectors, as detailed in Fig. 9. To the best of our knowledge, this obvious implementation has not yet been made explicit, and it is used in subsection 5.4, as detailed in section 4, allowing us to propose to simulate the different operations defined later in this paper at a macroscopic scale.

Appendix B. On VSA data structures

This section revisits the literature, emphasizing the properties of the data structures; it discusses in more detail their computational properties and limitations and links them to usual programming language data structures.

B.1. Unordered set or bundling

We first consider an unordered set $\mathcal{S}$ of N symbols grounded to values $\{\mathbf{s}_1, \cdots \mathbf{s}_i \cdots \mathbf{s}_N\}$, and we would like to be able to store them in such a way that we can check if a given symbol is in the set. Very simply, we ground $\mathcal{S}$ to the vector $\mathbf{s}$:

$$\mathbf{s} \stackrel{\text{def}}{=} \textstyle\sum_i \mathbf{s}_i,$$

which provides a solution, because given a symbol $\mathbf{s}_\bullet$, we observe that $\mathbf{s}_\bullet \cdot \mathbf{s} \simeq 1$ if it corresponds to a certain symbol $\mathbf{s}_i$, and it is almost 0 otherwise, because random vectors are almost orthogonal, as previously explained. This is called bundling [33] or superposition.

Furthermore, the representation intrinsically includes a notion of transitivity: If a set includes another subset, by construction, it includes the subset elements. More precisely,

$$\mathbf{s} \stackrel{\text{def}}{=} \textstyle\sum_i \mathbf{s}_i \text{ and } \mathbf{s}_i \stackrel{\text{def}}{=} \sum_j \mathbf{s}_{ij} \Rightarrow \mathbf{s} \stackrel{\text{def}}{=} \sum_{ij} \mathbf{s}_{ij},$$

and thus, $\mathbf{s}_{ij} \cdot \mathbf{s} > 0$ for all subset elements.

This generalizes to weighted symbols $\hat{\mathbf{s}}_i$, i.e., symbols with modality weighting. In that case, $\mathbf{s}_\bullet \cdot \mathbf{s} \simeq \tau$ makes it possible to retrieve the belief weight. This is equivalent to inputting a symbol $\mathbf{s}_\bullet$ that is approximately similar to a given symbol $\mathbf{s}_i$, thus indicating an approximate similarity; however, it neither allows us to retrieve the exact value of $\mathbf{s}_i$ nor indicates if a positive value below 1 corresponds to a weighted symbol that has been exactly retrieved or to a symbol approximation.

This has an interesting biological interpretation: $\mathcal{S}$ has features in common with a Hopfield network or other related attractor networks, where information has been stored in a distributed way while activating the map with an input makes it possible to determine whether the symbol is stored or not. This is also called self-associative. The main difference with auto-associative, or clean-up memory, is that in the Hopfield network attractor networks converge to the exact stored value, providing a mechanism of associative memory, which is now developed while introducing superposition, allowing us to better understand the need for a more sophisticated mechanism.

Let us provide an analogy with programming data structures, explicitizing the similarities and differences between what is proposed here and what is available in common programming languages[29]. This unordered set representation corresponds to a "set" container (e.g., a `std::unordered_set` in C++ or a `set()` in Python) that has only an insertion method and a membership test function, without the capability to intrinsically enumerate the elements, as formerly discussed.

B.1.1. Symbol enumeration

At this stage, this structure does not allow us to directly enumerate all symbols $\mathbf{s}_i$, because from $\mathbf{s}$, it is not possible to decode the superposed vectors. In [8], for instance, where data structures are defined using superposition, the intrinsic memory enumeration of the stored information is not addressed. We thus need an external mechanism to

[29] We will do the same for other cognitive structures because we think that it illustrates the computing capability of the cognitive object.

select all elements and perform an operation on each one. However, at the implementation level, in NENGO [13], an explicit list of the defined vocabulary $\{\cdots \mathbf{s}_i \cdots\}$ is maintained, and the way to select the elements is to test $(\mathbf{s}^T \mathbf{s}_i)$ for each element of the vocabulary. This select operator has a complexity of $O(K)$, where K is the size of the vocabulary. Later in this section, we will also propose a biologically plausible indexing mechanism, in order, for instance, to manipulate sequences.

B.1.2. Symbol sorting selection

With the notion of weight vectors by a τ value, another mechanism can be considered in a bundling, either sorting symbols in decreasing τ value order, or the $N' <= N$ symbols with the highest τ. Although, this has a complexity of $O(N\log(N))$ at the programmatic level, or $O(N)$ if selecting without sorting symbols above a threshold, there is a very efficient not at the mesoscopic, but at the microscopic level [6] in link with rank coding [7], when implementing as a spiking neural network. In a nutshell, following [39], given two values encoded by spikes, the highest the value, the shortest the spike time: Therefore, assuming a connectivity where each value is stored, in the arrival order using a triggered memory, we immediately (i.e., at the end of the emission of the N' values) a sorting selection. This is made available at the macroscopic level, taking into consideration the fact that two values are indistinguishable if too close, as discussed and quantified in this paper.

Formally this performs the transformation:

$$\sum_i \tau_i \, \mathbf{s}_i \quad \rightarrow \quad \sum_i \tau_i \, B_{\nu_i} \, \mathbf{s}_i$$

from a weighted bundling to an indexed list, ν_i being a known vector as developed below in subsection B.4.

B.2. Associative map

We now consider an unordered associative memory, or "map," of N correspondences $\{\mathbf{s}_1 \rightarrow \mathbf{o}_1, \cdots \mathbf{s}_i \rightarrow \mathbf{o}_i \cdots \mathbf{s}_N \rightarrow \mathbf{o}_N\}$ between subjects and objects. To this end, we use the binding operation $B_{\mathbf{s}_i}$, defined in Appendix C, with a pseudo-inverse, i.e., an unbinding operator, $B_{\mathbf{s}_i^\sim}$:

$$\mathbf{m} \stackrel{\text{def}}{=} \sum_i B_{\mathbf{s}_i} \, \mathbf{o}_i,$$

so that

$$B_{\mathbf{s}_\bullet^\sim} \, \mathbf{m} \simeq \sum_{i,\mathbf{s}_\bullet = \mathbf{s}_i} \mathbf{o}_i + \mathbf{unknown},$$

where $\mathbf{unknown} \stackrel{\text{def}}{=} \sum_{\mathbf{s}_\bullet \neq \mathbf{s}_i} B_{\mathbf{s}_\bullet^\sim} \, B_{\mathbf{s}_i} \, \mathbf{o}_i$ is an "unknown´´ vector, i.e., a vector that has, in the general case, no similarity with the other vectors.

In other words, the unbinding operation makes it possible to retrieve up to an orthogonal vector **unknown** the set, i.e., the additive superposition of all objects $\mathbf{o}_i$ associated with a given subject $\mathbf{s}_\bullet$, while $B_{\mathbf{s}_k^\sim} \, \mathbf{m} = \mathbf{unknown}$ if none.

Then using a similarity operation:

$$B_{\mathbf{s}_\bullet^\sim} \, \mathbf{m} \cdot \mathbf{o}_\bullet = \delta_{\mathbf{s}_\bullet \rightarrow \mathbf{o}_\bullet} + \nu(O(1/d^{1/4}))$$

allows to check whether the value $\mathbf{o}_\bullet$ is associated to the key $\mathbf{s}_\bullet$. Here $\nu(O(1/d^{1/4}))$ comes from the unbinding operation uncertainty.

This is done up to a level of noise of $O(1/d^{1/4})$, as derived in Appendix C (while the dot-product with the unknown vector is of negligible magnitude with respect to the former source of uncertainty), which is rather high with respect to the similarity precision, which is $O(1/d)$, as observed numerically [33]; however, in biological neuronal networks, where the dimension is an order of magnitude higher, this is no longer a limitation because d is high.

This allows us to detect if the information is in the table in one step if this is the case. However, as in the previous case, no mechanism allows to explicitly retrieve the value of $\mathbf{o}_i$, or to generate the enumeration of the map subjects or objects.

In the literature, the notion of clean-up memory corresponds to auto-associative memory that retrieves an exact "clean-up´´ value of an existing symbol, given an approximate or noisy input of this symbol [36]. A step further, the notion of hetero-associative memory corresponds to storing input-output relationships [43]. It must be noted that in the NENGO simulator of the Neural Engineering Framework (NEF) hetero-associative memory, the biologically plausible implementation is not directly based on the present binding/bundling algebraic mechanism but an

input/output architecture with suitable connections is used instead. Each input unit has an encoding vector in which input weights are tuned to fire for a specific key and drive a connected output vector that is optimized to estimate the value associated with the related key [43]. An associative memory of structured knowledge has been studied in detail, for a holographic reduced representation by [35], quantifying, depending on the design parameters, the memory performances.

This algebraic construction also makes it possible to retrieve the subjects associated with a given object, because of the commutator $\mathbf{B}_{\leftrightarrow}$, such that

$$\mathbf{B}_{\leftrightarrow}\,\mathbf{B}_{\mathbf{o}_i}\,\mathbf{s}_i = \mathbf{B}_{\mathbf{s}_i}\,\mathbf{o}_i,$$

yielding

$$\mathbf{m}_{\leftrightarrow} \stackrel{\text{def}}{=} \mathbf{B}_{\leftrightarrow}\,\mathbf{m} = \textstyle\sum_i B_{\mathbf{o}_i}\,\mathbf{s}_i,$$

which is now the numerical grounding of the reciprocal map $\{\mathbf{o}_1 \to \mathbf{s}_1, \cdots \mathbf{o}_i \to \mathbf{s}_i \cdots \mathbf{o}_N \to \mathbf{s}_N\}$.

The algebraic construction also offers the notion of the identity vector $\mathbf{i}$, with $\mathbf{B}_{\mathbf{i}} = \mathbf{I}$, so that

$$\mathbf{s}_i = \mathbf{i} \to B_{\mathbf{s}_i}\,\mathbf{o}_i = \mathbf{o}_i.$$

In other words, the binding reduces to a superposition. Theoretical details underlying the implementation of such associative memories are available in [36].

As for the previous structure, this obviously generalizes to weighted symbols $\hat{\mathbf{s}}_i$ and an approximate input $\mathbf{s}_\bullet \simeq \hat{\mathbf{s}}_i$, allowing us to retrieve the object $\mathbf{o}_i$ weighted by either the modality weighting or the input approximation, indistinctly.

There are several solutions used to define such binding, unbinding, and commutator operators. A proposed solution is developed in Appendix C after the work in [17], which was completed by [24]. This design choice is guided by the fact that we need to avoid spurious inferences: With a binding commutative operator (such as the convolution operator), $\mathbf{B}_{\mathbf{o}_i}\,\mathbf{s}_i$ would equal $\mathbf{B}_{\mathbf{s}_i}\,\mathbf{o}_i$, which could generate nonsense deductions (e.g., for a driver-vehicle map, this would mean that if Ming-Yue drives a bicycle, then the bicycle drives Ming-Yue unless some additional mechanism is considered to avoid such nonsense). The proposed VTB algebra avoids such caveats (see [33] for a recent comparison of different VSAs)[30]. A commutative binding operator can be used for auto-associative memory [36] or if key and values do not belong to the same symbol set, e.g., considering "symbol id´´ and "symbol value´´ [8].

This associative memory mechanism has an interesting biological interpretation: It implements an associative memory in the biological sense, with the association stored in a distributed way, and activating the associative memory with an input $\mathbf{s}_\bullet$ allows us to retrieve the associated symbol. This is what happens in several biological mechanisms, as reviewed, for instance, in [12].

In particular, a structure of the form

$$\mathbf{m} \stackrel{\text{def}}{=} \textstyle\sum_n B_{\mathbf{s}_i}\,\mathbf{s}_i$$

that maps an object onto itself allows the retrieval of an exact symbol from an approximate input, solving the caveats induced by using only a superposition mechanism that was presented previously. This is exactly what is expected in an associative encoder (e.g., a Hopfield network); if a symbol is close to an existing symbol, the associative memory will output a weighted version of the symbol.

At the computer programming level, this corresponds to a "map" container (e.g., a `Map` in JavaScript or a `dictionary()` in Python), again with only insertion and retrieval methods, and without intrinsic iterators.

To take this a step further, we can propose a complementary functionality, defining an additional symbol "something" whose numerical grounding is fixed to any new random vector σ that is never used elsewhere. This allows us to enhance the information to be obtained as follows: Each time a piece of information $\mathbf{s}_i \to \mathbf{o}_i$ is added, we also add $\mathbf{s}_i \to \sigma$ and $\sigma \to \mathbf{o}_i$, i.e., we make explicit the fact that $\mathbf{s}_i$ and $\mathbf{o}_i$ are defined in this table, which can be retrieved in one step, without the need to enumerate the different elements. In such a case,

$$\mathbf{m}_{\mathbf{s}_j} \stackrel{\text{def}}{=} \textstyle\sum_{i,\mathbf{s}_j=\mathbf{s}_i} \mathbf{o}_i = P_{\sigma^\perp}\,B_{\mathbf{s}_i^\sim}\,\mathbf{m} + \mathbf{unknown},$$

[30] An alternative to VTB algebra is called MBAT algebra; it requires matrix inversion instead of transposition, and thus it is less efficient.

where $P_{\sigma^\perp} \stackrel{\text{def}}{=} \mathbf{I} - \sigma\,\sigma^T$ is the projection onto the orthogonal of σ, i.e., we must eliminate the symbol "something" from the expected values.

B.3. Associative network

We also can consider for N correspondences $\{\mathbf{s}_1 \to \mathbf{o}_1, \cdots \mathbf{s}_i \to \mathbf{o}_i \cdots \mathbf{s}_N \to \mathbf{o}_N\}$ between subjects and objects, an associative network of the form:

$$\mathbf{M} \stackrel{\text{def}}{=} \textstyle\sum_i \mathbf{o}_i\,\mathbf{s}_i^T,$$

which is no more a vector, but a matrix, allowing one to explicitly retrieve the object since:

$$\mathbf{M}\,\mathbf{s}_i = \|\mathbf{s}_i\|^2\,\mathbf{o}_i + \nu((N-1)/d),$$

as obtained from obvious algebra.

This corresponds to a macroscopic implementation of associative networks, as proposed, e.g. in [8]. With respect to the previous associative map, we recover directly the object value up to a scale and additive noise, beyond only testing if it is in the map or not.

It is worth noticing that $\mathbf{M}^T$ allows us to retrieve keys associated with a given value, while we also can consider a "multi-map" i.e. associate the bundling of several values to a given key.

B.4. Indexed and chained list

B.4.1. Construction of indexes

In order to define an indexed list, we need indexes, i.e., a mechanism that generates ordinal values. Our main purpose here is to make explicit that what has been developed using convolution operators [19] still holds with VTB. We fix the symbol grounding of the "zero" symbol ν_0, which is never used elsewhere, and define the following recursively:

$$\nu_{n+1} \stackrel{\text{def}}{=} B_{\nu_0}\,\nu_n,$$

i.e., the $(n+1)$-th ordinal value is obtained by binding the n-th, and we easily obtain, from a few algebra operations,

$$B_{\nu_p}\,\nu_q = B_{\nu_q}\,\nu_p = B_{\nu_{p+q}}\,\nu_0, \quad B_{\nu_p}\,\nu_q^{\sim} = B_{\nu_q^{\sim}}\,\nu_p \simeq B_{\nu_{p-q}}\,\nu_0.$$

In particular, $\nu_{n-1} \simeq B_{\nu_0^{\sim}}\,\nu_n$, so that the definition holds for $n \in \mathcal{Z}$.

Here, we only consider the minimal material needed to build an indexed list; numerical information in the brain is a much more complex subject [27] beyond the scope of this work.

In fact, what has not been noticed previously is the fact that the accumulation of binding operations leads to an important increase of noise, more precisely:

$$\nu_n = (B_{\nu_0})^n\,\nu_0 + \nu(O(n/d^{1/4}))$$

so that for as soon as $n^4 > d$ the noise order of magnitude is higher than 1, i.e., the value order of magnitude. However, this is not a problem in practice because as soon as we do not repeat the calculation twice, i.e., we do not redraw the values, but keep a trace of the previous pre-calculated values, so that each "number" has a unique vector as an identifier.

B.4.2. Indexed list

We can now define an indexed list or array, often called a vector, since the previous mechanism allows us to generate a "counter" that can be incremented or decremented using the binding or unbinding operator.

To this end, an associative map indexed by these ordinals can be managed as a list whose values can be enumerated. Such a representation is also present at several cognitive levels when considering temporal sequences, actions, or any enumeration. This is also the tool that allows us to enumerate all elements of a symbol set $\mathcal{S}$, which was defined previously, or the subjects of an associative map.

To make this mechanism explicit, let us consider a list $\mathbf{l} \stackrel{\text{def}}{=} \sum_i B_{\nu_i}\,\mathbf{l}_i$, and a variable index $\mathbf{k}$. A construct of the form

for $\mathbf{k} \leftarrow v_0$; **while** $\|B_{\mathbf{k}}\,\mathbf{l}\| > 0$; **next** $\mathbf{k} \leftarrow B_{v_0}\,\mathbf{k}$ **do**
  $\mathbf{l}_i \leftarrow B_{\mathbf{k}^\sim}\,\mathbf{l}$
  ../..
end for

allows us to enumerate[31] all elements, this being indeed only an algorithmic ersatz to illustrate the mechanism beyond the biologically plausible implementation of sequential memory organization.

At the biological plausibility level, following [12], we may consider that the brain can have three kinds of memory: associative, sequential, and hierarchical (called schematic by the author of [12]) memory. All three memory types are present and required for cognitive processes. The VSA approach provides both associative and sequential memory. Let us consider the third type of memory, which has not, to the best of our knowledge, been addressed concerning VSAs.

At the computer programming level, this corresponds to an extensible "array" (e.g., a `std::vector` in C++ or `java.util.AbstractList` in Java), with basic edition and retrieval methods available.

B.4.3. Chained list

We can also define a chained list using an associative memory of the form:

$$\begin{array}{r} \texttt{first} \rightarrow \texttt{second} \\ \texttt{second} \rightarrow \texttt{third} \\ \cdots \\ \texttt{last} \rightarrow \texttt{END + first} \end{array},$$

where every value of the list acts as a key to the value of its successor in the list, thus enumerating the values. `END` is a predefined specific symbol that makes it possible to know when the list ends that we can superpose to a pointer to the first value in case we need to iterate through the entire list again.

We also could have considered multiple binding[32], as proposed in [24].

Appendix C. Using VTB algebra

At the mesoscopic level, symbols represent a numerical grounding to real or complex vectors of dimension d, with each numerical grounding corresponding to some distributed activity of a spiking neuronal assembly and each algebraic operation corresponding to some transformation of this activity.

Let us review and further develop one of the algebras used to manipulate such symbols at an abstract level in this paper: vector-derived transformation binding (VTB) algebra. We follow [17] and complete the developments in that paper by deriving the different operations at the component level, yielding an optimal implementation, and making explicit the computational complexity and related first-order noise. This is in particular used in section 4 to derive the macroscopic computations.

We also have to reconsider the binding output magnitude since vector magnitudes correspond to a belief value, as discussed in section 2.2.

We consider that $d \stackrel{\text{def}}{=} (d')^2$ for some integer d'; thus, it is a quadratic number and we start from the standard definition of the VTB binding operation:

$$\mathbf{z} \stackrel{\text{def}}{=} \mathbf{B}_{\mathbf{y}}\,\mathbf{x},$$

where $\mathbf{B}_{\mathbf{y}}$ is a block-diagonal matrix defined as follows:

[31] In fact, considering $\mathbf{l} \stackrel{\text{def}}{=} \sum_i B_{v_i}\,\mathbf{l}_i + B_{v_{-1}}\,\lambda$, where λ is the list length, which is updated when an element is added or deleted, would improve the algorithmic ersatz implementation, which is not the issue here.

[32] In such a case, a list of the form $l = [v_1, v_2, \cdots]$ is encoded without associative memory as

$$\mathbf{l} = B_{\texttt{value}}\,\mathbf{v}_1 + B_{\texttt{next}}\,\left(B_{\texttt{value}}\,\mathbf{v}_2 + B_{\texttt{next}}\,\left(\cdots + B_{\texttt{next}}\,(\texttt{list-end})\right)\right),$$

allowing us to obtain by unbinding the list's head value and its tail value, and allowing us to detect its end. This corresponds, for instance, to the `rdf:first`, `rdf:rest`, and `rdf:nil` symbols of the RDF representation. However, as discussed in Appendix C, chaining unbinding operations are not numerically very robust due to the additional residual noise.

$$\mathbf{B}_\mathbf{y} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{B}'_\mathbf{y} & 0 & \dots & 0 \\ 0 & \mathbf{B}'_\mathbf{y} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{B}'_\mathbf{y} \end{bmatrix}, \text{ with } \mathbf{B}'_\mathbf{y} \stackrel{\text{def}}{=} \sqrt{d'} \begin{bmatrix} y_1 & y_2 & \cdots & y_{d'} \\ y_{d'+1} & y_{d'+2} & \cdots & y_{2d'} \\ \vdots & \vdots & \ddots & \vdots \\ y_{d-d'+1} & y_{d-d'+2} & \cdots & y_d \end{bmatrix},$$

or equivalently[33], for $i = 1 \cdots d$,

$$\begin{cases} [\mathbf{z}]_i \stackrel{\text{def}}{=} \mathbf{B}_\mathbf{y}\, \mathbf{x} = \sqrt{d'} \sum_{k=1}^{k=d'} [\mathbf{y}]_{k+\beta(i)}\, [\mathbf{x}]_{k+\alpha(i)}, \\ \quad [\mathbf{B}_\mathbf{y}]_{ij} = \sqrt{d'}\, \delta_{i \leqslant d'} \text{ and } {}_{j \leqslant d'} [\mathbf{y}]_{k+\beta(i)-\alpha(i)}, \\ \text{written} \begin{cases} \alpha(i) \stackrel{\text{def}}{=} d'\, ((i-1) \text{ div } d'), \\ \beta(i) \stackrel{\text{def}}{=} d'\, ((i-1) \bmod d'), \end{cases} \end{cases} \tag{1}$$

with the matrix multiplication explicitized as a sum, which can be easily verified. Here, $[\mathbf{z}]_k$ stands for the k-th coordinate of the vector $\mathbf{z}$, and $\delta_\mathcal{P}$ is 1 if $\mathcal{P}$ is true; otherwise, it is 0. This is our basic definition, and reformulating the VTB operation using (1) will allow us to better understand its properties.

This operation is bi-linear in $\mathbf{x}$ and $\mathbf{y}$, and thus it is distributive with respect to addition and the scalar product.

Since $\mathbf{y}$ and $\mathbf{x}$ are random vectors $[\mathbf{z}]_i$, in (1) it is estimated up to a standard-deviation[34] of $O\left(1/d^{\frac{1}{4}}\right)$ which is an order of magnitude higher than for similarity estimation, which related standard-deviation was of $O\left(1/d\right)$. This also explains the relatively limited numerical performances of simulations with $d < 10^3$, as reported, for instance, in [33].

The $\sqrt{d'}$ renormalization factor allows $\mathbf{z}$ to have a unary order of magnitude[35]. However, the magnitude is not exactly one, but[36]:

$$\|\mathbf{B}_\mathbf{y}\, \mathbf{x}\| \simeq \begin{cases} \|\mathbf{y}\|\, \|\mathbf{x}\| & \text{if } \mathbf{y} \perp \mathbf{x} \\ \sqrt{2}\, \|\mathbf{y}\|\, \|\mathbf{x}\| & \text{if } \mathbf{y} \| \mathbf{x} \end{cases}$$

which is of importance in our context since the magnitude of the vector corresponds to the τ value.

At the algorithmic implementation level, the calculation of $\mathbf{z}$ is performed in[37] $O\left(d^{\frac{3}{2}}\right)$ operations, and the $\mathbf{y}_{k+\beta[i]}$ and $\mathbf{x}_{k+\alpha[i]}$ indexing can be tabulated in two fixed look-up tables $\beta[i]$ and $\alpha[i]$, avoiding any additional calculations. Furthermore, the fact that $\sqrt{d'}$ is an integer makes it possible to limit numerical approximations in order to improve

---

[33] All algebraic derivations reported here are straightforward and were verified using a piece of symbolic algebra code available at https://raw.githubusercontent.com/vthierry/onto2spa/main/figures/VTB-algebra.mpl.

[34] Each component $[\mathbf{z}]_i$ corresponds to the computation of a dot-product between two d' dimensional vectors, yielding a standard-deviation of $O\left(1/d'\right)$, renormalized by $\sqrt{d'} = d^{\frac{1}{4}}$, leading to the final order of magnitude. As for similarity estimation, chi-square distribution is approximated by a normal distribution of the same standard-deviation, which is known as a conservative choice.

[35] More precisely, two random normalized vectors of dimension d drawn from a random normal distribution of independent samples verify that $\mathbf{x} \cdot \mathbf{y} \sim \mathcal{N}(0, 1/d')$, as described in subsection 2.1. Then, applying a permutation on all indices on a random vector $\mathbf{x}$ yields another random vector, which is not correlated with any vector $\mathbf{y}$ if $\mathbf{x}$ is not. Thus, when computing the components $[\mathbf{z}]_i$ in (1) for two general random vectors $\mathbf{x}$ and $\mathbf{y}$, we compute the dot product of two random vectors of dimension d' renormalized by $\sqrt{d'}$, and thus this dot product comes from the distribution $\mathcal{N}(0, 1)$; this corresponds to drawing a random vector unary on average.

[36] We can easily derive:

$$\begin{aligned} \|\mathbf{B}_\mathbf{y}\, \mathbf{x}\|^2 &= d' \sum_{i=1}^{d} (\sum_{k=1}^{k=d'} [\mathbf{y}]_{k+\beta(i)}\, [\mathbf{x}]_{k+\alpha(i)})\, (\sum_{l=1}^{l=d'} [\mathbf{y}]_{l+\beta(i)}\, [\mathbf{x}]_{l+\alpha(i)}) \\ &= d' \sum_{ikl} [\mathbf{y}]_{k+\beta(i)}\, [\mathbf{x}]_{k+\alpha(i)}\, [\mathbf{y}]_{l+\beta(i)}\, [\mathbf{x}]_{l+\alpha(i)} \\ &= d' \sum_{ik} ([\mathbf{y}]_{k+\beta(i)}\, [\mathbf{x}]_{k+\alpha(i)})^2 + \text{ noise}\,, \end{aligned}$$

the 1st line being obtained by substitution from the definition, the 2nd line by expansion, and the 3rd line is obtained by considering that if $k \neq l$ we are multiplying four almost independent random distributions which product expectation cancels in average (i.e., formally: $E[Y_k X_k Y_l X_l] = E[Y_k] E[X_k] E[Y_l] E[X_l] \simeq 0$ by construction). If $\mathbf{y} \perp \mathbf{x}$ we are left with numerically approximating the mean of the square of normal distribution, i.e., its second momentum, equal to 1. If $\mathbf{y} \| \mathbf{x}$ we are left with numerically approximating the mean of the fourth power of a normal distribution, i.e., its fourth momentum, equal to 2 (it is known that if X is centered normal variable of unary standard deviation, then $E[X^{2n}] = (2-1)!!$ where !! denotes the double factorial, i.e., the product of all numbers down to 1 which have the same parity as the argument.)

[37] Each of the d components $[\mathbf{z}]_i$ requires a dot product of size $d' = \sqrt{d}$ that is not factorizable in the general case, since involving different elements of the vectors as readable on the matrix form.

numerical conditioning. This will be verified for all other explicit formulae later in this paper. We make explicit these formulae in detail not to re-implement these operations, which are already available in the NENGO simulator, but to study in detail their complexity and their precision, with the goal of proposing a macroscopic algorithmic ersatz of these operations.

This can be compared to the fastest binding operation, which is convolution implemented via the fast Fourier transform [33], and thus it has a complexity of $O\left(d\ \log(d)\right)$:

	d = 10	d = 100	d = 500	d = 1000	d = 10000
VTB	$10^{1.5}$	10^{3}	10^{4}	$10^{4.5}$	10^{6}
Convolution	$10^{1.4}$	$10^{2.6}$	$10^{3.5}$	$10^{3.8}$	10^{5}
Ratio $= \frac{\sqrt{d}}{\log(d)}$	$\simeq 1$	$\simeq 2$	$\simeq 3.5$	$\simeq 4.5$	$\simeq 10$

However, at the implementation level, we are going to observe that due to compiler and online processor optimization, the average computation time is an order of magnitude lower, because the VTB binding formula in eq. (1) is easy to optimize.

As stated in [17] and reviewed in [24], the key point is that this binding operation generates a new vector $\mathbf{z}$ that is almost orthogonal to $\mathbf{x}$ and $\mathbf{y}$:

$$(\mathbf{B}_\mathbf{y}\,\mathbf{x})\cdot\mathbf{x}\simeq 0,$$

and this operation is neither commutative,

$$(\mathbf{B}_\mathbf{x}\,\mathbf{y})\cdot(\mathbf{B}_\mathbf{y}\,\mathbf{x})\simeq 0,$$

nor associative[38], in the following sense:

$$(\mathbf{B}_{(\mathbf{B}_\mathbf{z}\,\mathbf{y})}\,\mathbf{x})\cdot((\mathbf{B}_\mathbf{z}\,\mathbf{B}_\mathbf{y})\,\mathbf{x})\simeq 0.$$

These properties ensure that we do not infer spurious derivations.

To take this a step further, in the real case, the random matrix is almost orthogonal, i.e.,

$$\mathbf{B}_\mathbf{y}^\top\,\mathbf{B}_\mathbf{y}\simeq\mathbf{I},$$

for the same reasons evoked above[39].

We thus define

$$\mathbf{B}_{\mathbf{y}^\sim}\stackrel{\text{def}}{=}\mathbf{B}_\mathbf{y}^\top\ \text{with}\ [\mathbf{y}^\sim]_i\stackrel{\text{def}}{=}[\mathbf{y}]_{\sigma(i)},$$

with

$$\sigma(i)\stackrel{\text{def}}{=}1+d'\left((i-1)\bmod d'\right)+(i-1)\ \text{div}\ d'.$$

In other words, $\mathbf{B}_\mathbf{y}^\top$ has the same structure as $\mathbf{B}_\mathbf{y}$, except that the vector coordinates are subject to a permutation $\sigma(i)$, which is idempotent ($\sigma(\sigma(i))=i$) and thus its own inverse, so that if $\mathbf{z}'\stackrel{\text{def}}{=}\mathbf{B}_{\mathbf{y}^\sim}\,\mathbf{x}$, we obtain

$$[\mathbf{z}']_i=\sqrt{d'}\ \textstyle\sum_{k=1}^{k=d'}[\mathbf{y}]_{\sigma(k+\beta(i))}\ [\mathbf{x}]_{(k+\alpha(i))}$$

(where $\beta(i)$ and $\alpha(i)$ are the indexing defined to calculate $\mathbf{B}_\mathbf{y}\,\mathbf{x}$ explicitly), and this makes it possible to define a left unbinding operation:

$$\mathbf{B}_{\mathbf{y}^\sim}\,(\mathbf{B}_\mathbf{y}\,\mathbf{x})=\mathbf{B}_\mathbf{y}^\top\,\mathbf{B}_\mathbf{y}\,\mathbf{x}=\mathbf{x}+\mathit{noise}\simeq\mathbf{x}.$$

---

[38] Of course, as a product of matrices, the combination of three bindings or two binding operations and a vector is associative, but the operator $\mathbf{B}$ itself is not, as made explicit in the formula.

[39] From (1), we derive

$$\begin{aligned}\left[\mathbf{B}_\mathbf{y}^\top\,\mathbf{B}_\mathbf{y}\right]_{ij}&=\textstyle\sum_{k=1}^{d'}[\mathbf{B}_\mathbf{y}]_{ki}\,[\mathbf{B}_\mathbf{y}]_{kj}\\&=d'\ \textstyle\sum_{k=1}^{d'}[\mathbf{y}]_{k+\beta(i)-\alpha(i)}\,[\mathbf{y}]_{k+\beta(j)-\alpha(j)}\\&=d'\ \textstyle\sum_{l=1}^{d'}[\mathbf{y}]_{l}\,[\mathbf{y}]_{k+(\beta(j)-\beta(i))-(\alpha(j)-\alpha(i))}\\&=d'\ \textstyle\sum_{l=1}^{d'}[\mathbf{y}]_{l}\,[\mathbf{y}]_{k+d'\ ((j-i)\ \text{div}\ d')-((j-i)\ \text{div}\ d')}.\end{aligned}$$

- $\left[\mathbf{B}_\mathbf{y}^\top\,\mathbf{B}_\mathbf{y}\right]_{ii}=\sum_{l=1}^{d'}[\mathbf{y}]_l^2=1+\mathit{noise}$; and
- $\left[\mathbf{B}_\mathbf{y}^\top\,\mathbf{B}_\mathbf{y}\right]_{i,i\neq j}=0+\mathit{noise}$, because it is easy to verify that $((j-i)\ \text{div}\ d')-((j-i)\ \text{div}\ d')\neq 0$ when $i\neq j$, so that the dot product of d' random components of $[\mathbf{y}]_l$ with d' other random components yields approximately normal noise.

From similar derivations, as detailed in footnote[36], also verified at the numerical level, we obtain the magnitude:

$$\|\mathbf{B}_{\mathbf{y}}^{\sim}\,\mathbf{B}_{\mathbf{y}}\,\mathbf{x}\| \simeq \begin{cases} \sqrt{2}\,\|\mathbf{y}\|^2\,\|\mathbf{x}\| & \text{if } \mathbf{y} \perp \mathbf{x} \\ \sqrt{5}\,\|\mathbf{y}\|^2\,\|\mathbf{x}\| & \text{if } \mathbf{y} \| \mathbf{x}, \end{cases}$$

which is to be considered to properly normalize unbinding operations at the mesoscopic level.

The right identity vector $\mathbf{i}$ such that $\mathbf{B}_{\mathbf{i}} = \mathbf{I}$ can be written explicitly as follows:

$$[\mathbf{i}]_i = \tfrac{1}{\sqrt{d'}}\,\delta_{i=\sigma(i)}.$$

In other words, we get i_B by "unfolding" the identity matrix I'_d line by line, writing a 1, then d times 0, then another 1, and so on.

Considering the mirroring matrix $\mathbf{B}_{\leftrightarrow}$, which is defined as

$$[\mathbf{B}_{\leftrightarrow}]_{ij} \stackrel{\text{def}}{=} \delta_{j=\sigma(i)}$$

(which is thus not block-diagonal in the way that a matrix of the form $\mathbf{B}_{\mathbf{y}}$ is), so that:

$$\mathbf{B}_{\leftrightarrow}\,\mathbf{x} = \mathbf{x}^{\sim},$$

we obtain

$$\mathbf{B}_{\leftrightarrow}\,\mathbf{B}_{\mathbf{y}}\,\mathbf{x} = \mathbf{B}_{\mathbf{x}}\,\mathbf{y}, \text{ while } \mathbf{B}_{\leftrightarrow}\,\mathbf{B}_{\leftrightarrow} = \mathbf{I} \text{ and } \mathbf{B}_{\leftrightarrow}^{\top} = \mathbf{B}_{\leftrightarrow},$$

which makes it possible to define a right unbinding operation:

$$(\mathbf{B}_{\mathbf{x}^{\sim}}\,\mathbf{B}_{\leftrightarrow})\,(\mathbf{B}_{\mathbf{y}}\,\mathbf{x}) = \mathbf{B}_{\mathbf{x}^{\sim}}\,\mathbf{B}_{\mathbf{x}}\,\mathbf{y} \simeq \mathbf{y},$$

and expand nested binding operations:

$$\mathbf{B}_{\mathbf{B}_{\mathbf{y}}}\,\mathbf{x} = \mathbf{B}_{\leftrightarrow}\,\mathbf{B}_{\mathbf{x}}\,\mathbf{B}_{\mathbf{y}}.$$

This could extend the actual binding algebra considering the dual operator $^{\sim}$.

Unfortunately, $\mathbf{B}_{\leftrightarrow}$ is not a binding matrix, i.e., it is not of the form $\mathbf{B}_{\mathbf{z}}$ for some vector $\mathbf{z}$, which is easily verified by the fact that some components that must be equal to 0 for a binding matrix are equal to 1 in $\mathbf{B}_{\leftrightarrow}$. Furthermore, the left or right multiplication of a binding matrix by this mirroring matrix does not yield a binding matrix, because of the same observation; components that must be equal to 0 for a binding matrix are equal to 1 in $\mathbf{B}_{\leftrightarrow}$.

Beyond [17], the authors of [24] introduced a vector composition operator $\oslash$ to make explicit the composition of two binding operations, namely,

$$\mathbf{B}_{\mathbf{v}} = \mathbf{B}_{\mathbf{y}}\,\mathbf{B}_{\mathbf{x}} \Leftrightarrow \mathbf{v} \stackrel{\text{def}}{=} \mathbf{y} \oslash \mathbf{x},$$

which can be explicitly written as follows[40]:

$$[\mathbf{v}]_i = \sqrt{d'}\,\textstyle\sum_{k=1}^{k=d'} [\mathbf{y}]_{(i-1)\,d'+k}\,[\mathbf{x}]_{1+d'\,(k-1)+(i-1) \bmod d'}.$$

At the algebraic level, the key point is that the product of two binding matrices is still a binding matrix. As a consequence, this composition operator is bi-linear, and thus it is distributive with respect to addition; it is not commutative, but it is associative and commutes with the inversion as follows:

$$(\mathbf{y} \oslash \mathbf{x})^{\sim} = \mathbf{x}^{\sim} \oslash \mathbf{y}^{\sim},$$

while $\mathbf{x}^{\sim} \oslash \mathbf{x} \simeq \mathbf{i}$; all these results can be easily derived by considering usual matrix properties. This allows us to combine two binding matrices without an explicit matrix product in $O\left(d^{\frac{3}{2}}\right)$ operations only. At the numeric level, since $\mathbf{v}$ is up to a $\sqrt{d'}$ factor, the dot product of segments of random vectors of dimension d', we obtain the same order of magnitude of noise level, as discussed previously. Altogether this can enrich the actual binding algebra to obtain more elegant formulas, in particular when combinations of binding operations are used.

---

[40] Since $\mathbf{B}_{\mathbf{y}}$ and $\mathbf{B}_{\mathbf{x}}$ are block-diagonal matrices, it is easy to verify that $\mathbf{B}_{\mathbf{v}}$ is a block-diagonal matrix with a $d' \times d'$ block $\mathbf{B}_{\mathbf{v}}{}' = \mathbf{B}_{\mathbf{y}}{}'\,\mathbf{B}_{\mathbf{x}}{}'$ using the notation from the beginning of this section, and we can explicitly write that

$$[\mathbf{B}_{\mathbf{v}}{}']_{ij} = \sqrt{d'}\,\sqrt{d'}\,\textstyle\sum_{k=1}^{d'} [\mathbf{y}]_{k+(i-1) \text{ div } d'}\,[\mathbf{x}]_{(k-1)\,d'+j},$$

from which we obtain the desired formula.

Using VTB algebra in the complex case

All of the developments described in this section generalize to complex numbers. Although it is not directly used here, such a generalization is of general interest because complex implementations of VSA frameworks have also been considered [33]. Furthermore, it is also of interest to see if our macroscopic implementation could be easily adapted to the complex case.

Stating that two resources are semantically equivalent if the unary vectors are aligned can be written in the complex case as follows[41]:

$$\mathbf{x} \simeq \mathbf{y} \Leftrightarrow < \mathbf{x}|\mathbf{y} > \simeq 1,$$

while the orientation is usually defined as

$$\widehat{\mathbf{x}\,\mathbf{y}} \stackrel{\text{def}}{=} \arccos(Re(< \mathbf{x}|\mathbf{y} >)),$$

as explained in the previous footnote.

Provided that the space dimension d is large enough, two randomly chosen different complex vectors $\mathbf{x}$ and $\mathbf{y}$[42] will also be approximately orthogonal in the sense that

$$\mathbf{x} \neq \mathbf{y} \Leftrightarrow < \mathbf{x}|\mathbf{y} > \simeq 0.$$

As a consequence, the VTB matrix is almost a unitary matrix, i.e.,

$$\mathbf{B_y}^* \, \mathbf{B_y} \simeq \mathbf{I},$$

considering the conjugate transpose.

All other algebraic operations are common to both real and complex linear algebra, and this is also the case for other VSA binding operators.

More than just a confirmation, these derivations allow us to observe that using a complex representation would be interesting if the conjugate of a vector could have a semantic interpretation. In that case, if, say, $\mathbf{x}$ and $\mathbf{y}^*$ are similar, then $< \mathbf{x}|\mathbf{y} > \simeq I$, as easily verified from the previous derivations.

---

[41] If we are in the real case $\mathbf{x}$ and $\mathbf{y} \in \mathcal{R}^d$, with $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$, then the equality is written as

$$\mathbf{x} = \mathbf{y} \Leftrightarrow \mathbf{x} \cdot \mathbf{y} = \sum_i x_i \, y_i = \cos\left(\widehat{\overrightarrow{\mathbf{x}}\;\overrightarrow{\mathbf{y}}}\right) = 1 \Leftrightarrow \widehat{\overrightarrow{\mathbf{x}}\;\overrightarrow{\mathbf{y}}} = 0 \pmod{2\,\Pi},$$

i.e., both unary vectors have the same direction; in other words, they are aligned. If we are in the complex case $\mathbf{x}$ and $\mathbf{y} \in \mathcal{C}^d$, let us consider the canonical embedding in $\mathcal{R}^{2\,d}$, i.e., we consider the real (Re) and imaginary (Im) parts as two real coordinates, denoting by $\overrightarrow{\mathbf{x}}$ the corresponding vector:

$$\mathbf{x} \stackrel{\text{def}}{=} (x_1, x_2, \cdots)^T \Leftrightarrow \overrightarrow{\mathbf{x}} \stackrel{\text{def}}{=} (Re(x_1), Im(x_1), Re(x_2), Im(x_2), \cdots)^T,$$

where z^* is the conjugate of a complex number z, while $< \mathbf{x}|\mathbf{y} >$ stands for the complex inner product:

$$\begin{aligned} < \mathbf{x}|\mathbf{y} > &\stackrel{\text{def}}{=} \textstyle\sum_i x_i \, y_i^* \\ &= \textstyle\sum_i (Re(x_i)\, Re(y_i) + Im(x_i)\, Im(y_i)) + I\,(Re(x_i)\, Im(y_i) - Im(x_i)\, Re(y_i)) \\ &= \overrightarrow{\mathbf{x}} \cdot \overrightarrow{\mathbf{y}} + I\,\overrightarrow{\mathbf{x}}^* \cdot \overrightarrow{\mathbf{y}}, \end{aligned}$$

so that $Re(< \mathbf{x}|\mathbf{y} >) = \overrightarrow{\mathbf{x}} \cdot \overrightarrow{\mathbf{y}}$ and $\|\mathbf{x}\| = \sqrt{< \mathbf{x}|\mathbf{x} >} = \|\overrightarrow{\mathbf{x}}\| - 2 = \sqrt{\overrightarrow{\mathbf{x}} \cdot \overrightarrow{\mathbf{x}}}$, and since vectors are unary,

$$< \mathbf{x}|\mathbf{y} >= 1 \Leftrightarrow \overrightarrow{\mathbf{x}} \cdot \overrightarrow{\mathbf{y}} = 1 \Leftrightarrow \overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{y}} \Leftrightarrow \mathbf{x} = \mathbf{y},$$

making explicit the obvious fact that unary real or complex vectors are equal if and only if their inner product equals one, while we consider the "angle" of two complex vectors as the angle of their $2\,d$ real embedding, i.e.,

$$\widehat{\mathbf{x}\,\mathbf{y}} \stackrel{\text{def}}{=} \arccos(Re(< \mathbf{x}|\mathbf{y} >)).$$

[42] Considering again the canonical embedding in $\mathcal{R}^{2\,d}$ and the fact that

$$< \mathbf{x}|\mathbf{y} >= \overrightarrow{\mathbf{x}} \cdot \overrightarrow{\mathbf{y}} + I\,\overrightarrow{\mathbf{x}^*} \cdot \overrightarrow{\mathbf{y}},$$

because $\overrightarrow{\mathbf{x}}$ and thus $\overrightarrow{\mathbf{x}}^*$ and $\overrightarrow{\mathbf{y}}$ are random vectors, their dot product almost vanishes; thus, the real and imaginary parts of $< \mathbf{x}|\mathbf{y} >$ also almost vanish.

C.1. Binding computation duration.

Let us finally observe the VTB binding computation time on a optimized processor in comparison of the $O\left(d^{\frac{3}{2}}\right)$ order of magnitude.

In Fig. 10, we observe the VTB binding mechanism computation time, which is expected to evolve with $O\left(d^{\frac{3}{2}}\right)$. In fact, at the implementation level, we observe that due to compiler and on-line processor optimization, the average computation time is an order of magnitude lower, because the VTB binding formula in eq. (1) is easy to optimize, at both the compilation and multi-core processor levels. Fitting the results we obtain, for one binding computation average time, something like:

$$T_{milli-seconds} = 0.048 + \frac{0.28\, d^{1.35}}{10000},$$

on a standard `Intel® Core™ i5-8265U CPU @ 1.60GHz × 8` processor, while the obtained result is not very stable around $O\left(d^{1.35}\right)$. This instability obviously depends on the processor multi-core actual state, when running the simulation.

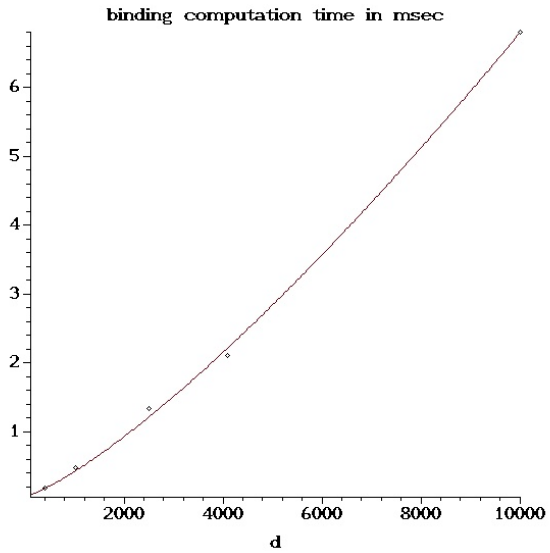

Fig. 10. Binding average computation time as a function of the space dimension. Although not very stable, a better fit is always obtained for a $O\left(d^{p}\right), 1 < p < \frac{3}{2}$ interpolation.

This result is not directly useful for our purpose, but is in favor of using the VTB algebra, including at the mesoscopic level, despite a less performing computation time.