Cognitive Neurodynamics

Where is Aristotle in our brain? On biologically plausible reasoning embedded in neuronal computation.

--Manuscript Draft--

Manuscript Number:	CODY-D-23-00028R2
Full Title:	Where is Aristotle in our brain? On biologically plausible reasoning embedded in neuronal computation.
Article Type:	Original Research
Keywords:	Ontology; Modal Logic; Semantic Pointer Architecture; Abstract Thought; Neuro-symbolism; Problem Solving
Corresponding Author:	Thierry Viéville Inria Bordeaux: Inria Centre de recherche Bordeaux Sud-Ouest Antibes, FRANCE
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Inria Bordeaux: Inria Centre de recherche Bordeaux Sud-Ouest
Corresponding Author's Secondary Institution:	
First Author:	Thierry Viéville
First Author Secondary Information:	
Order of Authors:	Thierry Viéville
	Chloé Mercier
Order of Authors Secondary Information:	
Funding Information:	Institut national de recherche en informatique et en automatique (INRIA) (AEx AIDE) — Not applicable
Abstract:	Human cognition involves tightly interleaved numerical but also symbolic (including logical) computations. Determining how the brain can implement such processing is an important issue, and we would like to address some aspects of this issue here by combining two approaches. On the one hand, ontology-oriented languages allow us to describe symbolic structured knowledge and perform logical inference using entailment rules. To what extent this could provide a rather natural representation of usual human reasoning is an open question that we are going to discuss here, considering a generalization to modal logic; we will also go beyond deductive reasoning, discussing whether this could also apply to inductive and abductive reasoning. On the other hand, 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. In the present work, we consider such an architecture with the vector symbolic architecture (VSA) formalism, allowing us to describe neuronal implementations at an algebraic level. This development illustrates how the former cognitive mechanisms can naturally emerge from distributed calculus, yielding neuro-symbolic computations. Our aim is to show that it can be implemented, considering a VSA approach, in biological neuronal computations. Such neuro-symbolic deductive mechanisms are especially useful in complex problem solving.
Response to Reviewers:	Editor: > We have received the reports from our advisors on your manuscript, "Where is Aristotle in our brain? On biologically plausible reasoning embedded in neuronal computation.", which you submitted to Cognitive Neurodynamics.

Based on the advice received, I feel that your manuscript could be reconsidered for publication should you be prepared to incorporate major revisions.

Thanks a lot for offering a new chance to this work, we are especially thankful for the reviewers' feedback and have reworked the paper accordingly.

We really consider reviewer#2 feedback, as very precious, deeply helping us and it is absolutely clear that if this paper will be of some scientific interest, it will be thanks to her or his patient corrections and re-corrections. No will to "break" his or her anonymous status but if by no means it could be mentioned how much this paper has to acknowledge her/his benevolent work, please tell us.

Reviewer #1:

> The authors have incorporated the necessary corrections in the manuscript.

Thanks a lot for this feedback.

Reviewer #2:

> I thank the authors for responding to my previous comments and revising their manuscript. However, I am unfortunately still unable to follow the paper, let alone recognize its contributions. I have done research on VSA and have a decent understanding of logic, but still I am unable to understand many paragraphs in the paper. I might be to blame though, and I don't want a potential lack of proper background on my side to lead to a negative evaluation of the paper. So I refrain from giving any recommendations and ask the editorial board to seek other reviewers who may be better equipped to review this paper.

On our side, we are deeply thankful that you took the time to review this paper and really appreciate that you gave this paper a chance.

> The first equation on page 5 needs magnitudes to be exactly 1, not just O(1).

Thank you for this remark, it is true that depending on the random draw it could be exactly one if the vector is renormalized, but only approximately 1 otherwise. We have removed the ambiguous O(1) shortcut and made the point explicit.

> The second equation on page 5: is 1/sqrt(d) standard deviation or variance? Because multiplying this by sqrt(d) is said to give standard normal distribution below, which is not consistent with the conventional notation N(mu, sigma^2).

Sorry about that, we are now consistent in the whole paper with respect to the conventional notation.

> After the second equation on page 5: What do these percent biases exactly mean? Why do we have a bias in the first place, and how are these calculated (for one of them, e.g.)?

We agree, and have now not only shared the source but described the simulation. Wee have also replaced the term "bias" with "relative precision".

> First sentence of Section 2.2: Why? X has a similarity of 1 with X + Y when Y is independent from X, but X and X+Y are not semantically equivalent.

We fully agree with this comment. More than that, we have taken the point and proposed a footnote that make explicit, at the algebraic level, what is "semantic similarity" (as quoted by other authors). Yes, semantically similar does not mean equivalent, but contains an element that is semantically similar. It means that our sentence is unclear and we have rewritten it.

> First equation of page 8: Why is this a good representation of belief? Why weight the symbol by someone's belief? Belief in what actually?

Following this remark, we have restructured the whole section:
- make clearer this notion of belief
- better (hopefully) providing arguments for our design choice and quoting previous works.

> The paragraph after the first equation of page 8: The authors are drawing an association between two things that happen to range from -1 to 1: cosine similarity, and belief in possibility theory. But why is this a good association? If two vectors have a cosine similarity of 0, why is the answer to the question "are two vectors similar?" unknown? In my opinion the answer is no!

We understand your answer, we should have better made explicit the fact we are in an “open world” in which what is not true isi not necessarily false, but unknown. This is now better made explicit.

> End of page 8: So concept = symbol = hypervector?

Concepts are represented by a symbol, indeed. And all symbols are encoded in hypervector. But not all symbols are concepts, except if an individual is considered as a concept “singleton”. This is now rewritten to precise this point.

> Second paragraph of page 9: Why only these 4 predicates? "John likes apples", "John speaks Spanish", ... Aren't these knowledge?

Indeed, this is not obvious and the (McClelland & Rogers, 2003) design proposal requires a development not easy to report in a few words. We however, have taken benefit of this reviewer's remark to exemplify the idea and add a short complementary explanation.
Quoting McClelland & Rogers: « In this approach, the four propositional relations from Quillian’s hierarchy are viewed as distinct contexts. The is a relation corresponds to a naming or explicit categorization context, in which the object’s name or category label is of relevance, as these might be indicated verbally (e.g., by a sentence such as “This is a bird” or “This is a canary”). The can relation corresponds to a context in which the behaviors of the object might be observed; the is relation corresponds to a context in which its appearance properties are highlighted; and the has relation corresponds to a context in which its parts are highlighted. »

> Section 3.3.2: This section is supposed to be all about "relational maps", but what is a relational map ultimately?

Sorry, again. An introduction was required, now done adding a precise definition, making the link with the previous development more explicit.

> I don't understand the significance or the point of defining t_{p_j}, nor the equation B_{p_j^~} s = t_{p_j}. What is s actually? And why is this true?

This is used at the implementation level. And the derivation is proposed in the related footnote. We have now rewritten this paragraph and corrected a typo in the notations, so sorry for that.

> On page 13, there is a sentence that says "A key point is that ..." until "from t_{spo}". Why? Aren't the equations you give a few lines earlier doing exactly that?

Oh, this means that our notations are confusing between t_{pso} and t_{spo}: we have now more clearly mentioned this double similar notation.

> Section 4.1: Neither OWL nor RDFS have been properly defined before.

Indeed, it is now added.

Where is Aristotle in our brain? On biologically plausible reasoning embedded in neuronal computation

Chloé Mercier[1] and Thierry Viéville[1,2†]

[1]Mnemosyne Team, Inria Bordeaux, U. Bordeaux, LaBRI and IMN.
[2]LINE Laboratory, U. Côte d'Azur.

Contributing authors: chloe.mercier@inria.fr; thierry.vieville@inria.fr;
†Supported by Inria,AExAIDE.

Abstract

Human cognition involves tightly interleaved numerical but also symbolic (including logical) computations. Determining how the brain can implement such processing is an important issue, and we would like to address some aspects of this issue here by combining two approaches. On the one hand, ontology-oriented languages allow us to describe symbolic structured knowledge and perform logical inference using entailment rules. To what extent this could provide a rather natural representation of usual human reasoning is an open question that we are going to discuss here, considering a generalization to modal logic; we will also go beyond deductive reasoning, discussing whether this could also apply to inductive and abductive reasoning. On the other hand, 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. In the present work, we consider such an architecture with the vector symbolic architecture (VSA) formalism, allowing us to describe neuronal implementations at an algebraic level. This development illustrates how the former cognitive mechanisms can naturally emerge from distributed calculus, yielding neuro-symbolic computations. Our aim is to show that it can be implemented, considering a VSA approach, in biological neuronal computations. Such neuro-symbolic deductive mechanisms are especially useful in complex problem-solving.

Keywords: Ontology, Modal Logic, Semantic Pointer Architecture, Abstract Thought, Neuro-symbolism, Problem Solving.

1 Introduction

1.1 From sensorimotor processing to logical reasoning

It is generally admitted (as reviewed, e.g., in [1]) that human logical reasoning emerges progressively from the sensorimotor association, with the formation of stable concepts, even at a symbolic level, before it is possible to manipulate them at a concrete level, by performing inductive reasoning, and perform more formal deductive reasoning. Furthermore, as pointed out in [2], such a mechanism includes a cultural bias, since not all cultures feel the need to develop formal logical operation competencies, and obviously, most people do not use such formal operations in all aspects of their lives. However, as discussed in, e.g., [3], deductive reasoning, especially for goal-driven behavior, is deeply interleaved with heuristic deduction, which involves analogy and metaphor. Furthermore, as thoroughly studied by, e.g., [4], the experience of conscious or subconscious emotion has a powerful influence on rational decisions, including the choice of alternatives in deductive reasoning. Does this mean that the human brain does *not* need to develop deductive reasoning, except for singular cultural needs (e.g., for scholarship or to practice formal science)? Our understanding is that the situation is not binary. We need to make deductions to solve problems in everyday life, while the elements we briefly reviewed here demonstrate that such deductions are not Boolean (either true or false) but related to a given context, weighted by a certain level of belief, and biased by motivational elements in a wide sense.

In this study, we attempt to reconcile deductive reasoning with such cognitive mechanisms of inference to reach a biologically plausible neuronal implementation and show to what extent this could be extended to approximate deductive reasoning, in addition to inductive and abductive reasoning mechanisms[1].

Taking this discussion a step further, how does the brain give meaning to the symbols considered here? We will not address this issue here but would like to clarify some points. First of all, in neuro-symbolic studies, as reviewed in [5],

[1]Here we make the distinction between
- *deductive reasoning*, which is the process of determining the formal logical consequence of some assumptions considered as true or approximately true;
- *inductive reasoning*, which is the process of inferring some general principle from a set of knowledge and plausible induction rules; and
- *abductive reasoning*, which is the process of inferring an explanation of some assertions, i.e., hypothesizing the precondition of a consequence.

grounding is understood as the process of embedding symbolic computations onto real-valued features [6] because it provides a semantic interpretation or model (in the sense of a model of a set of logical assertions) of the symbolic system. This means that it is no longer an abstract set of assessments (potentially without any concrete implementation) but something that corresponds to a real formal object. Our approach thus does not solve the grounding problem, but in some sense, it does solve what we can call the "anchoring" problem of relating symbols to the neural substrate. This will be further discussed in the last section after the present approach is described in detail.

1.2 Representing neuronal activity at a symbolic level

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., [7] for an introduction). VSAs have been proven to be helpful in modeling high-level cognition and accounting for multiple biological features [8, 9]. More specifically, the semantic pointer architecture (SPA) [9] 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 in that it provides a way to translate the former into the latter. Consequently, complex knowledge representations in the form of compositional structures that are traditionally restricted to symbolic approaches can now be distilled in numerical and even neural[2] systems [10].

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 of the whole assembly. Let us consider a spiking neuron network and quantify its activity using some statistics, e.g., the neuron rates or higher-order statistics (see, e.g., [11] for a discussion). As developed in [12], this includes timing codes and population codes (i.e., relative timing codes between neurons), and the authors of [12] show how, with their developed neural engineering framework (NEF), we can collect this high-dimensional set of bounded quantitative values, which can be normalized, as a unitary stochastic vector in a high-dimensional space (with a few thousand dimensions for a biological neuronal map and often a few hundred dimensions at the simulation level), defining a SPA. This includes a time representation in spiking neuron systems, not just a rate representation. The key point is that compared to other representations, e.g., based on synchrony within the neural assembly, the NEF alternative is much more scalable.

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

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

and define them at this abstract algebraic level. Mainly, following [13], we will consider the auto-association mechanism, as developed in [14], and functional transformations, as detailed in [12]; their development is based, in a nutshell, on parameterized kernel-based approaches.

1.3 What is this paper about?

We first revisit how to encode symbols within the VSA approach based on the framework introduced in [9], analyzing in more detail than previous works the numerical approximation statistic, because this is needed to model the VSA mechanism at a macroscopic level. We also describe how to generalize symbol encoding considering a related degree of belief, beyond binary information, and following [13], we explain the semantic interest of such a generalization.

We then consider knowledge hierarchical structure encoding in the sense of, e.g., [15], as a complement to associative and sequential memorization, and we discuss how to implement such a memory structure using the VSA. To this end, we have to review VSA data structures and demonstrate that they are linked to cognitive memory classification according to [15]. To better understand their computational properties, we also illustrate how such existing VSA data structures compare to programming language containers.

This new data structure is the basic tool that is used to then study to what extent symbolic inference could be implemented by specific connectivity feedback, and, as a second contribution, we show that the vanilla fixed-point algorithm used for a forward deduction on a decidable set of entailment rules can be adapted to this biologically plausible framework, making it possible to perform deductive reasoning and also, to some extent, inductive and abductive reasoning.

We finally illustrate the proposed mechanism using both a simulation at the mesoscopic level, utilizing the well-established Nengo simulator, and also a simulation at the macroscopic scale. We show, as the third contribution of this paper, that such computations may be, up to a certain point, approximated without explicitly performing computations at the vector component level; instead, an algorithmic ersatz can be used.

2 Symbolic information encoding

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. A typical dimension of 256 has been considered in related studies, such as [13]. Here, our macroscopic implementation uses the same dimension. In a cortical or brain map, the order of magnitude is higher since the vector corresponds to the neuronal map activity (thus it is closer to $10^{5 \cdots 6}$) and the number of encoded symbols depends on which map is considered.

The vector components are drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$, i.e., this distribution has zero mean and a standard deviation $\sigma \overset{\text{dec}}{=} 1/\sqrt{d}$, in order to have an average magnitude of 1.

A similarity measure is now introduced in order to semantically compare two vectors. This precisely means that both vectors[3] in and carries similar information. Classically, the cosine similarity (i.e., normalized dot product, denoted $\cdot$) is used to compute the semantic similarity between two unit vectors:

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

where $\mathbf{x}^\top$ denotes the transpose of $\mathbf{x}$. This measure obviously 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,

$$\mathbf{x} \cdot \mathbf{y} \sim \delta_{\mathbf{x}=\mathbf{y}} + \mathcal{N}(0, 1/d),$$

i.e., it is almost 1 if equal and 0 otherwise, plus centered normal noise [16]. At the numerical level[4], 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. 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[5], with d samples and a known standard deviation with an order of magnitude $O(1/d)$, is the following:

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

It follows an almost distribution, which can be easily verified numerically, as shown in Fig. 1 (left column). For two vectors that are not independent but have an angular dependency, we can numerically observe, in Fig. 1 (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.

This makes it possible, on the one hand, to consider, for instance, a ± 2 threshold for the standard deviation, along with considering this z-score to have

[3]Let us consider two vectors $v_1 = \mathbf{u} + \mathbf{w}_1$ and $v_2 = \mathbf{u} + \mathbf{w}_2$ thus carrying the same semantic information encoded in $\mathbf{u}$ plus, say other independent information $\mathbf{w}_1$ and $\mathbf{w}_2$ independent from $\mathbf{u}$ and form each other. Then $v_1^T v_2 = u^T u = 1$, since other vectors orthogonal. This is the precise meaning of semantic similarity.

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

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

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

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

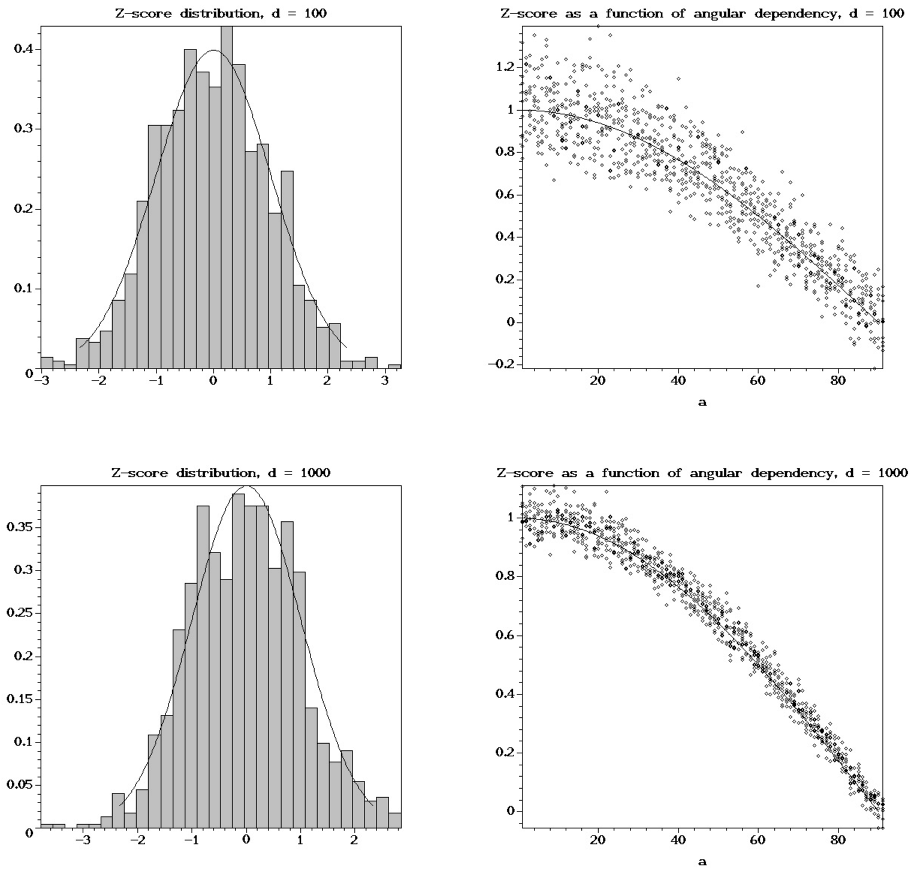


Fig. 1 Numerical observations of the similarity defined by the dot product of two random vectors for $d = 100$ in the upper row and $d = 1000$ in the lower row. The left column shows the histogram of the z-score ($\sqrt{d}\,(\mathbf{x}\cdot\mathbf{y})$) for two normal vectors, in comparison with a normal 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 $a \overset{\text{dec}}{=} \widehat{\mathbf{x},\mathbf{y}} = \arccos(\mathbf{x}\cdot\mathbf{y})$, making it possible to visualize the dispersion with respect to the expected cosine profile.

a confidence interval better than 99%, and to relate the similarity estimation to an angular dependence between two vectors, as detailed in Fig. 1. To the best of our knowledge, this obvious implementation has not yet been made explicit, and it is used in subsection 5.1, as detailed in Appendix D, allowing us to propose to simulate the different operations defined later in this paper at a macroscopic scale.

2.2 Modality encoding

2.2.1 The notion of belief

Most VSA approaches consider that two vectors $\mathbf{x}$ and $\mathbf{y}$ contain (not exclusively) equivalent information that is semantically equivalent, when the similarity τ equals 1, but 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. 2. 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 absolutely not necessary value.

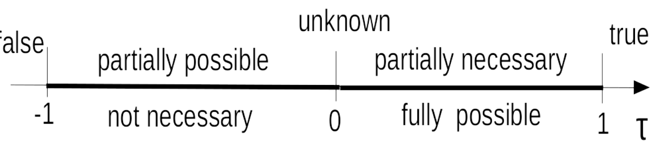


Fig. 2 Representation of the partial truth $\tau \in [-1, 1]$ with regard to necessity and possibility, as defined in the possibility theory. The interpretation is that what is "not so false" is partially possible but not necessary and what is "partially true" is entirely possible but partially necessary. Such a formulation corresponds qualitatively to the human appreciation of the degree of belief in a fact.

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, the human "level of truth" is more subtle and related to possibility and necessity, as formalized in the possibility theory discussed in [17] and [18]. This theory stems from modal logic, i.e., something being true in a "given context" [19], which is also considered representative of what is modeled in educational science and philosophy [20]; namely, it corresponds to common-sense reasoning in the sense of Piaget [21], which involves taking exceptions into account, i.e., considering non-monotonic reasoning. In other words, 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. 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 [22]. 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 is not only

[6]This deterministic representation of partial knowledge can be generalized in order 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

epistemic or doxastic but also deontic and so on, i.e., it has several semantic interpretations, depending on the concept feature. 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 [25], where this deterministic representation of partial knowledge is generalized in order 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 symbols. A symbol representing a piece of information with a partial degree of belief $\tau \in [-1, 1]$ could be defined as:

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

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 used in different cognitive data structures.

3 Knowledge structure encoding

3.1 Knowledge representation

From early artificial intelligence knowledge representations such as previously avowed frameworks to modern web semantic data structures, the basic idea of symbolic representation is to consider symbols that represent objects and express knowledge through relationships, i.e., triple statements of the form `($subject, $predicate, $object)`, as schematized in Fig. 3.

Following, e.g., [26], we start by introducing the notion of a concept, represented by a symbol encoded in a hypervector. Several symbols may be elements

probability distributions, i.e., a consonant plausibility measure in the Dempster–Shafer theory of evidence [23]. In [24, 25], 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.

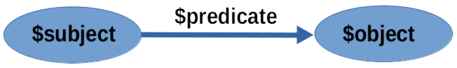


Fig. 3 Atomic representation of knowledge: To express some knowledge regarding a symbol, the subject, we define a feature with a predicate that has an object as an attribute (i.e., a quantitative data value or a qualitative symbol).

of such a concept. A concept corresponding to a unique symbol, i.e., a singleton, corresponds to an individual. Concepts, in, [26] are specified with the simple common idea that a concept can be defined by "feature dimensions," i.e., attributes with some typed value. The object can be either qualitative or quantitative "data" or another object that describes relations between objects. This is also the basic syntax of ontology languages. In the brain, such feature dimensions are usually anchored in sensorimotor feature spaces [27], in coherence with the present representation. Furthermore, given a concept, as developed in [26], this choice of representation induces the notion of prototypes, which makes it possible to represent the state space region corresponding to the concept.

Taking this a step further, the definition of a concept is completed by relations between concepts, i.e., predicates. Predicates can be generic, in the sense of, e.g., [28], defining a hierarchical taxonomy using the "`is-a`" predicate, as in almost any such language, but they also have capability qualities ("`can`"), extrinsic qualities ("`has`"), and intrinsic qualities ("`is`"); thus, there are four general predicates. They can also be unconstrained, as in the RDFS framework (see, e.g., [29] for an introduction), describing any property, and in that case, properties also form a taxonomy[7]. This is further illustrated in subsection 3.2. Conversely, as a limit case, we could consider relationships between subjects and objects only, without taking into account the nature of a relationship (predicate) or its direction.

An important point is that features can be hierarchical because the value itself may have some features: For instance, a quantitative physical value is not just a "number" stating the current or default value. It may also be specified by a unit and a precision value or some bounds. In terms of the data structure, this forms a tree, and the whole data structure is a set of trees, i.e., a forest, and thus it is a graph.

We are going to specify how we can represent such symbolic information at a biologically plausible level in the present study using VSAs.

3.2 Hierarchical schematic organization of information

To make things more concrete, we aim at manipulating the symbolic representation of knowledge of the form shown in Fig. 4.

[7]For instance, stating that `Amid is-a-descendant-of Yang-Li` implies that `Amid is-a-relative-of Yang-Li`, the former property is a sub-property of the latter. This example also illustrates that such properties have meta-properties, such as being transitive or symmetric.

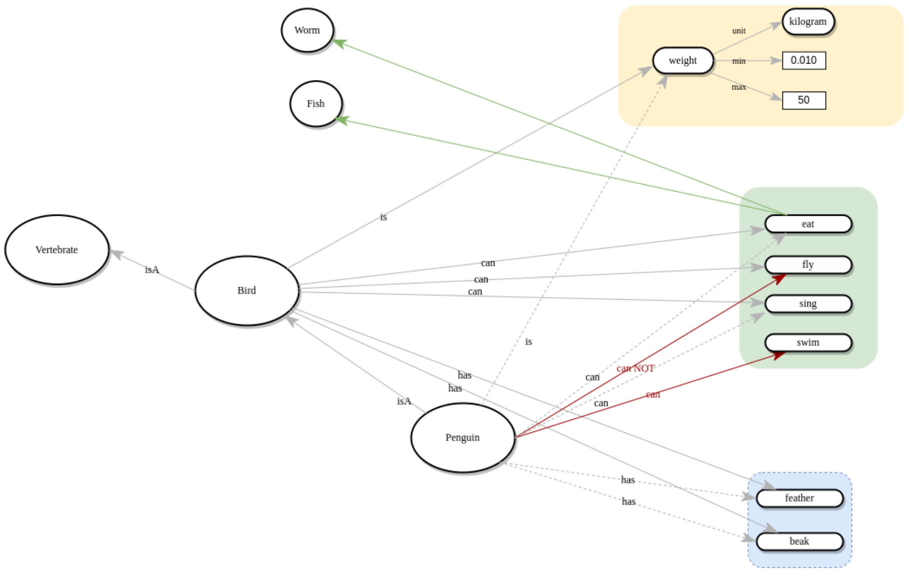


Fig. 4 In our context, we represent concepts as a hierarchical data structure. Concepts are anchored in an input/output, i.e., stimulus/response, framework, which might consist of sensorimotor feature spaces (colored regions) corresponding, for example, to different sensor modalities. Inherited features (e.g., the penguin "is-a" bird and thus inherits the features of a bird) are shown with dotted lines, while red lines represent overwritten values (e.g., a penguin can also swim but cannot fly). Green arrows point toward concepts that are themselves attributes of other concept features, accounting for inter-concept relationships. Values are completed by meta-information that is not explicitly manipulated by the agent but is used for process specification or interpretation (e.g., the weight unit and bounds).

In other words, we follow [26], with the simple idea that an individual resource can be defined by "feature dimensions," i.e., attributes with some typed value. For instance, a bird could be the following[8]:

```
bird: {
  is_a: vertebrate
  can: { sing fly eat: { worm fish } }
  has: { feather beak }
  is: { weight : { min: 0.010 max: 50 unit: kilogram } }
},
```

with some exceptions like penguins:

```
penguin: {
  is_a: bird
  can: { fly: false walk }
}.
```

Equivalently, this could be decomposed[9] in terms of triples such as `bird is_a vertebrate` and so on.

[8]The syntax used is a weak form of the JSON syntax: https://line.gitlabpages.inria.fr/aide-group/wjson.

[9]Conceptually, considering a distributed version of this hierarchical specification by enumerating each node and the directed edge is rather obvious; at the concrete implementation level, some

Here, we choose the general approach of semantic knowledge representation using a hierarchical taxonomy (`is-a`) with capability features (`can`), including those related to other resources, extrinsic features (`has`), and intrinsic features (`is`) [28][10]. A more granular partition could be envisioned, but this illustrative example is sufficient to allow us to detail the main characteristics of our representation. Some features are properties, and others are relations. A property can be qualitative, e.g., the `is-covered-by` property takes a value in an enumeration (e.g., {`sing`, `fly`}), or quantitative (e.g., the `weight`). The features can be hierarchical, either because the value is an enumeration (e.g., `can`) or because the value has some features (e.g., `weight`). @vthierry:[For instance, the statement "Zheng-You speaks Sundanese" means "Zheng-You can [has the capability to] speak Sundanese", while the statement "Zheng-You is speaking Sundanese" describes an ongoing action rather than a concept attribute, while the statement "Leila likes kiwis" means "Leila has kiwis in her favorite fruits set". These four verbs seem sufficient, as developed by the authors to describe concept properties in the wide sense. Choosing such normalized representation allows us to better represent knowledge in a canonical form.]

Such a data structure defines a "concept" in the sense of [26] (e.g., "a bird"), which is both a convex region of the state space (e.g., the region of all birds) and a prototype: Each feature has a default value, and this also defines a prototype (e.g., a typical, i.e., prototypical, bird). It corresponds to the third cognitive memory architecture, as proposed by [15], and we are now going to discuss how to implement it in a VSA framework.

Let us now review how such a cognitive symbolic data structure can be implemented in a biologically plausible way using the proposed VSA framework.

3.3 Biologically plausible implementation

To proceed, we thus have to consider a set of items of knowledge of the form shown in Fig. 3 bounded to a triple of vectors $\{(\mathbf{s}_1, \mathbf{p}_1, \mathbf{o}_1.), \cdots (\mathbf{s}_N, \mathbf{p}_N, \mathbf{o}_N.)\}$. This corresponds to relations, which are the basis of semantic information.

3.3.1 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 [15], and to different programming containers. The key point of the present work is, on the one hand, to

details have to be carefully taken into account and the so-called turtoise specification takes care of proposing a well-defined one-to-one correspondence (with an open-source implementation). It is not necessary to describe this in detail here.

[10]Quoting the authors *In this approach, the four propositional relations from Quillian's hierarchy are viewed as distinct contexts. The "is a" relation corresponds to a naming or explicit categorization context, in which the object's name or category label is of relevance, as these might be indicated verbally (e.g., by a sentence such as "This is a bird" or "This is a canary"). The can relation corresponds to a context in which the behaviors of the object might be observed; the is relation corresponds to a context in which its appearance properties are highlighted; and the has relation corresponds to a context in which its parts are highlighted.*

verify that these VSA mechanisms generalize to modal symbol encoding, which is technically obvious (see Appendix B) but worthwhile to mention, and, on the other hand, to show that they also very easily generalize to the so-called "relational map," as developed in the next subsection. Further details on a scalable biologically plausible knowledge representation can be found in [10].

Container	VSA mechanism	Cognitive usage	Main available operations
Set	Bundling or superposition		+ Element insertion/modification + Check membership - No enumeration
Map or dictionary	Binding superposition	Associative memory	+ Element insertion/modification + Value·s retrieval from key + Key·s retrieval from value + Exact symbol recovery 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

Table 1 Biologically plausible data containers; see Appendix B for details.

3.3.2 Relational maps

How should we such information in a biological data structure? As developed previously all knowledge is decomposed in the form of triples, as shown in Fig. 3, i.e., in the form of relations between entities. We now discuss how to implement such a distributed representation, a kind of "triple store" using superposition, i.e., bundling, and a few binding operations, i.e., a *relational map*. The most natural choice might be to consider the triple set $\mathcal{T}$ grounded to a vector $\mathbf{t}$:

$$\mathbf{t_{pso}} \overset{\text{def}}{=} \sum_i \mathbf{B}_{\mathbf{p}_i}\, \mathbf{B}_{\mathbf{s}_i}\, \mathbf{o}_i.$$

Here,
- $\mathbf{o}_i$ represents vectors encoding symbols;
- quantities of the form $\mathbf{B}_\mathbf{y}$ are binding matrices that make it possible to create a $\mathbf{B}_\mathbf{y}\,\mathbf{x}$ key-value pair (the key is $\mathbf{y}$ and the value is $\mathbf{x}$), as defined in equation (A1) of Appendix A;

- combining these quantities to create a vector $\mathbf{B}_{\mathbf{p}_i}\,\mathbf{B}_{\mathbf{s}_i}\,\mathbf{o}_i$ makes it possible to encode a triple as a new random vector, i.e., as a neuronal ensemble activity; and
- finally, the sum makes it possible to superpose a different triple. A reader not familiar with this VSA formalism will find in Appendix B a didactic introduction, while the choice of the binding operator from among several available binding operators [16] is discussed in Appendix A. For this section to be self-contained, one just has to consider that binding makes it possible to create a key-value symbol pair, while the unbinding operation, which is written as $\mathbf{B}_{\mathbf{y}^\sim}$, makes it possible to retrieve the value from the key, as discussed below.

This design corresponds to a nested associative map: Each property $\mathbf{p}_j$ is defined through a mapping between subjects and objects[11]:

$$\mathbf{t}_{\mathbf{p}_j} \overset{\text{def}}{=} \sum_{i,\mathbf{p}_j=\mathbf{p}_i} \mathbf{B}_{\mathbf{s}_i}\,\mathbf{o}_i = B_{\mathbf{p}_j^\sim}\,\mathbf{t}_{\mathbf{pso}}.$$

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:

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

and this obviously generalizes to a triple multiplied by a modal τ value.

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

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

using the notation of Appendix A. We can also easily define

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

A dual construction, $\mathbf{t}_{\mathbf{spo}} \overset{\text{def}}{=} \sum_i \mathbf{B}_{\mathbf{s}_i}\,\mathbf{B}_{\mathbf{p}_i}\,\mathbf{o}_i$, with similar decoding formulae makes it possible to further access the properties of a given subject $\mathbf{t}_{\mathbf{s}_j} \overset{\text{def}}{=} \sum_{i,\mathbf{s}_j=\mathbf{s}_i} \mathbf{B}_{\mathbf{p}_i}\,\mathbf{o}_i$ or the properties of a given subject-object couple $\mathbf{t}_{\mathbf{s}_j,\mathbf{o}_j} \overset{\text{def}}{=} \sum_{i,\mathbf{s}_j=\mathbf{s}_i,\mathbf{o}_j=\mathbf{o}_i} \mathbf{p}_i$ using similar formulae. We thu shave now two relation maps $\mathbf{t}_{\mathbf{pso}}$ and $\mathbf{t}_{\mathbf{spo}}$. A key point is that, to the best of our knowledge, 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 $\mathbf{t}_{\mathbf{spo}}$. 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.

Moreover, 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.

[11] Obviously,

$$B_{\mathbf{p}_j^\sim}\,\mathbf{t}_{\mathbf{pso}} = \sum_{i,\mathbf{p}_j\neq\mathbf{p}_i} B_{\mathbf{p}_j^\sim}\,\mathbf{t}_{\mathbf{p}_i} + \sum_{i,\mathbf{p}_j=\mathbf{p}_i} B_{\mathbf{p}_j^\sim}\,\mathbf{B}_{\mathbf{p}_i}\,\mathbf{B}_{\mathbf{s}_i}\,\mathbf{o}_i \simeq \mathbf{t}_{\mathbf{p}_j}.$$

The first term vanishes because $\mathbf{t}_{\mathbf{p}_i}, \mathbf{p}_j \neq \mathbf{p}_i$, is generically orthogonal to $\mathbf{p}_j^\sim$, while the second reduces to $\mathbf{t}_{\mathbf{p}_j}$ thanks to the approximate inverse property.

To take this a step further, we can also consider an additional symbol "something," and each time a triplet $(\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_{i\cdot})$ is added, we can also add $(\sigma, \mathbf{p}_I, \mathbf{o}_{i\cdot})$, $(\mathbf{s}_i, \sigma, \mathbf{o}_{i\cdot})$, 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, as previously discussed.

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., [15].

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 $\mathbf{t_{spo}}$ construction and a hierarchical edge set for $\mathbf{t_{pso}}$ construction.

At this stage, we have reviewed and developed the VSA elements needed for the next step, in particular the notion of a relational map, which is a combination of associative maps enumerated by an indexed list that builds on the bundling structure reviewed at the beginning of this section. We are now going to explain how symbolic computation can be performed by making such memory structures interact.

4 Knowledge transformation encoding

Let us now consider how to define operators that make it possible to enrich the memorized information using biologically plausible transformations.

4.1 Considering symbolic operations

We are going to consider symbolic operations as entailment rules of deductive, inductive, or abductive inference. It is worthwhile to note that both induction, as, for instance, in [30], and abduction could be formalized by inference rules (see, e.g., [31] for a computational example and [32] for a contribution regarding temporal reasoning). Our position is that reasoning in the brain is mainly related to the construction of mental models, as discussed in, e.g., [33], considering common-sense reasoning. Such a framework better corresponds to human reasoning than pure logic reasoning, including modal logic reasoning [34]. In any case, a mechanism to construct, enrich, and modify such a mental model must be described.

Here, we focus on rule-based reasoning. Regarding an ontology such as OWL[12] semantic web knowledge representation language family, the RDF Schema (RDFS)[13] layer (see Appendix C for a detailed presentation) is purely rule-based, and several property inferences such as the inverse, symmetry, transitivity, and rules of equating objects are rule-based. This corresponds to the so-called OWL 2 RL profile. It allows scalable reasoning without sacrificing

[12]https://en.wikipedia.org/wiki/Web_Ontology_Language
[13]https://en.wikipedia.org/wiki/RDF_Schema

too much expressive power and can be implemented using rule-based reasoning mechanisms[14]. By carefully considering the related entailment rules of this specification, we have observed that they require either one, two, or at most three premises, which is an important aspect in our context. Beyond this, other rule-based mechanisms can complete the description of rule-based reasoning, such as by adding function-free Horn rules (e.g., using the SWRL[15] language with suitable conditions [35]). This is what we target here.

Beyond deductive reasoning, following [30], for instance, or using inductive logic programming[16], inductive reasoning is easily formulated using the rule-based mechanism provided; we can add counting operations in order to perform enumerative induction. This means that we will have rules that are going to scan the whole relational map, but we are going to observe that this is also required for other rules. This being stated, we will not further develop this aspect, and it is only mentioned as an interesting perspective of the present work.

Taking this a step further, rule-based abduction has been formalized, following, e.g., [31], as in fact a set of possible causes of a given observation that is deduced (to make short a more sophisticated story). In this rather restricted but still powerful context, the rule-based mechanism does not "invent" the cause but does infer parameterized predefined causes, performing what could be called model-based abduction.

To summarize, our design choice is to propose a rather generic mechanism, specific enough to produce a robust implementation and make it possible to specify what has been reviewed here.

4.2 A generic mechanism

In order to proceed, let us write

$$(\$\mathbf{s}_0, \tau_0\, \$\mathbf{p}_0, \$\mathbf{o}_0.),$$

which represents the fact that a variable subject $\$\mathbf{s}_0$ is associated with a variable object $\$\mathbf{o}_0$ in an associative table related to the predicate variable $\$\mathbf{p}_0$ with a level of belief τ_0, i.e., in the triple store,

$$\mathbf{t}_{pso} = \tau_0\, \mathbf{B}_{\$\mathbf{p}_0}\, \mathbf{B}_{\$\mathbf{s}_0}\, \$\mathbf{o}_0 + \cdots ,$$

where we use the \$ prefix to make explicit the fact that a given symbol is a variable symbol.

We are going to consider entailment rules of the form

$$\bigwedge_{i=1}^{i=I} (\$\mathbf{s}_i, \$\mathbf{p}_i, \$\mathbf{o}_i.) \rightarrow \bigoplus_{j=1}^{j=J} (\$\mathbf{s}_j, \tau_j(\cdot)\, \$\mathbf{p}_j, \$\mathbf{o}_j.),$$

where

- The left-hand-side expression corresponds to a conjunction of premises, with $\$\mathbf{x}_i$ ($\$\mathbf{x}$ stands for the subject, predicate, or object) receiving the corresponding item as input.

[14]See, for instance, https://www.w3.org/TR/owl2-profiles/#OWL_2_RL for details.
[15]https://en.wikipedia.org/wiki/Semantic_Web_Rule_Language
[16]https://en.wikipedia.org/wiki/Inductive_logic_programming

- The weight $\tau_j(\cdot)$ is a function of left-hand-side elements; it is equal to 0 if the rule does not apply to the element, it is negative if the consequence is to partially delete an element, and it is positive otherwise.
- Each right-hand-side value $\$\mathbf{x}_j$ ($\$\mathbf{x}$ stands for the subject, predicate, or object) is either a constant or equals a left-hand-side element $\$\mathbf{x}_i$, or it could be a more complex expression of left-hand-side elements.
- The right-hand-side expression corresponds to $\mathbf{t}_{pso} \mathrel{+}= \sum_j \tau_j(\cdot)\, \mathbf{B}_{\$\mathbf{p}_j}\, \mathbf{B}_{\$\mathbf{s}_j}\, \$\mathbf{o}_0$, i.e., it updates the related structure. In almost all the cases that we have considered, only one triple is generated; this part of the setup is thus stated for future use.

This includes modifying (including deleting) existing triplets since they are weighted by a τ value that can be modified and thus set to 0. The key point is that the calculation of $\tau_j(\cdot)$ allows the integration of not only modal logical formulae but also threshold mechanisms or counting operations that are required for induction.

Such a general setting clearly corresponds to the usual production rules. In the binary mode, e.g., if $\tau \in \{0, 1\}$, this is nothing but a specification equivalent to Horn clauses. Introducing the calculation of $\tau_j(\cdot)$ allows us to go beyond this, but at the cost of lowering the decidability and implementation tractability, as detailed later in this paper, so we will need to consider suitable convergence conditions.

At the implementation level, the application of such rules can simply be implemented using feedback connections between an iterator over the triples of the relational map, as discussed in detail for a typical example before a general implementation is described.

4.3 Class inheritance entailment rule

In order to better understand what is proposed here, let us start by detailing an illustrative and quite universal example.

4.3.1 Class inheritance as a major deductive mechanism

The most common entailment rule is likely the class inheritance rule, which states that "if a subject belongs to a class, and if this class is a subset of a superclass, then the subject belongs to the superclass," e.g., "if Tom is a cat, and cats are animals, then Tom is an animal." This is a deductive rule, i.e., a particular syllogism, which belongs to common-sense reasoning and is well understood as soon as formal reasoning emerges in children [21].

The notion of "class" corresponds to Boolean properties (e.g., if you are alive, you belong to the class of living organisms) and capabilities (e.g., if you can fly, you belong to the class of flying organisms). It is associated with a given individual feature. In addition, such Boolean features are organized in a hierarchy, leading to a taxonomy that describes the information about the considered individuals.

At the syntax level, it is based on two predicates, namely, **is_a**, which states that an individual belongs to a class, and **are**, which states that all individuals of a class are also individuals of a more general class. They correspond, respectively, to **rdf:type** and **rdfs:subClassOf** when the RDFS vocabulary is used, as discussed in Appendix C.

The implementation of the class inheritance rule using the VSA has already been successfully developed and numerically tested in [13] using the Nengo simulator [36], while we propose here to rely on this work to propose a more general mechanism. In Appendix C, we make explicit the fact that the mechanisms of a commonly used semantic web modeling language, the Resource Description Framework Schema (RFDS), can be implemented in such a numerical framework.

4.3.2 Design of the inference rule

The class inheritance rule can be written in its "binary equality" form as follows:

$$(\$\mathbf{s}, \mathbf{is\_a}, \$\mathbf{c}_1.) \wedge (\$\mathbf{c}_1, \mathbf{are}, \$\mathbf{c}_2.) \Rightarrow (\$\mathbf{s}, \mathbf{is\_a}, \$\mathbf{c}_2.),$$

which states that if any subject $\$s$ belongs to the class $\$\mathbf{c}_1$, i.e., is of "type" $\$\mathbf{c}_1$, and this class $\$\mathbf{c}_1$ is a subclass of $\$\mathbf{c}_2$, then $\$\mathbf{s}$ also belongs to the class $\$\mathbf{c}_2$. Here, the equality is either verified or not; we are in the exact reasoning case.

When considering an approximate form, as schematized in Fig. 5, the rule is written as follows:

$$(\$\mathbf{s}_1, \$\mathbf{p}_1, \$\mathbf{o}_1.) \wedge (\$\mathbf{s}_2, \$\mathbf{p}_2, \$\mathbf{o}_2.) \Rightarrow (\$\mathbf{s}_1, \tau\, \mathbf{is\_a}, \$\mathbf{o}_2.),$$

with

$$\tau \overset{\text{def}}{=} (\$\mathbf{p}_1 \cdot \mathbf{is\_a}) \;\&\; (\$\mathbf{o}_1 \cdot \$\mathbf{s}_2) \;\&\; (\$\mathbf{p}_2 \cdot \mathbf{are}),$$

so that τ equals 0 unless the three equalities are at least partially verified. If τ equals 0 nothing is inferred, whereas if it is greater than 0, a new triple is output. It is a forward schema in the sense that given some input data, a new result is inferred. This second form allows approximate correspondences, e.g., $\$\mathbf{p}_1$ can be approximately **is_a**.

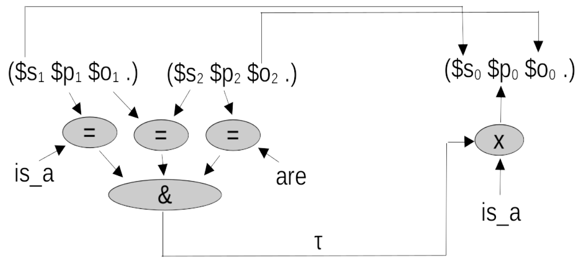


Fig. 5 The forward implementation of the class inheritance entailment rule.

The rule requires two numerical operators:
- the similarity operator $(\mathbf{x} \cdot \mathbf{y})$, and
- a numerical implementation of the conjunction $(\mathbf{x} \;\&\; \cdots \;\&\; \mathbf{y})$ operator, which is equal to
  - 0 if one operand is equal to or less than 0,
  - 1 if all operands are equal to 1,

with intermediate values if the operands are between 0 and 1. This could be a min operator in accordance with the numeric modal logic implementation, as discussed in [25], and this is what we propose here; it is easily implementable[17] at a neuronal level [37], while any other T-norm[18] would be suitable.

4.3.3 Forward versus backward inference

This previous setup corresponds to a forward application of the rule, i.e., given two left-hand-side triples, it describes how to numerically calculate the right-hand side.

In its backward form, given the right-hand side, we calculate

$$\begin{aligned}\tau \overset{\text{def}}{=} & (\$\mathbf{s}_0 \cdot \$\mathbf{s}_1) \;\&\; (\$\mathbf{o}_0 \cdot \$\mathbf{o}_2) \;\&\; (\$\mathbf{p}_0 \cdot \mathbf{is\_a}) \\ & \;\&\; (\$\mathbf{p}_1 \cdot \mathbf{is\_a}) \;\&\; (\$\mathbf{o}_1 \cdot \$\mathbf{s}_2) \;\&\; (\$\mathbf{p}_2 \cdot \mathbf{are}),\end{aligned}$$

i.e., given a right-hand-side input, we evaluate to what extent two left-hand-side triples could correspond to what is expected. See, e.g., [38] for an overview of this forward versus backward duality.

In the brain, as discussed, for instance, in [39] at a computational level and [40] at a more conceptual level, both forward and backward mechanisms are mixed in cognitive behaviors, as studied, for instance, in [41] at a more experimental level. In a nutshell, entailment rules are applied both "on query" when, given a goal-directed behavior, some information is required and at a more data-driven level, which here is the stimulus-driven level. In this paper, we are going to focus on forward inference, while in [13], a backward mechanism based on the VSA, for a specific couple of inference rules, was proposed and validated.

4.3.4 Exact versus approximate inference

This rule thus considers approximate similarity: If the input triple states that $\$\mathbf{s}_1$ is of class $\$\mathbf{o}_1$ only approximately, with $\$\mathbf{p}_1 = \tau\, \mathbf{is\_a}$, where $\tau \in [0, 1]$, and similar approximate equalities, while the similarity operator outputs a value between 0 and 1 and the conjunction operators interpolate values between 0 and 1 given the input, we will obtain as the output an appropriate predicate value $\tau'\, \mathbf{is\_a}$, with $\tau' \in [0, 1]$, which corresponds to the fact that the predicate is only approximately true.

[17]To be very precise, the max operator biological implementation is often discussed; it is obviously equivalent to a max up to linear transform

$$\min(x_1, \cdots x_n) = 1 - \max(1 - x_1, \cdots 1 - x_n), \text{ with } x_i \in [0, 1].$$

[18]https://en.wikipedia.org/wiki/T-norm

It is interesting to note that in the case of exact inference (i.e., $\mathbf{\$p}_1 = \mathbf{is\_a}$, $\mathbf{\$o}_1 = \mathbf{\$s}2$, $\mathbf{\$p}_2 = \mathbf{are}$), we obtain $\tau = 1$ as expected.

4.3.5 Implementation of negative inference

This, however, does not immediately generalize to negative inference (i.e., the fact that it approximately does *not* belong to a given class, which is beyond the RDFS specification but present at the OWL level). In this case, a symmetric rule

$$(\$\mathbf{s}, \neg\mathbf{is\_a}, \$\mathbf{c}_1.) \wedge (\$\mathbf{c}_2, \mathbf{are}, \$\mathbf{c}_2.) \Rightarrow (\$\mathbf{s}, \neg\mathbf{is\_a}, \$\mathbf{c}_2.)$$

has to be considered, with a similar development. The key point is that we must only consider the positive part of the similarity, i.e., it is combined with a rectification operator.

4.3.6 Implementation of the inference rule

At the implementation level, what is schematized in Fig. 5 is a simple analog calculation of a τ value given some left-hand-side input, while the backward implementation is a calculation that also involves the right-hand-side values: It is no more than a weighted connection between triples, i.e., between memory elements, with the generation of a new triple in the forward case and an evaluation of the queried triple in the backward case.

Since we consider approximate inference, and contrary to the binary equality case[19], we cannot select triples that exactly match a given pattern. In the forward mode, we must iteratively calculate $\tau(\cdot)$ on the whole conjunction of left-hand-side triples, generating new triples when $\tau(\cdot)$ is above a given threshold, as discussed in the general case below.

[19] In the binary equality case, let us illustrate our purpose by explicitizing how to generate the entailment rule closure, i.e., all deducible triples, given this rule and a set of input triples. An efficient mechanism must be able to select the pertinent triples, i.e., use a **select** operator, e.g., based on associative tables, indexed by property and subject. Given two constant values c_i and c_j,

```
for all ($\$s_i$ $\$p_i$ $\$o_i$.), $\$p_i = c_i$ & $\$s_i = c_j$ do
    ../..
end for
```

can be enumerated directly without scanning all triples by scanning only the one to be selected. This will allow us to incrementally calculate the closure, given a "closed" set of triples $\{(\$s_i\ \$p_i\ \$o_i.) \cdots\}$ with all possible triplets generated and a new triple $(\$s_0\ \$p_0\ \$o_0.)$ to be added. The standard algorithm is written as follows (see, e.g., [38] for an introduction):

```
input A new triple ($\$s_0$ $\$p_0$ $\$o_0$.) and a closed set of triples $\{(\$s_i\ \$p_i\ \$o_i.) \cdots\}$.
let $\{(\$s_0\ \$p_0\ \$o_0.)\}$ be an "open" triple set, initialized with the new triple inside.
repeat
    pull a triple ($\$s_0$ $\$p_0$ $\$o_0$.) from the open triple set.
    if $\$p_0$ = is_a then
        for all ($\$s_i$ $\$p_i$ $\$o_i$.), $\$p_i$ = are & $\$s_i = \$o_0$, in the closed triple set do
            add ($\$s_0$ is_a $\$o_i$.) to the open triple set.
        end for
    else if $\$p_0$ = are then
        for all ($\$s_i$ $\$p_i$ $\$o_i$.), $\$p_i$ = is_a & $\$o_i = \$s_0$, in the closed triple set do
            add ($\$s_i$ is_a $\$o_0$.) to the open triple set.
        end for
    end if
    add the triple ($\$s_0$ $\$p_0$ $\$o_0$.) to the closed triple set.
until the open triple set is empty.
```

4.4 Entailment rules in the general case

Going back to the general case, let us now describe an iterative closure algorithm, as schematized in Fig. 6, and discuss its properties.

The key idea is that the relational map is a two-level hierarchy with
- a known triples map storing all existing facts and all their derivations given a fixed set of entailment rules, and
- a new triples buffer that is input or feedback for entailment rules and for which derivations must still be done.

This is a very common setup used to calculate a fixed point or a closure.

For each new triple, we have to

```
- check if it is already a known triple; in this case, the job is done;
- otherwise,
 - for each entailment rule,
  - for each left-hand-side premise instantiated by the new triple,
   - for each known triplet set matching the other premises,
    - we have to calculate the related $\tau(\cdot)$ value and generate the corresponding new triple or triples; and
    - move this new triple from the new triple buffer to the known fact map.
```

At the initialization stage, the known triple map is empty, and the new triple buffer waits for an input.

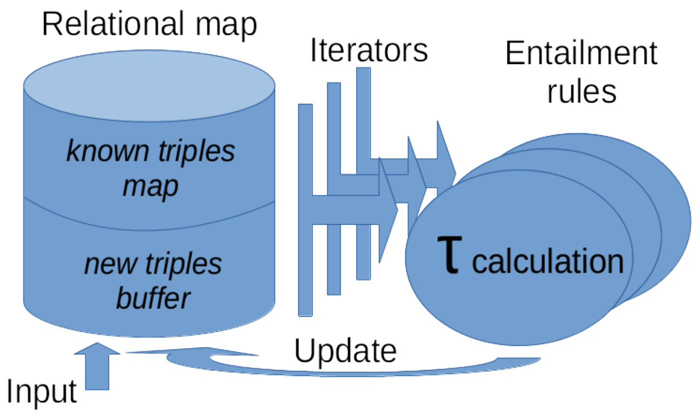


Fig. 6 The entailment rules feedback mechanism in the general case; see the text for details.

This basic mechanism raises several issues.

4.4.1 Comparing a new triple with a known existing triple

Given the relation map data structure, it is obvious that unbinding can be used to verify if the new triple has a similarity with a known triple of the relation map, as made explicit in subsection 3.3.2. This similarity corresponds to the product of the degree of belief of each triple and the cosine similarity.

Thanks to the development of a statistical test in section 2 and the easy first-order estimation of the noise level, which is made explicit in Appendix D at the implementation level, we have available, given a probability threshold, a mechanism to decide if the statistical difference between the degrees of belief of the new and known triples is negligible or not.

If it is not negligible, this means that both triples are similar and
- either the new triple enforces the degree of belief of the known one, up to the upper bound, which is 1, corresponding to the "true" value,
- or it decreases, potentially to the point that it contradicts the previous degree of belief (setting the value to -1, which corresponds to the "false" value, or setting it to 0, which corresponds to an "unknown" state).

This is very interesting but it has a drawback: It may generate an unstable state. In the binary case, the only degree of belief value is 1, and thus the only possibility is that the new triple equals the known one, meaning that we can avoid redoing the inference. At a more general level, if the only possibility is that the new triple enforces the degree of belief of the known one, we still have to redo inferences, but the system is monotonic, in the sense that the value can only increase and is bounded, and thus the values must converge. A simple way to implement a monotonic mechanism is to take into account the new triple only if its level of belief value is higher than the previously known triple's belief value.

At this stage, it is really a semantic alternative, to be chosen at the application level, depending on whether we design a cumulative knowledge mechanism in a stable universe or an adaptive mechanism in a changing knowledge environment. Both are possible at this implementation level while developing this point is beyond the scope of our purpose.

4.4.2 Algorithmic complexity for a given new triple

For one step, given T inputs, R entailment rules of arity I (i.e., number of left-hand-side premises), and S known facts, these nested loops generate $O(T\,R\,I\,S^{I-1})$ calculations of $\tau(\cdot)$. Fortunately, for most of the rules, we have $I = 2$ or even $I = 1$, except for some fragments of OWL-RL entailment rules[20], which often have $I = 3$, so that the complexity is mainly linear with respect to the relational map and seldom quadratic. Furthermore, beyond a sequential system, in our case, we have a completely distributed setup, so that such an operation set is simple to implement in parallel at a certain stage:
- Each new triple must be treated in sequence in order to work with a stable known triples map;
- however, given a new triple, the rule enumeration, the left-hand-side assignation, and the known triple enumeration on other premises' locations can all be performed in parallel, generating new triples in any order.

[20]The situation is even more complex because some entailment rules require the management of a variable arity depending, for instance, on the data structure length. See [42] for a detailed discussion. What we proposed here is thus directly applicable to the RDFS language layer and opportunistically to some OWL 2 reasoning mechanisms.

4.4.3 Global algorithmic time and space complexity size

The global algorithmic complexities in time and space are linked, since each step generates one new triple, while these operations are easily implemented in a distributed framework.

This complexity is highly dependent on each rule; for instance, it is easy to verify that a reflexivity property generates a closure whose order of magnitude is linear with respect to the input triple, a symmetry property generates a closure whose order of magnitude is quadratic with respect to the input triple, and transitivity generates a closure equivalent to the number of paths in an oriented graph, meaning that the complexity can be exponential for a fully connected graph.

Regarding the space complexity, an important aspect is "catastrophic forgetting," i.e., what happens if too many random vectors are superposed in the relation map, meaning that approximate orthogonality cannot be guaranteed. This has been numerically studied in, e.g., [16], showing on the one hand that the VT mechanism we have chosen outperforms other representations based on the approximate inverse, and it outperforms other dense methods (but not the sparse one) in terms of the superposition capability. The order of magnitude is rather low, with about 30 symbols reaching 99% accuracy in a space of dimension 1000, while the best sparse methods have limited performances of about 50 symbols for the same dimension.

As is, the proposed mechanism is limited to toy applications; we have two tracks to improve. On the one hand, when randomly drawing almost orthogonal symbols, it is always possible to use, for instance, the modified Gram-Schmidt methods to improve the stability and obtain precise orthogonality, with the benefit of improving the overall stability. On N symbols, the complexity is quadratic in the number of symbols, i.e., it is $O(d\,N^2)$. On a sphere of dimension d, we can obviously draw d orthogonal symbols.

On the other hand, at the microscopic level, in a real spiking neuronal network, the dimension is several orders higher; a neuronal map typically corresponds to a state space of dimension 10^5, which is not an order of magnitude higher. Extrapolating the linear approximation of [16], which is shown in Fig. 4 of that paper, we obtain a bit more than 3000 symbols for $d = 10^5$ and a bit more than 30000 symbols for $d = 10^6$.

This is of course impossible to calculate on a mesoscopic scale by manipulating symbols of such a dimension, but we are going to introduce the idea of simulating the VSA mechanism at a macroscopic level, i.e., directly manipulating the algebraic symbols and predicting the cumulative noise of the result. We make explicit this alternative method in Appendix D and share a preliminary implementation of such a mechanism that is no longer limited by the vector space dimension.

5 A preliminary experiment

To illustrate the use of this mechanism, we reconsider the example proposed in [13] and show a minimal ontology in Fig. 7, limiting this preliminary experiment to deductive rules and using the RDFs vocabulary. This allows us to better compare the two approaches. Here, the class inheritance *rdfs9* rules described in this paper and the subject domain inference (*rdfs2*), object range inference (*rdfs3*), and property inheritance (*rdfs7*) entailment rules detailed in Appendix C are implemented (refer to this appendix for an explanation of why this choice makes sense).

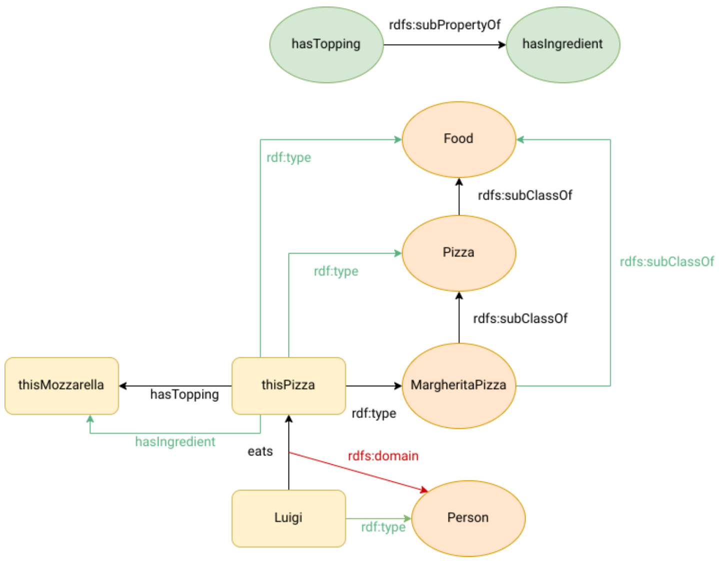


Fig. 7 An example of a simple ontology with three individuals. The black arrows correspond to factual statements input into the database and the green arrows correspond to inferred statements. Rectangular boxes stand for individuals, round boxes stand for classes, and properties are used to label arrows. 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.

5.1 Macroscopic implementation of the VSA system

As discussed throughout this paper, we are in a position to propose an algorithmic ersatz of the usual VSA mesoscopic linear algebra calculations involving high-dimensional random vectors, and this has been implemented and made available as public documented open-source code[21].

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, with some technical details about the noise calculation and software architecture given in Appendix D. We have considered a symbol encoding dimension of $d = 256$ to be consistent with previous mesoscopic experiments, such as those in [13]. This is tested with an associative map and relational map, as described in the previous section, and we have implemented the tiny Pizza experiment[22], obtaining, in the simplest case, the expected closure, as given in Fig. 8.

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)`
`(eats rdfs:domain Person)`	`(thisPizza hasIngredient thisMozzarella)`
`(eats rdfs:range Food)`	
`(thisPizza hasTopping thisMozzarella)`	
`(hasTopping rdfs:subPropertyOf hasIngredient)`	

Fig. 8 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.

Another interesting aspect is the calculation of the standard deviation of the level of noise modeling that happens at the mesoscopic level for either the calculation of the similarity between two random vectors or unbinding

[21] https://line.gitlabpages.inria.fr/aide-group/macrovsa/index.html

[22] 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, in accordance with Fig. 8, and it also shows the intermediate inference steps.

operations. On the one hand, our macroscopic model is consistent with that of [16], 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, and it would be interesting to further investigate, at the mesoscopic level, the numerical precision of unbinding on associative maps; to the best of our knowledge, this was not studied in the papers quoted here.

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.

5.2 Comparison with a mesoscopic implementation

Let us now discuss how, considering the NEF methodology [9], as implemented in the Nengo platform [36], the previous mechanisms can be implemented at the mesoscopic level. Such an implementation has already been proposed in [13] for the class inheritance *rdfs9* rules detailed in this paper, as a question-answering system, and thus it works in backward mode. In order to avoid lengthening the present paper, we do not describe again all the implementation mechanisms and expected results, but simply explicitize how to implement these mechanisms for this new approach.

The present implementation works in forward mode, computing the fixed point of the inference loops. To this end, it is possible to create a Nengo vocabulary containing all the resources of our ontology encoded as vectors, and we stored asserted memberships and relationships between these resources in associative memories[23], one for each predicate. For the sake of simplicity, we consider a simplified setup with $\tau = 1$ for each triple, while the use of

[23] https://www.nengo.ai/nengo-spa/examples/associative-memory.html

approximate inference is also discussed in [13] for a Nengo implementation using the VTB operator[24].

The RDFS entailment rules are implemented as feedback computations, with retro-actions enriching the associative memory's contents, as schematized in Fig. 9. We illustrate here the network connections for *rdfs9* entailments (in forward mode), but similar networks can be implemented for the other rules:
- Two input cues are fed into the network (in this case, they are initialized as TYPE and SUB_CLASS_OF because these are the predicates involved in applying the rule).
- Then, each associative memory stores a chained list of all known triples for a given predicate. A feedback connection makes it possible to use the last retrieved value as the key for the next value, making it possible to enumerate the triples just like a relational map, as discussed previously. Retrieved triples are successively stored in a state[25] that makes it possible to pass data, in this case, to enumerate each input triple, and thus it acts as the input triple buffer of Fig. 6.
- An action selection module[26] (corresponding roughly to a conditional test) makes it possible to nest the enumeration of the second associative memory by detecting the end of the enumeration of the first associative memory.
- Then, the rule itself is implemented by an interconnection similar to the one made explicit in Fig. 5, and another selection module acting as a threshold gate triggers the addition of a new triple if the rule is applicable.

Other entailment rules are implemented using similar networks; the associative memories are shared between them, while the enumeration mechanisms are generic, and only the inferred statement is specific to the given rules. The key point is that the whole computing process is distributed and "programmed" only by spatial interconnections, as expected.

This shows that our macroscopic implementation can translate to what is proposed to represent biologically plausible processes at a level of representation that can be "compiled" as a set of interacting spiking neural networks [9].

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

[24]Its transpose (TVTB) can also be used to manage left and right binding: https://www.nengo.ai/nengo-spa/modules/nengo_spa.algebras.html#nengo_spa.algebras.tvtb_algebra.TvtbAlgebra.

[25]https://www.nengo.ai/nengo-spa/modules/nengo_spa.modules.html?highlight=state#nengo_spa.modules.State

[26]https://www.nengo.ai/nengo-spa/modules/nengo_spa.html?highlight=actionselection#nengo_spa.ActionSelection

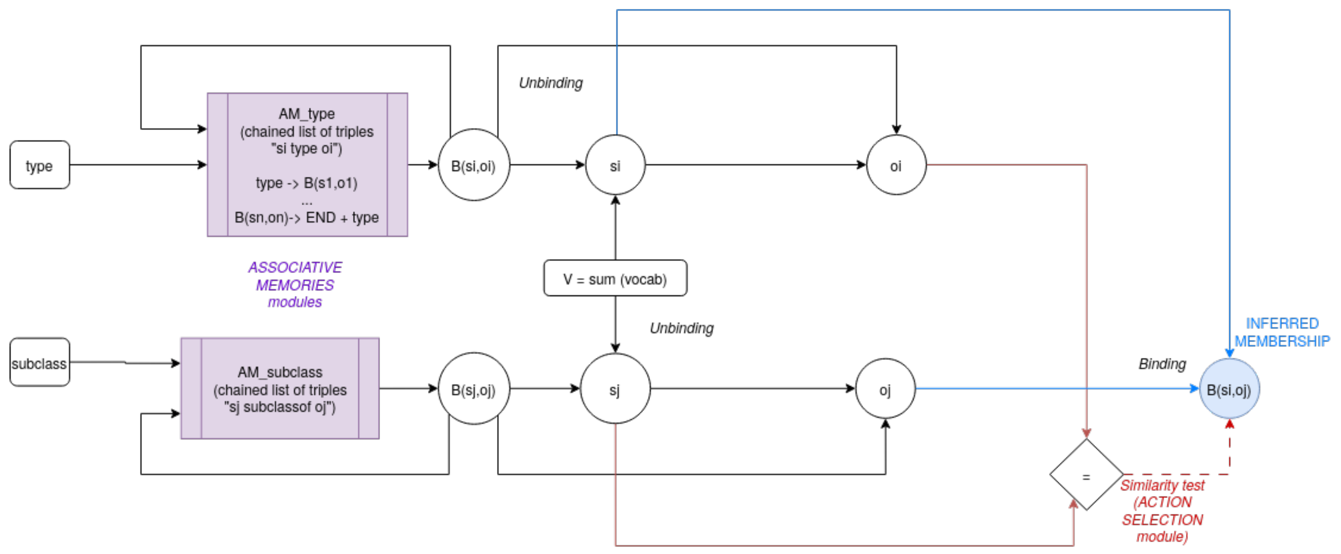


Fig. 9 The Nengo architecture for the class inheritance *rdfs9* forward inference rules; see the text for details. Associative memories storing the relationships are represented by purple rectangles (one for each predicate of interest). We did not represent here the action selection module that triggers the enumeration of the second associative memory, but the one triggering the application of the rule is accounted for by the similarity test explicitized in the red diamond. The rule itself requires the subject and object to be unbound from the retrieved triple; this is done by performing unbinding with the superposition of all semantic pointers from the vocabulary (this is possible thanks to the distributivity of VTB algebra), making it possible to retrieve an isolated vector regardless of what it was bound to. Finally, simply binding the subject s_j to the object o_i makes it possible to infer a new membership (if the gating condition applies), as shown by the blue connections.

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 suggested a design for a forward reasoning mechanism implemented via connectivity feedback on relational data structures, making it possible to perform deductive but also to some extent inductive and abductive reasoning.
- We introduced the idea of simulating such a mechanism at a macroscopic, more symbolic level in order to obtain 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.

6.2 On biological plausibility

We thus propose an anchoring, i.e., a numerical grounding of semantic information. This, indeed, does not mean that the brain performs such operations exactly, but this anchoring is biologically plausible in the sense that ontology rule-based 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 [9] with the NEF approach reviewed in this paper.

More technically, we hypothesize that such a neuron assembly stores the ontological assertions (i.e., A-box elements) as a compounded semantic pointer (SPA), while derivations of new assertions (i.e., T-box elements) are implemented as parameterized feedback on this SPA. What we show here is that a general feedback mechanism implements such derivations, while a specific derivation rule corresponds to a specific connectivity of the feedback input layer. This model implies that new rules are coded by new synapses' long-term connections of feedback connections, thanks to repetitive learning, for instance, due to prefrontal-hippocampus interactions, as modeled by, e.g., [43].

This model also implies that assertions are somehow enumerated off-line (in the sense of being scanned and replayed) to infer their consequences, corresponding to forward inference mechanisms (while we discussed a backward mechanism in [13], such a mechanism is also consistent with prefrontal-hippocampus interactions, as discussed in, e.g., [44]).

6.3 On numerical versus semantic grounding

Numerical grounding, or anchoring, fundamentally differs from the semantic symbol grounding problem, as reviewed and discussed in [45], 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[27], as discussed in, e.g., [46], in relation to the emergence of symbolic thinking (see, e.g., [47] for a detailed discussion). At the computational neuroscience level, the issue is addressed in [48] 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 priori or a posteriori. 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.

6.4 Approach limitations and perspectives

Although these contributions are of some theoretical and practical interest, it is clear that this is only a preliminary work, far from covering all the modeling and representative power of VSA approaches. It is rather limited regarding several aspects: Although the implementation of inductive and abductive rule-based mechanisms has been evoked and considered, this implementation has not yet been developed. While the RDFS deductive mechanism can clearly be implemented in the present framework, more sophisticated segments of knowledge representation languages such as the OWL 2 family must still be explored to evaluate what can be implemented without extending the present mechanism. Last but not least, another limitation is the fact that, depending on the choice of the entailment rules, an unstable or unbounded number of triples could be generated. This latter caveat can be easily avoided using our knowledge of the different levels of specification of semantic inferences, along with their related decidability and algorithm complexity properties.

The algorithm itself is a simple closure mechanism, although indeed, much more efficient reasoning algorithms are available (such as, e.g., the so-called "tableau methods"); however, to the best of our knowledge, their link to biologically plausible mechanisms is improbable, whereas we show that a simple set of feedback on iterators does the job and is much closer to what is expected to happen in neural assemblies (see [9] for a general discussion).

[27] 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.

The macroscopic simulator is operational but limited to the monotonic entailment rules used in forward mode. We have not yet implemented the capability to delete some rules that could be contradicted by new incoming facts or backward reasoning. Both could be easily implemented as a direct extension of the proposed software package, as described in this paper, but both also require one to make the proper design choices in order for this more general behavior to be coherent.

Following the usual VSA approaches, the symbolic information is embedded in a compact Riemannian manifold with a very simple topology, a hypersphere, 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 [49] 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 [50], showing that the satisfiability and algorithmic complexity can be drastically different[28]. 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 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 Inria, AExAIDE exploratory action.

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

Appendix A 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 [51] and complete the developments in that

[28] A version of these elements intended for a wider audience is available in a science popularization journal: https://interstices.info/calculer-dans-un-monde-hyperbolique.

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 Appendix D to derive the macroscopic computations.

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:

$$\mathbf{z} \stackrel{\text{def}}{=} \mathbf{B_y}\, \mathbf{x},$$

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

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

or equivalently[29], for $i = 1 \cdots d$,

$$\left\{ \begin{array}{rcl} [\mathbf{z}]_i \stackrel{\text{def}}{=} \mathbf{B_y}\, \mathbf{x} & = & \sqrt{d'} \sum_{k=1}^{k=d'} [\mathbf{y}]_{k+\beta(i)}\, [\mathbf{x}]_{k+\alpha(i)}, \\ {[\mathbf{B_y}]_{ij}} & = & \sqrt{d'}\, \delta_{i \le d' \text{ and } j \le d'} [\mathbf{y}]_{k+\beta(i)-\alpha(i)}, \end{array} \right. \text{written} \left\{ \begin{array}{l} \alpha(i) \stackrel{\text{def}}{=} d' \,((i-1) \text{ div } d'), \\ \beta(i) \stackrel{\text{def}}{=} d' \,((i-1) \text{ mod } d'), \end{array} \right. \tag{A1}$$

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 (A1) 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.

The $\sqrt{d'}$ renormalization factor allows $\mathbf{z}$ to have a unary magnitude[30].

At the algorithmic implementation level, the calculation of $\mathbf{z}$ is performed in[31] $O\left(d^{\frac{3}{2}}\right)$ 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 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.

[29]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.

[30]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 (A1) 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.

[31]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.

This can be compared to the fastest binding operation, which is convolution implemented via the fast Fourier transform [16], and thus it has a complexity of $O\left(d\,\log(d)\right)$:

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

As stated in [51] and reviewed in [13], the key point is that this binding operation generates a new vector $\mathbf{z}$ that is almost orthogonal to $\mathbf{x}$ and $\mathbf{y}$:

$$(\mathbf{B_y}\,\mathbf{x}) \cdot \mathbf{x} \simeq 0,$$

and this operation is neither commutative,

$$(\mathbf{B_x}\,\mathbf{y}) \cdot (\mathbf{B_y}\,\mathbf{x}) \simeq 0,$$

nor associative[32], in the following sense:

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

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.,

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

for the same reasons evoked above. It is straightforward to evaluate the precision of this approximation, which is again in $\mathcal{N}(0, O(1/d'))$ for each matrix element[33]. This also means that the level of noise is relatively high, $O\left(\frac{1}{d^{\frac{1}{4}}}\right)$ instead of $O\left(\frac{1}{d^{\frac{1}{2}}}\right)$, which explains the relatively limited numerical performances of simulations with $d < 10^3$, as reported, for instance, in [16].

We thus define

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

with

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

In other words, $\mathbf{B_y^\top}$ has the same structure as $\mathbf{B_y}$, except that the vector coordinates are subject to a permutation $\sigma(i)$, which is idempotent $(\sigma(\sigma(i)) =$

[32]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.

[33]From (A1), we derive

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

so that

- $\left[\mathbf{B_y^\top}\,\mathbf{B_y}\right]_{ii} = \sum_{l=1}^{d'} [\mathbf{y}]_l^2 = 1 + \mathcal{N}(0, O(1/d'))$, as derived in subsection 2.1; and
- $\left[\mathbf{B_y^\top}\,\mathbf{B_y}\right]_{ij, i \neq j} = \mathcal{N}(0, O(1/d'))$, 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 normal noise, as made explicit in subsection 2.1.

i) and thus its own inverse, so that if $\mathbf{z}' \overset{\text{def}}{=} \mathbf{B}_{\mathbf{y}^\sim}\,\mathbf{x}$, we obtain

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

(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:

$$\mathbf{B}_{\mathbf{y}^\sim}\,(\mathbf{B}_\mathbf{y}\,\mathbf{x}) = \mathbf{B}_\mathbf{y}^\top\,\mathbf{B}_\mathbf{y}\,\mathbf{x} = \mathbf{x} + \mathcal{N}(\mathbf{0}, O(1/d)) \simeq \mathbf{x}.$$

Thus, again, this results in additive noise of $O(1/d^{1/4})$.

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

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

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

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

(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:

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

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 [51], the authors of [13] introduced a vector composition operator $\oslash$ to make explicit the composition of two binding operations, namely,

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

which can be explicitly written as follows[34]:

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

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:

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

[34]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

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

from which we obtain the desired formula.

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' follows a normal distribution of standard deviation $O(1/d')$. We also find that

$$[\mathbf{v}]_i = \mathcal{N}(\mathbf{0}, O(1/d'^2)),$$

which is a rather high level of noise, when an unbinding operation is applied, for instance, to an associative map.

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 [16]. 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[35]:

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

while the orientation is usually defined as

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

as explained in the previous footnote.

Provided that the space dimension d is large enough, two randomly chosen different complex vectors $\mathbf{x}$ and $\mathbf{y}$[36] will also be approximately orthogonal in

[35]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

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

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:

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

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

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

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.,

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

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

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.

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 $\mathbf{s}$:

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

which provides a solution, because given a symbol $\mathbf{s}_\bullet$, we obviously 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 [16] 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,

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

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

This obviously 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.

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

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.

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. The main difference is that 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[37]. 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 [10], 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 select all elements and perform an operation on each one. However, at the implementation level, in Nengo [9], 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 paper, we will also propose a biologically plausible indexing mechanism, in order, for instance, to manipulate sequences.

B.2 Associative map

We now consider an unordered associative memory[38], 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 A, with a pseudo-inverse, i.e., an unbinding operator, $B_{\mathbf{s}_i^\sim}$:

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

so that

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

In other words, the unbinding operation makes it possible to retrieve the set,

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

[38]This is not the only way to implement such an associative memory: In [52], binding/unbinding is not explicitly used, and 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.

i.e., the additive superposition of all objects $\mathbf{o}_i$ associated with a given subject $\mathbf{s}_\bullet$, while $B_{\mathbf{s}_k^\sim}\,\mathcal{S} \simeq 0$ if none. This is done up to a level of noise of $O(1/d^{1/4})$, as derived in Appendix A, which is rather high with respect to the similarity precision, which is $O(1/d^{1/2})$, as observed numerically [16]; 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 and retrieve this information in one step if this is the case. However, as in the previous case, no mechanism allows the enumeration of the map subjects or objects.

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

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

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

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

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

In other words, the binding reduces to a superposition. Theoretical details underlying the implementation of such associative memories are available in [14].

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 A after the work in [51], which was completed by [13]. This design choice is guided by the fact that we need to avoid spurious inferences: With a 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 [16] for a recent comparison of different VSAs)[39].

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 [15].

In particular, a structure of the form

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

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

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 $\mathbf{s}_i \rightarrow \mathbf{o}_i$ is added, we also add $\mathbf{s}_i \rightarrow \sigma$ and $\sigma \rightarrow \mathbf{o}_i$, 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,

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

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

B.3 Indexed and chained list

B.3.1 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 [53] 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:

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

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 [54] beyond the scope of this work.

B.3.2 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 $\mathbf{l} \overset{\text{def}}{=} \sum_i B_{\nu_i}\, \mathbf{l}_i$, and a variable index $\mathbf{k}$. A construct of the form

```
**for** $\mathbf{k} \leftarrow \nu_0$; **while** $\|B_{\mathbf{k}}\, \mathbf{l}\| > 0$; **next** $\mathbf{k} \leftarrow B_{\nu_0}\, \mathbf{k}$ **do**
    $\mathbf{l}_i \leftarrow B_{\mathbf{k}^\sim}\, \mathbf{l}$
    ../..
**end for**
```

allows us to enumerate[40] 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 [15], we may consider that the brain can have three kinds of memory: associative, sequential, and hierarchical (called schematic by the author of [15]) 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 with regard to 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.3.3 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.

[40]In fact, considering $\mathbf{l} \overset{\text{def}}{=} \sum_i B_{\nu_i}\, \mathbf{l}_i + B_{\nu_{-1}}\, \lambda$, 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.

We also could have considered multiple binding[41], as proposed in [13].

Appendix C RDFS entailment rules implementation

As a side product of the present development, we would like to illustrate the computational capability of the proposed framework by briefly showing that our biologically plausible mechanism is at least able to perform RDFS[42] specification inferences. This is done, of course, without any assumption about the fact that the brain explicitly performs such computations[43].

RDFS modeling in a nutshell

In order to represent symbolic information, the RDFS knowledge representation is based on the RDF data model, which represents knowledge as triples, as made explicit in Fig. 3. More precisely, the *universe of discourse* is made of *resources*, referenced by some universal International Resource Identifier (IRI), i.e., a fixed lexical token. To structure this universe of discourse, we consider

(i) *individuals* that refer to real-world concrete or abstract objects;
(ii) *literals* to characterize individuals using data attributes, i.e., numerical values, character strings, or any structured information, such as dates;
(iii) *concepts* and *roles* (namely *classes* and *properties*) that allow us to structure the knowledge about individuals.

In fact, in our context, which is outside the web semantic application field, we have to point out that the RDF/RDFS framework has to be considered with the following variants:

- We conflate *name* with both the IRI and blank node since on the one hand, the blank node can be eliminated[44], and on the other hand, we only process the information locally at this stage, thus avoiding considering all issues regarding distributed information between different sources.

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

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

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 A, chaining unbinding operations is not numerically very robust due to the additional residual noise.

[42]This is true according to the https://www.w3.org/TR/rdf-schema specification.

[43]Considering knowledge representation and reasoning, the capabilities of semantic web modeling languages, such as RDFS and Web Ontology Language (OWL) (see, e.g., [55] for a recent didactic reference), make them a powerful way to solve modeling problems and manipulate high-level data representations. They also appear to be rather accessible to an educated person, as pointed out in [55], and correspond in part to the usual common-sense formalization, for instance, in terms of the notion of class (e.g., Garfield is a cat, which is an animal, which is a living organism). The brain is capable of such reasoning, as discussed in the introduction, although the data representation and processing mechanism is obviously different from what is performed in semantic web modelers and reasoners; the brain generally performs induction and abduction more than deduction, as reviewed here.

[44]It can be eliminated using a standard process related to Skolemization.

- We do not consider (i) semantic web-specific literals (e.g., `rdf:XMLLiteral`) or (ii) utility and annotation or other human-targeted properties (e.g., `rdfs:seeAlso`) at this stage.
- We will introduce both containers, i.e., ordered or unordered sequences, and collections, i.e., chained lists, later in these specifications, but in a somewhat different form, adapted to the numerical representation and easy to map to RDF representations.
- We do not consider all XSD data types, but we will introduce a precise notion of numerical values and describe how to represent structured data in our framework.

Given its ability to state facts, the RDFS framework allows us to structure concepts using the following construct:
- The notion of class inheritance (e.g., if Tom is a cat, and cats are animals, Tom is an animal) allows us to define a hierarchical taxonomy of classes, structure the objects in categories, and infer all that is possible from this taxonomy.
- The notion of property inheritance (e.g., if Tom is the brother of Jerry, Tom is in the same family as Jerry; the property of being in the same family is a super-property of being brothers) allows us to structure properties and also infer new properties by inheritance.
- The notion of domain and range (e.g., if Tom is the brother of Jerry, this also means that Tom is a boy) allows classes for subjects and/or objects of a category.

The language also allows us to define additional information, such as human-readable elements, but we consider it as a demonstrative subset to consider the main notions reviewed here.

Generality of numeric entailment rule implementation

The RDFS entailment, i.e., all that can be logically deduced from the input information, defines which elements are well-formed and which entailment relations allow us to deduce all derived information. In the case of the RDFS framework, the entailment rules are given in Table C1. It appears that each rule can be implemented with the mechanism proposed in section 4:

- The *rdfs9* class inheritance entailment rule,

$$(\$s \ \texttt{rdf:type} \ \$c_1.) \wedge (\$c_1 \ \texttt{rdfs:subClassOf} \ \$c_2.) \Rightarrow (\$s \ \texttt{rdf:type} \ \$c_2.),$$

states that if any subject $\$s$ belongs to the class $\$c_1$ and this class $\$c_1$ is a subclass of $\$c_2$, then $\$s$ also belongs to the class $\$c_2$, as in the major example in subsection 4.3.

- The *rdfs2* rule makes it possible to infer the subject domain class:

$$(\$s_1 \ \$p_1 \ \$o_1.) \wedge (\$p_1 \ \texttt{rdfs:domain} \ \$o_2.) \Rightarrow (\$s_1 \ \texttt{rdf:type} \ \$o_2.)$$

yields

$$(\$s_1 \ \$p_1 \ \$o_1.) \wedge (\$s_2 \ \$p_2 \ \$o_2.) \Rightarrow (\$s_1 \ \tau \ \texttt{rdf:type} \ \$o_2.),$$

with

$$\tau \overset{\text{def}}{=} (\$p_1 \cdot \$s_2) \ \ \& \ \ (\$p_2 \cdot \texttt{rdfs:domain}).$$

- The *rdfs3* rule makes it possible to infer the object range class:

$$(\$s_1\ \$p_1\ \$o_1.) \text{ and } (\$p_1\ \texttt{rdfs:range}\ \$o_2.) \Rightarrow (\$o_1\ \texttt{rdf:type}\ \$o_2.)$$

yields

$$(\$s_1\ \$p_1\ \$o_1.) \wedge (\$s_2\ \$p_2\ \$o_2.) \Rightarrow (\$o_1\ \tau\ \texttt{rdf:type}\ \$o_2.),$$

with

$$\tau \overset{\text{def}}{=} (\$p_1 \cdot \$s_2)\ \ \&\ \ (\$p_2 \cdot \texttt{rdfs:range}).$$

- The *rdfs7* rule makes it possible to infer sub-property inheritance:

$$(\$s_1\ \$p_1\ \$o_1.) \text{ and } (\$p_1\ \texttt{rdfs:subPropertyOf}\ \$o_2.) \Rightarrow (\$s_1\ \$o_2\ \$o_1.)$$

yields

$$(\$s_1\ \$p_1\ \$o_1.) \wedge (\$s_2\ \$p_2\ \$o_2.) \Rightarrow (\$s_1\ \tau\ \$o_2\ \$o_1.),$$

with

$$\tau \overset{\text{def}}{=} (\$p_1 \cdot \$s_2)\ \ \&\ \ (\$p_2 \cdot \texttt{rdfs:subPropertyOf}).$$

This easily generalizes to all pertinent rules in Table C1. We also notice that the left-hand side contains only one or two triple patterns, as stated in the main paper. In detail, the rules can be described as follows:
- The *se1*, *se2*, *lg*, and *gl* rules are technical rules related to the management of blank nodes and literal allocation, which is not considered in our context. Similarly, the *rdf2* rule applies to XML literals, which are not considered here.
- The *rdf1*, *rdfs1*, *rdf4a*, *rdf4b*, *rdfs8*, *rdfs12*, and *rdfs13* rules are unary rules that state basic meta-properties of the language; they are easy to implement.
- The *rdfs10* and *rdfs11* rules implement the reflexivity and transitivity of classes (thus, they are very similar to *rdfs9*); they are again easy to implement.
- The *rdfs6* and *rdfs5* rules implement the reflexivity and transitivity of properties, while *rdfs7* is equivalent to *rdfs9* but for property inheritance, and thus again they are very similar.

Rule set	Rule name	If E contains:	then add:
Simple entailment rules	*se1*	`uuu aaa xxx .`	`uuu aaa _:nnn .,` where `_:nnn` identifies a blank node allocated to xxx by rule *se1* or *se2*.
	se2	`uuu aaa xxx .`	`_:nnn aaa xxx .,` where `_:nnn` identifies a blank node allocated to uuu by rule *se1* or *se2*.
Special case of rule *se1* for literals	lg (literal generalization)	`uuu aaa lll .`	`uuu aaa _:nnn .,` where `_:nnn` identifies a blank node allocated to the literal `lll` by this rule.
Special case of rule *se1* for literals (RDFS)	*gl* (literal instanciation)	`uuu aaa _:nnn .,` where `_:nnn` identifies a blank node allocated to the literal `lll` by rule `lg`.	`uuu aaa lll .`
RDF entailment rules	*rdf1*	`uuu aaa yyy .`	`aaa rdf:type rdf:Property .`
	rdf2	`uuu aaa lll .,` where `lll` is a well-typed XML literal .	`_:nnn rdf:type rdf:XMLLiteral .,` where `_:nnn` identifies a blank node allocated to `lll` by rule *lg*.
RDFS entailment rules	*rdfs1*	`uuu aaa lll .,` where `lll` is a plain literal (with or without a language tag).	`_:nnn rdf:type rdfs:Literal .,` where `_:nnn` identifies a blank node allocated to `lll` by rule *lg*.

	rdfs2	`aaa rdfs:domain xxx .`
		`uuu aaa yyy .`
			`uuu rdf:type xxx .`

	rdfs3	`aaa rdfs:range xxx .`
		`uuu aaa vvv .`
			`vvv rdf:type xxx .`

	rdfs4a	`uuu aaa xxx .`	`uuu rdf:type rdfs:Resource .`
	rdfs4b	`uuu aaa vvv .`	`vvv rdf:type rdfs:Resource .`

	rdfs5	`uuu rdfs:subPropertyOf vvv .`
		`vvv rdfs:subPropertyOf xxx .`
			`uuu rdfs:subPropertyOf xxx .`

	rdfs6	`uuu rdf:type rdf:Property .`	`uuu rdfs:subPropertyOf uuu .`

	rdfs7	`aaa rdfs:subPropertyOf bbb .`
		`uuu aaa yyy .`
			`uuu bbb yyy .`

	rdfs8	`uuu rdf:type rdfs:Class .`	`uuu rdfs:subClassOf rdfs:Resource .`

	rdfs9	`uuu rdfs:subClassOf xxx .`
		`vvv rdf:type uuu .`
			`vvv rdf:type xxx .`

	rdfs10	`uuu rdf:type rdfs:Class .`	`uuu rdfs:subClassOf uuu .`

	rdfs11	`uuu rdfs:subClassOf vvv .`
		`vvv rdfs:subClassOf xxx .`
			`uuu rdfs:subClassOf xxx .`

	rdfs12	`uuu rdf:type rdfs:ContainerMembershipProperty .`	`uuu rdfs:subPropertyOf rdfs:member .`
	rdfs13	`uuu rdf:type rdfs:Datatype .`	`uuu rdfs:subClassOf rdfs:Literal .`

Table C1 The RDF/RDFS entailment rules, reproduced from https://www.w3.org/TR/rdf11-mt.

Appendix D 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, of the memorization and processing operations for which we developed an abstract symbolic description previously. At a higher scale, when the VSA is implemented as described in Appendix A 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. 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."

Symbol indexing

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 vector $\mathbf{x}_k$ using an integer number k, incremented for each new symbol. Weighted symbols of index i and vector number k_i also have 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 σ_i, which is equal to 0 by default when no approximate operation has been applied to the symbol. Two symbols may thus have the same vector number but different belief levels or different noise levels. At the input/output level, the human-readable string (s_i) representation of the symbol is utilized, but it is not considered further here. The associative table of symbols is thus a simple associative array data structure of (k_i, τ_i, σ_i), without the explicitization of the vector value $\tau_i \, \mathbf{x}_{k_i} + \nu(\sigma_i)$.

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 A.

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 $\sigma_\bullet \overset{\text{def}}{=} O(1/d)$ 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[45] for the dot product

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

up to the first order, considering that the noise is independent of the vector values up to the first order. Here, $\delta_{\mathcal{P}} = 1$ if $\mathcal{P}$ is true and it is false otherwise. We can thus perform this operation without explicitly computing the dot product.

Here, we propose 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. 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 table.

On the other hand, considering a symbol of index j bound by a symbol of index i and unbound 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[46],

[45] The derivation is written as follows:

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

If $k_i \neq k_j$, then $\mathbf{x}_{k_i} \cdot \mathbf{x}_{k_j} = \nu(\sigma_\bullet)$ since these random vectors are approximately orthogonal up to normal noise with a standard deviation with an order of magnitude of $\sigma_\bullet$ [52], whereas if $k_i = k_j$, then $\mathbf{x}_{k_i} \cdot \mathbf{x}_{k_j} = 1$ since these are unary vectors.

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

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

yielding the expected result.

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

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

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

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

Compounded symbols

Given atomic symbols that are randomly drawn, using the enumerating operations given in Appendix A, we have to compute compounded symbols composed through bundling and binding (or unbinding if the symbol has been permutated), 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, since we use only linear algebra, it is straightforward to define an "oracle" that can calculate the result of all operations as follows:
- A symbol corresponding to the *bundling* of other symbols is fully defined by the symbol set, and an operation over a bundling (binding or similarity) results from applying the operation to each element and considering either the bundling (in the case of binding) or numerical sum (in the case of similarity) of the result.
- 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. The details of the implementation involve simple applications of the algebraic rules, and we refer the reader to the documented source itself for further details[47].

The commutator operator $\mathbf{B}_\leftrightarrow$ and the composition operator $\oslash$ are not considered here because they are not used in this application, but similar considerations would easily allow us to implement the same approach at a macroscopic scale.

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

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

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

because these random vectors correspond to centered random vectors.

[47] https://line.gitlabpages.inria.fr/aide-group/macrovsa.

References

[1] Ness, D., Farenga, S.J.: Knowledge Under Construction: The Importance of Play in Developing Children's Spatial and Geometric Thinking. Rowman & Littlefield, Lanham, MD, US (2007). p. xxiii, 257

[2] Arnett, J.: Adolescence and Emerging Adulthood: A Cultural Approach, 2nd edn. (2001). https://doi.org/10.1093/acprof:oso/9780195309379.001.0001

[3] Keefer, L.A., Landau, M.J.: Metaphor and analogy in everyday problem solving. WIREs Cognitive Science **7**(6), 394–405 (2016). https://doi.org/10.1002/wcs.1407. Accessed 2022-02-06

[4] Purves, D., Augustine, G.J., Fitzpatrick, D., Katz, L.C., LaMantia, A.-S., McNamara, J.O., Williams, S.M.: The interplay of emotion and reason. In: Neuroscience, 2nd edn. Sinauer Associates, Sunderland, MA (2001). https://www.ncbi.nlm.nih.gov/books/NBK10822/ Accessed 2022-03-22

[5] Garcez, A.d., Lamb, L.C.: Neurosymbolic AI: The 3rd wave. arXiv:2012.05876 [cs] (2020). Accessed 2021-05-26

[6] Badreddine, S., Garcez, A.d., Serafini, L., Spranger, M.: Logic tensor networks. arXiv:2012.13635 [cs] (2021). Accessed 2021-07-28

[7] Levy, S.D., Gayler, R.: Vector symbolic architectures: A new building material for artificial general intelligence. In: Frontiers in Artificial Intelligence and Applications, p. 6 (2008)

[8] Gayler, R.: 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

[9] Eliasmith, C.: How to Build a Brain: A Neural Architecture for Biological Cognition. OUP USA, ??? (2013). Google-Books-ID: BK0YRJPmuzgC

[10] Crawford, E., Gingerich, M., Eliasmith, C.: Biologically plausible, human-scale knowledge representation. Cognitive Science **40**(4), 782–821 (2016). https://doi.org/10.1111/cogs.12261. Accessed 2020-11-17

[11] Cessac, B., Viéville, T.: On dynamics of integrate-and-fire neural networks with adaptive conductances. Frontiers in Neuroscience **2**(2) (2008). Accessed 2022-03-23

[12] Eliasmith, C., Anderson, C.H.: Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. A Bradford

Book. The MIT Press, ??? (2002). https://mitpress.mit.edu/books/neural-engineering Accessed 2021-02-09

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

[14] Stewart, T.C., Tang, Y., Eliasmith, C.: A biologically realistic cleanup memory: Autoassociation in spiking neurons. Cognitive Systems Research **12**(2), 84–92 (2011). https://doi.org/10.1016/j.cogsys.2010.06.006. Accessed 2021-02-02

[15] Eichenbaum, H.: Memory: Organization and control. Annual Review of Psychology **68**(1), 19–45 (2017). https://doi.org/10.1146/annurev-psych-010416-044131. Accessed 2021-02-02

[16] Schlegel, K., Neubert, P., Protzel, P.: A comparison of vector symbolic architectures. arXiv:2001.11797 [cs] (2020). Accessed 2021-02-02

[17] Denœux, T., Dubois, D., Prade, H.: Representations of uncertainty in AI: Beyond probability and possibility. In: Marquis, P., Papini, O., Prade, H. (eds.) A Guided Tour of Artificial Intelligence Research: Volume I: Knowledge Representation, Reasoning and Learning, pp. 119–150. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-06164-7_4

[18] Denœux, T., Dubois, D., Prade, H.: Representations of uncertainty in AI: Probability and possibility. In: Marquis, P., Papini, O., Prade, H. (eds.) A Guided Tour of Artificial Intelligence Research: Volume I: Knowledge Representation, Reasoning and Learning, pp. 69–117. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-06164-7_3

[19] Fischer, B.: Modal Epistemology: Knowledge of Possibility & Necessity (2018). https://1000wordphilosophy.com/2018/02/13/modal-epistemology/ Accessed 2021-11-07

[20] Rusawuk, A.L.: Possibility and Necessity: An Introduction to Modality (2018). https://1000wordphilosophy.com/2018/12/08/possibility-and-necessity-an-introduction-to-modality/ Accessed 2020-09-24

[21] Smith, L.: The development of modal understanding: Piaget's possibility and necessity. New Ideas in Psychology **12**(1), 73–87 (1994). https://doi.org/10.1016/0732-118X(94)90059-0. Accessed 2021-07-30

[22] Tettamanzi, A., Zucker, C.F., Gandon, F.: Possibilistic testing of OWL

axioms against RDF data. International Journal of Approximate Reasoning (2017). https://doi.org/10.1016/j.ijar.2017.08.012. Accessed 2020-11-04

[23] Beynon, M., Curry, B., Morgan, P.: The Dempster–Shafer theory of evidence: An alternative approach to multicriteria decision modelling. Omega **28**(1), 37–50 (2000). https://doi.org/10.1016/S0305-0483(99)00033-X. Accessed 2021-11-07

[24] Vallaeys, T.: Généraliser les possibilités-nécessités pour l'apprentissage profond. Report, Inria (September 2021). https://hal.inria.fr/hal-03338721 Accessed 2021-12-15

[25] Viéville, T., Mercier, C.: Representation of belief in relation to randomness. Research Report RR-9493, Inria & Labri, Univ. Bordeaux (December 2022). https://hal.inria.fr/hal-03886219 Accessed 2023-03-28

[26] Gärdenfors, P.: Conceptual spaces as a framework for knowledge representation. Mind and Matter **2**, 9–27 (2004)

[27] Freksa, C.: Strong spatial cognition. In: Fabrikant, S.I., Raubal, M., Bertolotto, M., Davies, C., Freundschuh, S., Bell, S. (eds.) Spatial Information Theory. Lecture Notes in Computer Science, pp. 65–86. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23374-1_4

[28] McClelland, J.L., Rogers, T.T.: The parallel distributed processing approach to semantic cognition. Nature Reviews Neuroscience **4**(4), 310–322 (2003). https://doi.org/10.1038/nrn1076. Accessed 2021-04-29

[29] Noy, N.: Ontology Development 101: A Guide to Creating Your First Ontology. Technical report, Stanford University (2001). /paper/Ontology-Development-101%3A-A-Guide-to-Creating-Your-Noy/c15cf32df98969af5eaf85ae3098df6d2180b637 Accessed 2021-06-04

[30] Domingos, P.: Unifying instance-based and rule-based induction. Machine Language **24**(2), 141–168 (1996). https://doi.org/10.1023/A:1018006431188. Accessed 2022-04-15

[31] Lakkaraju, S.K., Zhang, Y.: Rule based abduction. In: Proceedings of the 12th International Symposium on Foundations of Intelligent Systems. ISMIS '00, pp. 525–533. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/3-540-39963-1_55

[32] Shanahan, M.: An abductive event calculus planner. The Journal of Logic Programming **44**(1), 207–240 (2000). https://doi.org/10.1016/S0743-1066(99)00077-1. Accessed 2022-09-11

[33] Khemlani, S.S., Barbey, A.K., Johnson-Laird, P.N.: Causal reasoning with mental models. Frontiers in Human Neuroscience **8** (2014). https://doi.org/10.3389/fnhum.2014.00849. Accessed 2021-11-18

[34] Ragni, M., Johnson-Laird, P.N.: Reasoning about possibilities: Human reasoning violates all normal modal logics. CogSci, 6 (2018)

[35] Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics **3**(1), 41–60 (2005). https://doi.org/10.1016/j.websem.2005.05.001. Accessed 2022-11-17

[36] Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T.C., Rasmussen, D., Choo, X., Voelker, A.R., Eliasmith, C.: Nengo: A Python tool for building large-scale functional brain models. Frontiers in Neuroinformatics **7** (2014). https://doi.org/10.3389/fninf.2013.00048. Accessed 2021-02-04

[37] Bugmann, G.: Biologically plausible neural computation. Biosystems **40**(1), 11–19 (1997). https://doi.org/10.1016/0303-2647(96)01625-5. Accessed 2022-11-15

[38] Kapoor, N., Bahl, N.: Comparative study of forward and backward chaining in artificial intelligence. International Journal of Engineering and Computer Science (2016). https://doi.org/10.18535/Ijecs/v5i4.32

[39] O'Reilly, R.C., Hazy, T.E., Mollick, J., Mackie, P., Herd, S.: Goal-driven cognition in the brain: A computational framework. arXiv:1404.7591 [q-bio] (2014). Accessed 2021-04-21

[40] Friston, K.: Learning and inference in the brain. Neural Networks: The Official Journal of the International Neural Network Society **16**(9), 1325–1352 (2003). https://doi.org/10.1016/j.neunet.2003.06.005

[41] Amidu, A.-R., Boyd, D., Gobet, F.: A protocol analysis of use of forward and backward reasoning during valuation problem solving. Property Management **37**, 638–661 (2019). https://doi.org/10.1108/PM-10-2018-0056

[42] Cao, S.T., Nguyen, L.A., Szałas, A.: The web ontology rule language OWL 2 RL + and its extensions. In: Nguyen, N.-T., Le-Thi, H.A. (eds.) Transactions on Computational Intelligence XIII. Lecture Notes in Computer Science, pp. 152–175. Springer, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54455-2_7

[43] Spens, E., Burgess, N.: A generative model of memory construction and consolidation. bioRxiv (2023). https://doi.org/10.1101/2023.01.19.524711

[44] Santos-Pata, D., Amil, A.F., Raikov, I.G., Rennó-Costa, C., Mura, A., Soltesz, I., Verschure, P.F.M.J.: Entorhinal mismatch: A model of self-supervised learning in the hippocampus. iScience **24**(4), 102364 (2021). https://doi.org/10.1016/j.isci.2021.102364. Accessed 2023-05-02

[45] Taddeo, M., Floridi, L.: Solving the symbol grounding problem: A critical review of fifteen years of research. Journal of Experimental and Theoretical Artificial Intelligence **17** (2005). https://doi.org/10.1080/09528130500284053

[46] Raczaszek-Leonardi, J., Deacon, T.: Ungrounding Symbols in Language Development: Implications for Modeling Emergent Symbolic Communication in Artificial Systems, p. 237 (2018). https://doi.org/10.1109/DEVLRN.2018.8761016

[47] Villiers, T.D.: Why Peirce matters: The symbol in Deacon's symbolic species. Language Sciences **29**(1), 88–101 (2007). Accessed 2023-02-01

[48] Rougier, N.P.: Implicit and explicit representations. Neural Networks **22**(2), 155–160 (2009). https://doi.org/10.1016/j.neunet.2009.01.00. Accessed 2022-04-02

[49] Nickel, M., Kiela, D.: 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 Accessed 2022-11-14

[50] Delahaye, J.-P.: Complexités : Aux Limites des Mathématiques et de L'informatique. HAL, ??? (2006). Number: hal-00731936. https://ideas.repec.org/p/hal/journl/hal-00731936.html Accessed 2022-11-18

[51] Gosmann, J., Eliasmith, C.: Vector-derived transformation binding: An improved binding operation for deep symbol-like processing in neural networks. Neural Computation **31**(5), 849–869 (2019). https://doi.org/10.1162/neco_a_01179. Accessed 2021-03-07

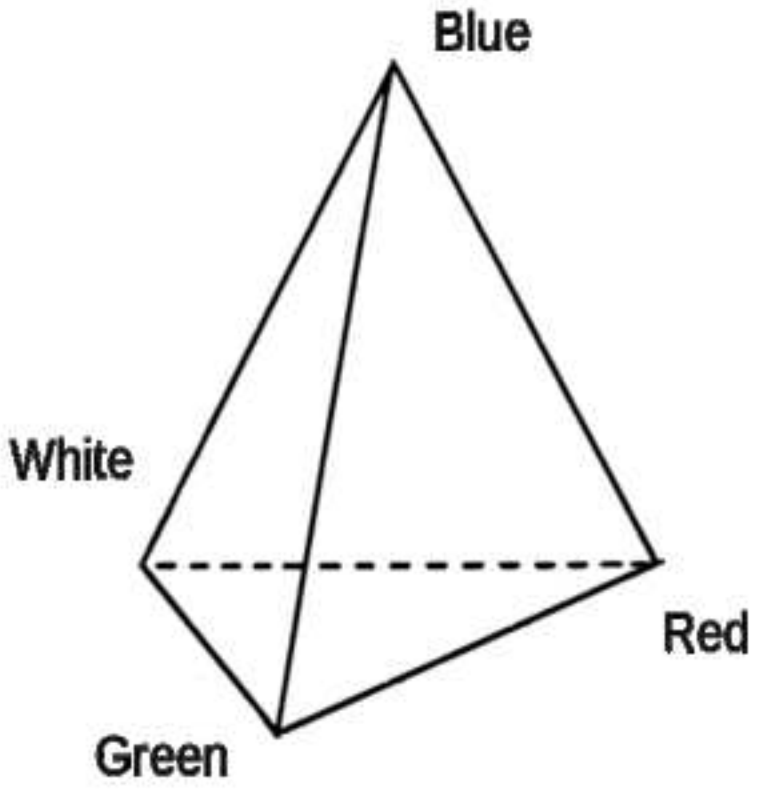
[52] Voelker, A., Crawford, E., Eliasmith, C.: Learning large-scale heteroassociative memories in spiking neurons. (2014). https://doi.org/10.13140/RG.2.1.1382.1688

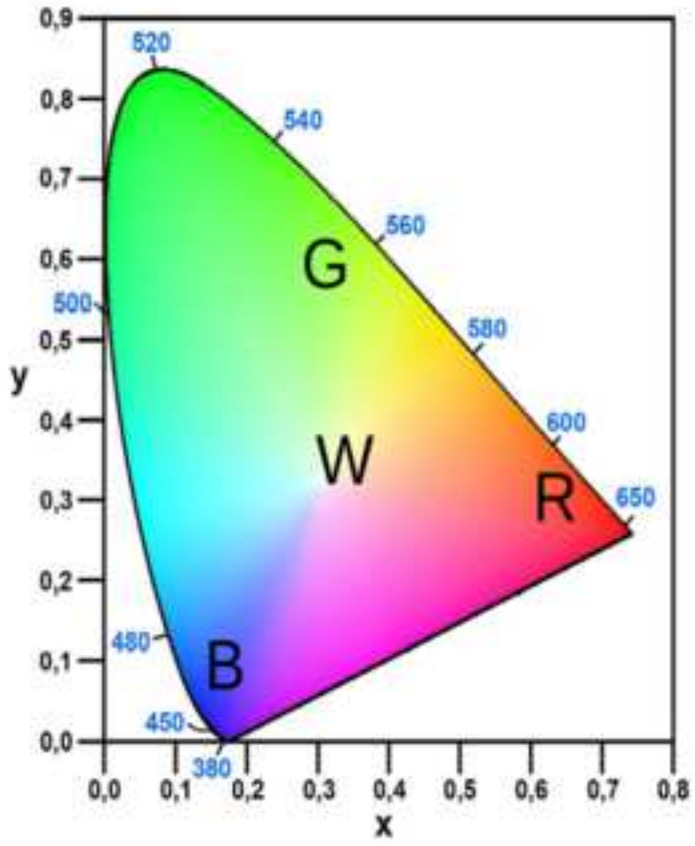
[53] Komer, B., Stewart, T.C., Voelker, A.R., Eliasmith, C.: A neural representation of continuous space using fractional binding. In: 41st Annual Meeting of the Cognitive Science Society, p. 6. Cognitive Science Society, Montreal, Canada (2019). http://compneuro.uwaterloo.ca/publications/komer2019.html

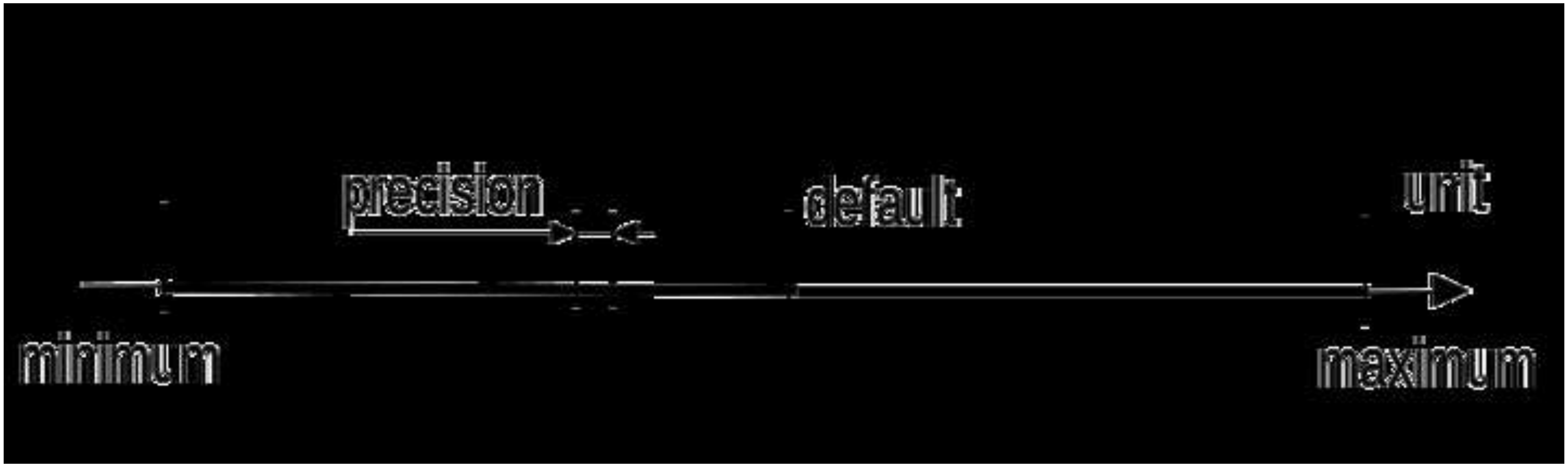
[54] Nieder, A.: Representation of numerical information in the brain. In: Funahashi, S. (ed.) Representation and Brain, pp. 271–283. Springer, Tokyo (2007). https://doi.org/10.1007/978-4-431-73021-7_11

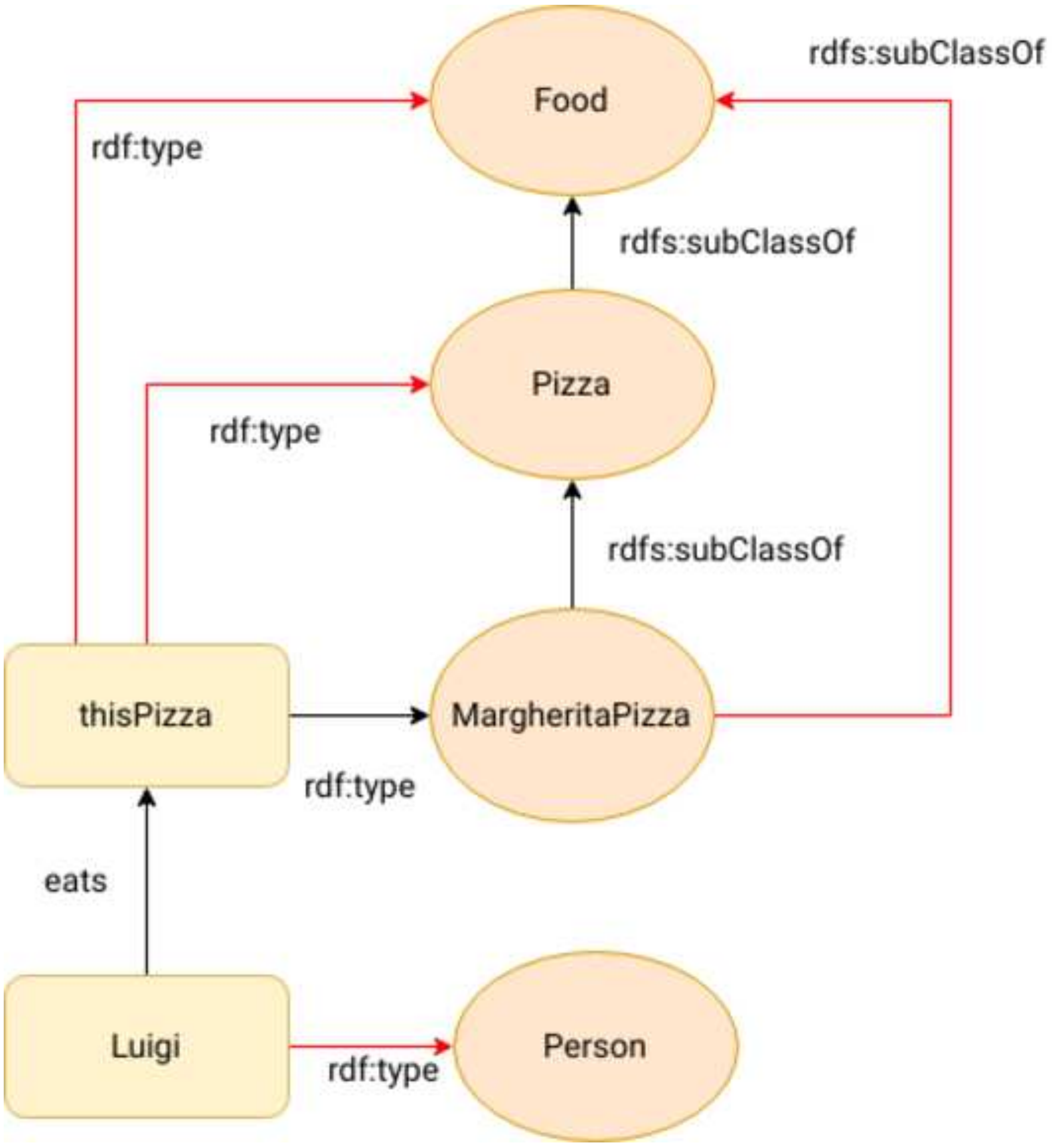
[55] Allemang, D., Hendler, J., Gandon, F.: Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL, 3rd edn. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3382097

Blue	1
Green	2
Red	3
White	4






precision
default
unit
minimum
maximum


Food
rdfs:subClassOf
rdf:type
rdfs:subClassOf
Pizza
rdf:type
rdfs:subClassOf
thisPizza
MargheritaPizza
rdf:type
eats
Luigi
Person
rdf:type

Ref overleaf: https://www.overleaf.com/2168222635knwfrrvkvqgm
Submission id: CODY-D-23-00028, Edition board login page: https://www.editorialmanager.com/cody/

Editor:

Dear Dr, Viéville

We have received the reports from our advisors on your manuscript, "Where is Aristotle in our brain? On biologically plausible reasoning embedded in neuronal computation.", which you submitted to Cognitive Neurodynamics.
Based on the advice received, I feel that your manuscript could be reconsidered for publication should you be prepared to incorporate major revisions.

Thanks a lot for offering a new chance to this work, we are especially thankful for the reviewers' feedback and have reworked the paper accordingly.

We really consider reviewer#2 feedback, as very precious, deeply helping us and it is absolutely clear that if this paper will be of some scientific interest, it will be thanks to her or his patient corrections and re-corrections. No will to "break" his or her anonymous status but if by no means it could be mentioned how much this paper has to acknowledge her/his benevolent work, please tell us.

Reviewer #1:

The authors have incorporated the necessary corrections in the manuscript.

Thanks a lot for this feedback.

Reviewer #2:

I thank the authors for responding to my previous comments and revising their manuscript. However, I am unfortunately still unable to follow the paper, let alone recognize its contributions. I have done research on VSA and have a decent understanding of logic, but still I am unable to understand many paragraphs in the paper. I might be to blame though, and I don't want a potential lack of proper background on my side to lead to a negative evaluation of the paper. So I refrain from giving any recommendations and ask the editorial board to seek other reviewers who may be better equipped to review this paper.

On our side, we are deeply thankful that you took the time to review this paper and really appreciate that you gave this paper a chance.

The first equation on page 5 needs magnitudes to be exactly 1, not just O(1).

Thank you for this remark, it is true that depending on the random draw it could be exactly one if the vector is renormalized, but only approximately 1 otherwise. We have removed the ambiguous O(1) shortcut and made the point explicit.

The second equation on page 5: is 1/sqrt(d) standard deviation or variance? Because multiplying this by sqrt(d) is said to give standard normal distribution below, which is not consistent with the conventional notation N(mu, sigma^2).

Sorry about that, we are now consistent in the whole paper with respect to the conventional notation.

After the second equation on page 5: What do these percent biases exactly mean? Why do we have a bias in the first place, and how are these calculated (for one of them, e.g.)?

We agree, and have now not only shared the source but described the simulation. Wee have also replaced the term "bias" with "relative precision".

First sentence of Section 2.2: Why? X has a similarity of 1 with X + Y when Y is independent from X, but X and X+Y are not semantically equivalent.

We fully agree with this comment. More than that, we have taken the point and proposed a footnote that make explicit, at the algebraic level, what is "semantic similarity" (as quoted by other authors). Yes, semantically similar does not mean equivalent, but contains an element that is semantically similar. It means that our sentence is unclear and we have rewritten it.

First equation of page 8: Why is this a good representation of belief? Why weight the symbol by someone's belief? Belief in what actually?

Following this remark, we have restructured the whole section:
 - make clearer this notion of belief
 - better (hopefully) providing arguments for our design choice and quoting previous works.

The paragraph after the first equation of page 8: The authors are drawing an association between two things that happen to range from -1 to 1: cosine similarity, and belief in possibility theory. But why is this a good association? If two vectors have a cosine similarity of 0, why is the answer to the question "are two vectors similar?" unknown? In my opinion the answer is no!

We understand your answer, we should have better made explicit the fact we are in an “open world” in which what is not true isi not necessarily false, but unknown. This is now better made explicit.

End of page 8: So concept = symbol = hypervector?

Concepts are represented by a symbol, indeed. And all symbols are encoded in hypervector. But not all symbols are concepts, except if an individual is considered as a concept “singleton”. This is now rewritten to precise this point.

Second paragraph of page 9: Why only these 4 predicates? "John likes apples", "John speaks Spanish", ... Aren't these knowledge?

Indeed, this is not obvious and the (McClelland & Rogers, 2003) design proposal requires a development not easy to report in a few words. We however, have taken benefit of this reviewer's remark to exemplify the idea and add a short complementary explanation. Quoting McClelland & Rogers: « In this approach, the four propositional relations from Quillian’s hierarchy are viewed as distinct contexts. The is a relation corresponds to a naming or explicit categorization context, in which the object’s name or category label is of relevance, as these might be indicated verbally (e.g., by a sentence such as “This is a bird” or “This is a canary”). The can relation corresponds to a context in which the behaviors of the object might be observed; the is relation corresponds to a context in which its appearance properties are highlighted; and the has relation corresponds to a context in which its parts are highlighted. »

Section 3.3.2: This section is supposed to be all about "relational maps", but what is a relational map ultimately?

Sorry, again. An introduction was required, now done adding a precise definition, making the link with the previous development more explicit.

I don't understand the significance or the point of defining t_{p_j}, nor the equation B_{p_j^~} s = t_{p_j}. What is s actually? And why is this true?

This is used at the implementation level. And the derivation is proposed in the related footnote. We have now rewritten this paragraph and corrected a typo in the notations, so sorry for that.

On page 13, there is a sentence that says "A key point is that ..." until "from t_{spo}". Why? Aren't the equations you give a few lines earlier doing exactly that?

Oh, this means that our notations are confusing between t_{pso} and t_{spo}: we have now more clearly mentioned this double similar notation.

Section 4.1: Neither OWL nor RDFS have been properly defined before.

Indeed, it is now added.