TTR at the SPA: Relating type-theoretical semantics to neural semantic pointers

Staffan Larsson and **Robin Cooper**
Dept. of Philosophy, Linguistics and Theory of Science
University of Gothenburg
sl@ling.gu.se, cooper@ling.gu.se

Jonathan Ginzburg
Université Paris Cité
yonatan.ginzburg@u-paris.fr

Andy Lücking
Université Paris Cité and Goethe University Frankfurt
luecking@em.uni-frankfurt.de

Abstract

This paper considers how the kind of formal semantic objects used in TTR (a theory of types with records, Cooper, 2023) might be related to the vector representations used in Eliasmith (2013). An advantage of doing this is that it would immediately give us a neural representation for TTR objects as Eliasmith relates vectors to neural activity in his semantic pointer architecture (SPA). This would be an alternative using convolution to the suggestions made by Cooper (2019a) based on the phasing of neural activity. The project seems potentially hopeful since all complex TTR objects are constructed from *labelled sets* (essentially sets of ordered pairs consisting of labels and values) which might be seen as corresponding to the representation of structured objects which Eliasmith achieves using superposition and *circular convolution*.

1 Introduction

Work on TTR, a theory of types with records, for example Cooper (2023), claims that it can be used to model types learned by agents in order to classify objects and events in the world. If this is true, types must be represented in some way in brains. In this paper we will explore the possibility of using Eliasmith's Semantic Pointer Architecture (SPA) (Eliasmith, 2013) for this purpose. The question of neural representations of types arises in connection with the theory of types proposed by TTR in a way that it does not in connection with more traditional type theories. The reason is that TTR aims to provide the kind of types that agents use in the perception of objects and events and which they use in interaction to communicate with each other. If it were to turn out that the kind of types used are in principle impossible to represent on arrays of neurons then this would call this project into question.

We chose SPA, since it is a model of a biological neural network. Notwithstanding their practical and methodological success, artificial neural networks (ANN) trained in deep learning leave open questions with respect to at least two areas of human cognition. Firstly, being sub-symbolic, it is unclear how they relate to 'Jackendoff's challenges'[1] (Jackendoff, 2002, §3.5) and to higher-order, symbolic processing as observed, for instance, in sentence processing (Goucha et al., 2017; Frankland and Greene, 2020a). Secondly, despite being inspired by the human brain and potentially useful for neuro-scientific research (Yang and Wang, 2020), ANNs differ from biological neural networks. The first issue is addressed by Vector Symbolic Architectures (VSA; Gayler, 2003; Schlegel et al., 2022), which define symbolic operations on high-dimensional numerical vectors.

The second issue is addressed by biological architectures, where high-dimensional vectors receive a neural interpretation in terms of spiking patterns (Eliasmith, 2013). Formal semantics provides symbolic systems for analysing natural languages. However, as Lücking and Ginzburg (2023, p. 149) argue, it is questionable whether traditional, 'anti-representationalist' formal semantics, which assigns truth conditions directly to sentences (Bezuidenhout, 2006) also lends itself to cognitive interpretations.

This is different with a Type Theory with Records (TTR; Cooper, 2023), which even has a neural interpretation (Cooper, 2019a). Indeed there has been a wide range of work in this formalism, introduced in section 3, which includes the modelling of intensionality and mental attitudes (Cooper, 2005, 2023), quantified NPs (Cooper, 2013; Lücking and Ginzburg, 2022; Cooper, 2023),

[1] Namely 'The massiveness of the binding problem', 'The problem of 2', 'The problem of variables', and 'Binding in working memory vs. long-term memory'.

Proceedings of the 4th Natural Logic Meets Machine Learning Workshop (NALOMA23), pages 41–50
June 20, 2023.

co-predication and dot types in lexical innovation, frame semantics for temporal reasoning, reasoning in hypothetical contexts (Cooper, 2011), spatial reasoning (Dobnik and Cooper, 2017), enthymematic reasoning (Breitholtz, 2020), self- and other-repair (Purver, 2006; Ginzburg et al., 2014), negation (Cooper and Ginzburg, 2012), non-sentential utterance resolution (Fernández et al., 2007; Ginzburg, 2012), iconic gesture (Lücking, 2016), multimodality (Lücking and Ginzburg, 2023) and symbol grounding (Larsson, 2015, 2021).

Accordingly, this paper offers a first attempt to combine TTR with a biologically-based VSA, namely the Semantic Pointer Architecture (SPA) of Eliasmith (2013). Sections 2 and 3 provide a brief overview of semantic pointers and TTR, respectively. How to 'translate' TTR objects into SPA is addressed in Section 4. We conclude in Section 5.

2 SPA (and NEF)

> [...] semantic pointers are neural representations that are aptly described by by high-dimensional vectors, are generated by compressing more sophisticated representations, and can be used to access those more sophisticated representations through decompression [...]. (Eliasmith, 2013, p. 83)

Hence, there are three perspectives on or levels of description for semantic pointers, namely (i) in terms of neural activation, (ii) as (high-dimensional) vectors, and (iii) as symbols. In this paper, we will not be concerned with the neural level beyond the assumption that there are biologically plausible neural mechanisms underlying what happens on the levels of vectors and, most central to our concerns, the level of symbols. Here, we simply refer to and make use of the Neural Engineering Framework (Eliasmith and Anderson, 2003) and its Python implementation Nengo (Bekolay et al., 2014).

Schlegel et al. (2022) in their very useful survey of VSAs offer a comparison of different approaches in terms of four distinct parameters:

Hypervector selection: When selecting vectors to represent basic entities one aims to create maximally different encodings. Higher dimensional vector spaces offer sufficient space to maintain a large class of vectors distinct and moreover, they have the useful property that two random vectors are with very high probability quasi-orthogonal. A common strategy is to use a real range which is normally distributed with a mean of 0 and a variance of $1/D$ where D defines the number of dimensions.

Similarity measurement: VSAs use similarity metrics to evaluate vector representations, in particular, to assess whether the represented symbols have a related meaning. The similarity metric plays the essential role of selecting the correct denoised vector from the database and to ensure a robust operation of VSAs. The dot product of two vectors A, B is standardly computed as the sum of the product of their components, as in (1a). This is the basis for defining the cosine between two vectors as in (1b) in terms of the dot product and the vectors' lengths:

(1) a. $$A \cdot B = \sum_{k=0}^{D-1} a_k b_k$$

b. $$cos\ \theta = \frac{A \cdot B}{||A|| * ||B||}$$

Following most VSA approaches, we use cosine as a measure of similarity. Given (1b), this reduces to the dot product when the vectors are normalized (i.e., of length 1). If $A \cdot B \approx 1$, the vectors are (nearly) identical. For any vector A,

(2) $A \cdot A \approx 1$

Bundling: VSAs use a bundling operator to superimpose (or overlay) given hypervectors. Plate (1997) argues that a bundling operator must satisfy unstructured similarity preservation, namely $A + B$ is similar to A and to B and to any bundle $A + C$ that contains one of the vectors. Bundling is typically handled using vector addition, but in the approach adopted here this requires a normalization step to a vector length of one.

Binding: Binding $\times$ is used to connect two vectors, e.g., role-filler pairs. The output is again a vector from the same vector space. Plate (1997) argues that binding needs to satisfy:

- Non-similarity of bindees to output: $A \times B \not\approx A, B$
- Similarity preservation: $A \approx A'$, $B \approx B'$ implies $A \times B \approx A' \times B'$
- 'x' is invertible: if $C = A \times B$, there exists A^{-1} such that $C \times A^{-1} = B$

In the current paper we generally follow the approach known as Holographic Reduced Representations (HRR), first defined by Plate (1991), which is the approach utilized by Eliasmith and implemented in Nengo. However, as Eliasmith notes, one could make different choices if clear motivation for these arises. Specifically, with respect to binding we use circular convolution $C = A \circledast B$ defined as follows in a space of dimension D:

(3) **Circular convolution**

$$c_j = \sum_{k=0}^{D-1} b_k a_{j-k(modD)}$$
for $j \in \{0, \ldots, D-1\}$

Circular convolution approximates the standard tensor outer product by summing over all of its (wrap-around) diagonals. This operator is commutative as well as associative. Circular correlation provides an approximated inverse for circular convolution used for unbinding. The inverse is defined in (4a), exemplified in (4b), and its use for unbinding is given in (4c):[2]

(4) **Inverse for circular convolution**

a. $a_j^{-1} = a_{D-j(modD)}$
where $j \in \{0, \ldots, D-1\}$

b. In other words: $\langle a_0, a_1, \ldots, a_{D-1}\rangle^{-}1 = \langle a_0, a_{D-1}, \ldots, a_1\rangle$

c. $A \circledast B \circledast B^{-1} \approx A$

In what follows, we use B' for B^{-1}.

3 TTR

We give a brief sketch of those aspects of TTR which we will use in this paper. For more detailed accounts see Cooper (2023).

$s : T$ represents a judgement that s is of type T. Types may be either *basic* or *complex* (in the sense that they are structured objects which have types or other objects introduced in the theory as components). One basic type that we will use is *Ind*, the type of individuals; another is *Real*, the type of real numbers.

[2]In algebra an element A's multiplicative inverse A^{-1} is by definition an element such that $A \times A^{-1} = 1$ (the unit element of multiplication). An approximate inverse of an element A ApproxInv(A)$^{-1}$ is one where $A \times ApproxInv(A)^{-1} \approx 1$.

Among the complex types are *ptypes* which are constructed from a predicate and arguments of appropriate types as specified for the predicate. Examples are 'man(a)', 'see(a,b)' where a, b : *Ind*. The objects or *witnesses* of ptypes can be thought of as situations, states or events in the world which instantiate the type. Thus s : man(a) can be glossed as "s is a situation which shows (or proves) that a is a man".

Another kind of complex type is *record types*. In TTR *records* are modelled as a labelled set consisting of a finite set of fields. Each field is an ordered pair, $\langle \ell, o\rangle$, where ℓ is a *label* (drawn from a countably infinite stock of labels) and o is an object which is a witness of some type. No two fields of a record can contain the same label. Importantly, o can itself be a record.

A *record type* is like a record except that the fields are of the form $\langle \ell, T\rangle$ where ℓ is a label as before and T is a type. The basic intuition is that a record, r is a witness for a record type, T, just in case for each field, $\langle \ell_i, T_i\rangle$, in T there is a field, $\langle \ell_i, o_i\rangle$, in r where $o_i : T_i$. (Note that this allows for the record to have additional fields with labels not included in the fields of the record type.)

The types within fields in record types may *depend* on objects which can be found in the record which is being tested as a witness for the record type. We use a graphical display to represent both records and record types where each line represents a field. Example (5) represents the type of records which can be used to model situations where a man runs.

(5) $$\begin{bmatrix} \text{ref} & : & \textit{Ind} \\ \text{c}_{\text{man}} & : & \text{man(ref)} \\ \text{c}_{\text{run}} & : & \text{run(ref)} \end{bmatrix}$$

A record of this type would be of the form

(6) $$\begin{bmatrix} \text{ref} & = & a \\ \text{c}_{\text{man}} & = & s \\ \text{c}_{\text{run}} & = & e \\ \ldots & & \end{bmatrix}$$

where a : *Ind*, s : man(a) and e : run(a).

Some of our types will contain *manifest fields* like the c_{man}-field below:

(7) $$\begin{bmatrix} \text{ref} & : & \text{Ind} \\ \text{c}_{\text{man}=s_{23}} & : & \text{man(ref)} \end{bmatrix}$$

Here, $\left[\mathrm{c_{man}}{=}s_{23}\mathrm{:man(ref)}\right]$ is a convenient notation for $\left[\mathrm{c_{man}}\mathrm{:man(ref)}_{s_{23}}\right]$ where $\mathrm{man(ref)}_{s_{23}}$ is a *singleton type*. If $a : T$, then T_a is a singleton type and $b : T_a$ iff $b = a$.[3] Manifest fields allow us to progressively specify what values are required for the fields in a type.

It is possible to combine record types. Suppose that we have two record types C_1 and C_2:

(8) $$C_1 = \begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c_{man}} : \mathrm{man(x)} \end{bmatrix}$$
$$C_2 = \begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c_{run}} : \mathrm{run(x)} \end{bmatrix}$$

In this case, $C_1 \wedge C_2$ is a type; more specifically, a meet type. In general if T_1 and T_2 are types then $T_1 \wedge T_2$ is a type and $a : T_1 \wedge T_2$ iff $a : T_1$ and $a : T_2$. A meet type $T_1 \wedge T_2$ of two record types can be simplified to a new record type by a process similar to unification in feature-based systems. If T_1 and T_2 are record types then there will be a type $T_1 \dot{\wedge} T_2$ equivalent to $T_1 \wedge T_2$ (in the sense that something will be of the first type if and only if it is of the second type). The operation $\dot{\wedge}$ is referred to as merge.

(9) $$C_1 \dot{\wedge} C_2 = \begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c_{man}} : \mathrm{man(x)} \\ \mathrm{c_{run}} : \mathrm{run(x)} \end{bmatrix}$$

We will introduce further details of TTR as we need them in subsequent sections.

4 Relating SPA and TTR

4.1 The basic idea

We define a mapping, σ, from types in TTR to patterns (types) of neural activity represented as vectors in SPA[4]. On the basis of this we define neural judgement conditions of the form "agent A judges s to be of type T if a particular neural condition involving $\sigma(T)$ holds. The connective here is a conditional rather than a biconditional because we allow more than one pattern of neural activity to correspond to the same TTR judgement. For example, A may judge s to be of T because of, say visual perception, or because s has been stored in memory corresponding to the witness cache discussed in Cooper (2019b). This is in contrast to the proposal in Cooper (2019a) which defines a function from types to patterns (types) of neural activity but does not take the additional step of giving neural judgement conditions. The move from representing types to representing judgements, which belong to the theory of action defined on the theory of types, appears to us to be a conceptual improvement. Essentially, the correspondence we define characterizes the brain activity of an agent when engaged in an act of making a type judgement, rather than simply giving a neural representation of a type. This seems promising for building a theory of how an embodied agent perceives its environment rather than creating a neural representation of a type without specifying how it would link to the world.

Another way in which the approach taken here differs from that of Cooper (2019a) is that the approach to representing the structure of complex types relies on the vector operations used in SPA, such as circular convolution, rather than the phasing of neural activity as in Cooper (2019a) following in a tradition of neural modelling stemming from Shastri (1999). This raises a question of whether the modelling in terms of vector operations reveals enough structure which we will leave open in this paper.

Our aim in this paper is to begin mapping out a possible correspondence between TTR and SPA. We do not yet have a complete definition and there are a number of questions about what we have so far. Nevertheless, we hope that what we have represents a promising beginning. Below, we often use $T \sim \mathbf{T}$ to mean $\sigma(T) = \mathbf{T}$. We will also often use $\mathbf{T}$ to represent $\sigma(T)$

We will frequently let equality or near similarity between two patterns of neural activation in SPA terms characterize TTR neural judgement conditions. In doing this we will exploit the fact the the dot product of two (nearly) identical vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a} \cdot \mathbf{b}$ is approximately equal to 1 (see Eliasmith, 2013, p. 389).

4.2 Basic types

We will use semantic pointers to correspond to basic TTR types. For basic types, we assume a

[3] Cooper (2023) uses a modification of this characterization of singleton types: if a is of some type, then T_a is a singleton type. $b : T_a$ iff $b : T$ and $b = a$. This allows for there to be types T_a where $a \not: T$. Such types have no witnesses.

[4] In this paper, we are not concerned with the converse mapping, from SPA to TTR.

function β that provides a unique semantic pointer corresponding to each basic type and that the function σ is defined relative to β:

(10) If T is a basic type, $\sigma_\beta(T) = \beta(T)$

We will suppress the β-subscript on σ in what follows.

4.3 Judgements

In TTR, judgements involving basic perceptual types can be made either using a classifier or based on a witness cache (Larsson, 2020). Type judgements based on classifiers take real-valued (e.g. perceptual) inputs.

In SPA, as exemplified by the MNIST dataset (Deng, 2012) and perceptual/cortical modelling, a classifier can be implemented as a hierarchical statistical model, which constructs representations of the input, which in turn are mapped into mechanistic SPA models (Tang and Eliasmith, 2010). At the highest level of the hierarchy, we have compressed representation summarising what has been presented to the lowest level. Following Eliasmith (2013), this compressed representation is a semantic pointer.

To judge whether a situation s is of a (perceptual) type T, the perception of s by an agent A generates a representation (in the form of neural activity, e.g. on V1, the primary visual cortex) $\mathbf{s_A}$ (A's take on s in the terminology of Larsson, 2020). A hierarchical statistical model, call it κ, when fed $\mathbf{s_A}$ as input to the lowest level of κ (e.g. V1) produces a compressed representation (neural activity) $\kappa[\mathbf{s_A}]$ on the highest level (IT, the inferotemporal cortex) of κ—see Figure 1 for an illustration. The semantic pointer $\mathbf{T}$ specifies a certain type of activity on the highest level of κ, and if this activity is triggered by A perceiving s, this corresponds to A judging s to be of type T. If T is a perceptual basic type related to the statistical model κ, then the neural judgement condition can be expressed as (11a) or equivalently (11b) .

(11) a. $s :_A T$ if $\kappa[\mathbf{s_A}] \approx \mathbf{T}$

b. $s :_A T$ if $\kappa[\mathbf{s_A}] \cdot \mathbf{T} \approx 1$

Below we will often suppress the A-subscript on ':'.

Type judgements can also be based on a witness cache. The witness cache in TTR is a function F

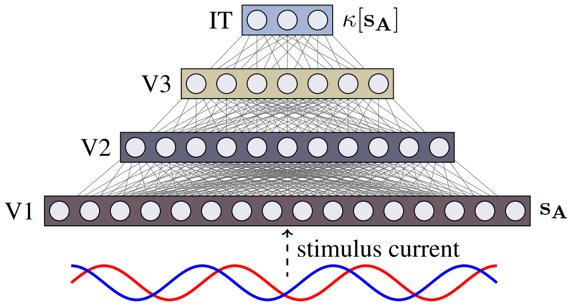


Figure 1: Illustration of hierarchical statistical model κ. To the left of each layer is the name of the layer, and to the right is the activity in that layer.

that takes a type T and returns a set of objects so that $x : T$ if $x \in F(T)$. We can let $\mathbf{F}$ be a structure that binds types with a bundling of semantic pointers $\mathbf{a}_0 + \mathbf{a}_1 + \ldots + a_n$, for example

(12) $\mathbf{F} = (\mathbf{Ind} \circledast (\mathbf{a} + \mathbf{b} + \ldots)) +$
$(\mathbf{Int} \circledast (\mathbf{1} + \mathbf{2} + \ldots)) + \ldots$

In SPA, a bundle is similar to any of its elements. However, this similarity is more approximate than similarity between near-identical vectors. For this reason, we do not require the dot product of bundle and element to be 1, but only that it does not approximate 0:

(13) $(\mathbf{A_1} + \mathbf{A_2} + \ldots + \mathbf{A_n}) \cdot \mathbf{A_i} \not\approx 0$,
$(1 \leq i \leq n)$

Given this, type checking can be done by looking up the witness cache in $\mathbf{F}$ and checking its similarity to the object:

(14) $x : T$ if $\mathbf{F} \circledast \mathbf{T}' \approxeq \mathbf{x}$

where we use $\approxeq$ so that this means

(15) $x : T$ if $\mathbf{F} \circledast \mathbf{T}' \cdot \mathbf{x} \not\approx 0$

(15) says that the vector which results from unbinding $\mathbf{T}$ associated with type T from $\mathbf{F}$ is (approximately) identical to the semantic pointer $\mathbf{a}$. For example:

(16) $a : \mathit{Ind}$ if $\mathbf{F} \circledast \mathbf{Ind}' \approxeq \mathbf{a}$

See Figure 2 for an example.[5]

[5]The code for this and the following examples can be found at `https://github.com/aluecking/SPA-TTR`.

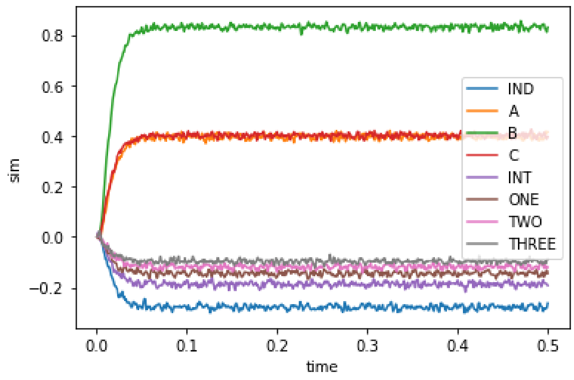


Figure 2: Given an $\mathbf{F}$ structure consisting of pointers for two basic types *IND* and *INT* bound to three object pointers each—*A*, *B*, *C*, respectively *ONE*, *TWO*, *THREE*—the (correct) result of unbinding $\mathbf{F}$ with $\mathbf{IND}'$ is approxiately (≈) similar to pointers $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.

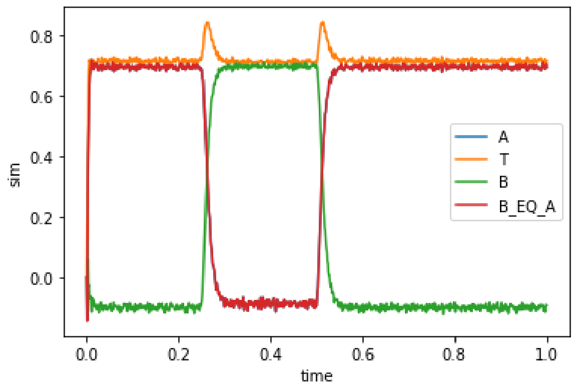


Figure 3: The similarity of T_a with b is only high if $b \approx a$. Comparing the similarity of $\mathbf{T}+\mathbf{a}$ ($t < 0.25\,\mathrm{s}$), $\mathbf{T}+\mathbf{b}$ ($0.25\,\mathrm{s} < t < 0.5\,\mathrm{s}$) and $\mathbf{T}+\mathbf{b}=\mathbf{a}$ (notated 'B_EQ_A'; $t > 0.5\,\mathrm{s}$) to all pointers in question (note that 'A' is masked by 'B_EQ_A').

4.4 Singleton types

A special case is typechecking for singleton types $T_a \sqsubseteq T$. We define the SPA structure to correspond to singleton types thus:

(17) $T_a \sim (\mathbf{T}+\mathbf{a})$

To check if $b : T_a$, we can check the equality $\mathbf{a} \approx \mathbf{b}$ and that $b : T$:

(18) $b : T_a$ if $\mathbf{a} \approx \mathbf{b}$ and $b : T$

—see Figure 3.

4.5 Labelled sets

Many structures in TTR are defined as labelled sets. We take labelled sets in TTR to correspond to SPA structures according to the following:

(19) $$\{\langle \ell_1, x_1 \rangle, \ldots, \langle \ell_n, s_n \rangle\} \sim \boldsymbol{\ell_1} \circledast \mathbf{d_1} + \ldots + \boldsymbol{\ell_n} \circledast \mathbf{x_n}$$

This move, however, involves treating labels as proper pointers, that is, compressed high(er) level semantic representations, which seems to be at odds with the status of labels as arbitrary book-keeping devices. A potential way for reconciliation is to think of labels as indicating functional roles, as is initially attested in fMRI studies on processing, where it has been found that general agency (e.g., *owl-as-agent*) is represented in different cortical regions than narrow agency (e.g., *owl-as-chaser*) (Frankland and Greene, 2020b). This is reminiscent of an inferential view of thematic roles (Dowty, 1991), which seem to justify a semantic pointer representation, but poses the question whether this approach extends to all labels.

Labelled sets are sets of ordered pairs where the first item in each pair is a label. In SPA-TTR, we are using the binding operator $\circledast$ to associate two SPA terms. In both frameworks, given an item x and structure associating items (in TTR, a set S of ordered pairs of items; in SPA, a vector $\mathbf{S}$ as shown above) it is possible to retrieve the item y which x is associated with in S. In TTR, this is done by finding a pair $\langle x, t\rangle$ in S. In SPA-TTR, this is done by unbinding y from a binding $x \circledast y$ in S.

An important difference between TTR and SPA is that in TTR, it is easy to retrieve the labels that are used in a record type, which then enables relabelling the record as needed. In SPA-TTR, retrieving the labels requires probing $\mathbf{S}$ for the presence of each of a (finite) set of labels. If the set of labels is large, this may be inefficient. We do not offer a full solution to this problem here, but leave it for future work. However, we believe that a solution can be to keep around an index of the labels used in different record types.

4.6 Record types

We will not attempt here to represent TTR records in SPA, but focus instead of record types. Since TTR record types are labelled sets where the labels are paired with types, we use our SPA coding of labelled sets for record types.

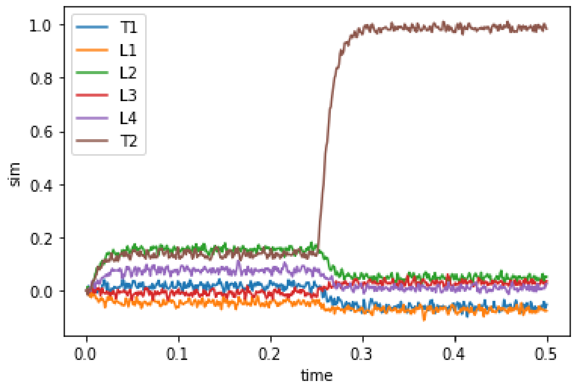


Figure 4: Recovering $\mathbf{T_2}$ from its path $\mathbf{T_1} \circledast \mathbf{L_1} \circledast \mathbf{L_2} \circledast \mathbf{L_3} \circledast \mathbf{L_4}$ is successful, but lossy as can be seen by comparison to querying $\mathbf{T_2}$ directly starting from 0.25 s.

(20) $$\begin{bmatrix} \ell_1 & : & T_1 \\ \dots & & \\ \ell_n & : & T_n \end{bmatrix} \sim$$
$$\ell_\mathbf{1} \circledast \mathbf{T_1} + \ldots + \ell_\mathbf{n} \circledast \mathbf{T_n}$$

4.7 Paths in record types

In TTR, labels coinjoined by '.' form paths in records and record types. We can use unbinding in SPA to achieve something similar. If T_1 is a record type and T_2 is a type and $T_1.\ell_1.\ldots.\ell_m : T_2$ and $T_1 \sim \mathbf{P_1}$, $T_2 \sim \mathbf{P_2}$, $\ell_i \sim \mathbf{L_i}(1 \leq i \leq m)$ then

(21) $\mathbf{P_1} \circledast \mathbf{L'_1} \circledast \ldots \circledast \mathbf{L'_m} \approx \mathbf{P_2}$

We can recover **P2** (i.e., type T_2) from $\mathbf{P_1}$ by following the path $\mathbf{L'_1} \circledast \ldots \circledast \mathbf{L'_m}$, that is, by unbinding it with all the pointers used to construct it. Note that this retrieval is lossy, as illustrated in terms of a path consisting of four labels in Figure 4.

4.8 Meet and Merge

We take both the meet type $T_1 \wedge T_2$ of two types T_1 and T_2 and the merge $T_1 \underset{\cdot}{\wedge} T_2$ of two record types T_1 and T_2 to correspond to the SPA summing operation $+$.

(22) a. $T_1 \wedge T_2 \sim \mathbf{T_1} + \mathbf{T_2}$ for types T_1 and T_2

b. $T_1 \underset{\cdot}{\wedge} T_2 \sim \mathbf{T_1} + \mathbf{T_2}$ for record types T_1 and T_2

c. $\sigma(T_1 \wedge T_2) = \sigma(T_1 \underset{\cdot}{\wedge} T_2) = \mathbf{T_1} + \mathbf{T_2}$

The SPA summing operation is distributive in the same way that $\underset{\cdot}{\wedge}$ is—'binding distributes over bundling' (Schlegel et al., 2022, p. 4536)[6]—, so that

(23) $(\ell_\mathbf{1} \circledast \mathbf{T_1} + \ell_\mathbf{1} \circledast \mathbf{T_2} = (\ell_\mathbf{1} \circledast (\mathbf{T_1} + \mathbf{T_2}))$

corresponding to

(24) $[\ell_1{:}T_1] \underset{\cdot}{\wedge} [\ell_1{:}T_2] = [\ell_1{:}T_1 \underset{\cdot}{\wedge} T_2]$

Conflating $\wedge$ and $\underset{\cdot}{\wedge}$ means we are not making a distinction between $T_1 \wedge T_2$ and $T_1 \underset{\cdot}{\wedge} T_2$ for record types T_1, T_2 (for non-record types, they work in the same way also in TTR.).

4.9 Ptypes

Cooper (2023) defines a ptype $P(a_1, \ldots, a_n)$ as representing a labelled set $\{\langle \text{pred}, P\rangle, \langle \text{arg}_1, a_1\rangle, \ldots \langle \text{arg}_1, a_n\rangle\}$. We follow this, so that e.g.

(25) a. run(a) $\sim$ $(\mathbf{pred} \circledast \mathbf{run} + \mathbf{arg1} \circledast \mathbf{a})$

b. hug(a,b) $\sim$ $(\mathbf{pred} \circledast \mathbf{hug} + \mathbf{arg1} \circledast \mathbf{a} + \mathbf{arg2} \circledast \mathbf{b})$

An important area for future research is to enable classifier-based judgements of sensory input as being of ptypes and record types involving ptypes. For example, given a situation s where a boy hugs a dog, we want an agent A's take on s to be judged to be of a complex type involving properties and relations.

4.10 Subtyping

Since subtyping can be defined in terms of a TTR equality between two types, this could appear to be a means of formulating the corresponding SPA-TTR definition:

(26) a. $T_1 \sqsubseteq T_2$ if $T_1 \underset{\cdot}{\wedge} T_2 = T_1 \sim (\mathbf{T_1} + \mathbf{T_2}) \approx \mathbf{T_1}$

b. $\sigma(T_1 \sqsubseteq T_2) = (\mathbf{T_1} + \mathbf{T_2}) \approx \mathbf{T_1}$

For example,

[6] In fact, in Nengo the vocabulary parses of, e.g., 'A * B + A * C' and 'A * (B + C)' result in the same vector.

(27) $\sigma(\begin{bmatrix}\text{x:a}\\\text{y:b}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{x:a}\end{bmatrix}) = ((\mathbf{x}\circledast\mathbf{a}+\mathbf{y}\circledast\mathbf{b})+(\mathbf{x}\circledast\mathbf{a})) \approx (\mathbf{x}\circledast\mathbf{a}+\mathbf{y}\circledast\mathbf{b})$

However, the above solution does not work because (27) holds only if $\mathbf{T_1} = \mathbf{T_2}$, which is of course a much stronger requirement than subtyping. An alternative could be to apply an element-wise maximum function:

(28) a. $T_1 \sqsubseteq T_2$ iff $T_1 \dot{\wedge} T_2 = T_1 \sim \max(\mathbf{T_1}, \mathbf{T_2}) \approx \mathbf{T_1}$

b. $\sigma(T_1 \sqsubseteq T_2) = \max(\mathbf{T_1}, \mathbf{T_2}) \approx \mathbf{T_1}$

The similarity of the maximum is indeed larger than the (cosine) similarity of supertype and subtype (see `https://github.com/aluecking/SPA-TTR`). However, further work is needed to further specify and verify this proposal.

4.11 Functions

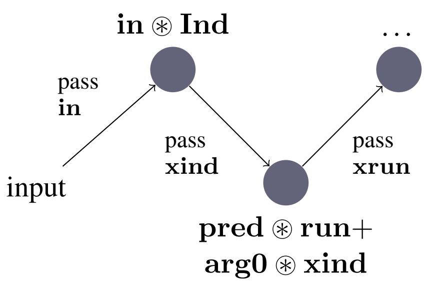
TTR functions can be represented as labelled sets, but doing so says nothing about how they are applied to arguments. For this reason, we will here be focusing on TTR functions as lambda abstracted expressions. We will not offer a complete account of TTR functions in SPA here, but only offer some initial remarks.

For instance, assume we have a function

(29) $\lambda r{:}\begin{bmatrix}\text{x : Ind}\end{bmatrix} \cdot \begin{bmatrix}\text{c} : \text{run}(r.\text{x})\end{bmatrix}$

This function corresponds to the following mini-network:

(30)



Or in SPA syntax:

```
d = 64 # use vectors of 64 dimensions
xind = spa.State(vocab=d)
xrun = spa.State(vocab=d)

input * spa.sym("IND") >> xind
xind * spa.sym("ARG0") + spa.sym("PRED *
    RUN")) >> xrun
```

where 'input' can, for instance, receive activation from another network such as κ (see (11b)) or sequentially range over (any subset of) the objects bound to **IND** in the witness cache (see (12)):

```
def inputs(t):
    if t < 0.25:
        return "A"
    elif t < 0.5:
        return "B"
    ...

input = spa.Transcode(inputs,
    output_vocab=d)
```

5 Summary and conclusions

In this paper, we took initial steps towards relating TTR to SPA, with mostly encouraging results. We accounted for basic types, perceptual and cache-based judgements, singleton types, record types, meet types and merging of record types, ptypes, and subtyping. As indicated above, more work is needed to account for subtyping and judgements involving ptypes. Work is ongoing to cover more aspects of TTR in SPA, including records and functions. In addition to these, several TTR elements remain to be covered, including join types, asymmetric merge, and type stratification to name but a few.

The benefit of succeeding with this effort would be a true hybrid between formal and neural semantics that could potentially have the benefits of both but the drawbacks of neither. We also hope that this work may throw light on many puzzling issues regarding the relation between formal and neural semantics.

Acknowledgements

The work of the first and second author was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith. 2014. Nengo: a Python tool for building large-scale functional brain models. Frontiers in Neuroinformatics, 7.

Anne Bezuidenhout. 2006. VP-ellipsis and the case for representationalism in semantics. ProtoSociology, 22:140–168. Compositionality, Concepts and Representations II.

Ellen Breitholtz. 2020. Enthymemes and Topoi in Dialogue: the use of common sense reasoning in conversation. Brill.

Robin Cooper. 2005. Austinian truth, attitudes and type theory. Research on Language and Computation, 3(4):333–362.

Robin Cooper. 2011. Copredication, quantification and frames. In Logical Aspects of Computational Linguistics (LACL 2011). Springer.

Robin Cooper. 2013. Clarification and generalized quantifiers. Dialogue and Discourse, 4:1–25.

Robin Cooper. 2019a. Representing types as neural events. Journal of Logic, Language and Information, 28:131–155.

Robin Cooper. 2019b. Types as Learnable Cognitive Resources in PyTTR. In Cleo Condoravdi and Tracy Holloway King, editors, Tokens of Meaning: Papers in Honor of Lauri Karttunen. CSLI Publications.

Robin Cooper. 2023. From Perception to Communication. Number 16 in Oxford Studies in Semantics and Pragmatics. Oxford University Press, Oxford, UK.

Robin Cooper and Jonathan Ginzburg. 2012. Negative inquisitiveness and alternatives-based negation. In Logic, Language and Meaning, pages 32–41. Springer.

Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6):141–142.

Simon Dobnik and Robin Cooper. 2017. Interfacing language, spatial perception and cognition in type theory with records. Journal of Language Modelling, 5(2):273–301.

David Dowty. 1991. Thematic proto-roles and argument selection. Language, 67(3):547–619.

Chris Eliasmith. 2013. How to Build a Brain: A Neural Architecture for Biological Cognition. Oxford University Press, Oxford.

Chris Eliasmith and Charles H. Anderson. 2003. Neural Engineering. Computational Neuroscience. MIT Press, Cambridge, MA.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2007. Classifying ellipsis in dialogue: A machine learning approach. Computational Linguistics, 33(3):397–427.

Steven M. Frankland and Joshua D. Greene. 2020a. Concepts and compositionality: In search of the brain's language of thought. Annual Review of Psychology, 71(1):273–303. PMID: 31550985.

Steven M. Frankland and Joshua D. Greene. 2020b. Two ways to build a thought: Distinct forms of compositional semantic representation across brain regions. Cerebral Cortex, 30(6):3838–3855.

Ross W. Gayler. 2003. Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. pages 133–138. Slightly updated 2004 in paper cs/0412059 on arXiv.

Jonathan Ginzburg. 2012. The interactive stance. Oxford University Press.

Jonathan Ginzburg, Raquel Fernández, and David Schlangen. 2014. Disfluencies as intra-utterance dialogue moves. Semantics and Pragmatics, 7(9):1–64.

Tomás Goucha, Emiliano Zaccarella, and Angela D. Friederici. 2017. A revival of *Homo loquens* as a builder of labeled structures: Neurocognitive considerations. Neuroscience & Biobehavioral Reviews, 81:213–224. The Biology of Language.

Ray Jackendoff. 2002. Foundations of Language. Oxford University Press, Oxford, UK.

Staffan Larsson. 2015. Formal semantics for perceptual classification. Journal of Logic and Computation, 25(2):335–369. Published online 2013-12-18.

Staffan Larsson. 2020. Discrete and probabilistic classifier-based semantics. In Proceedings of the Probability and Meaning Conference (PaM 2020), pages 62–68.

Staffan Larsson. 2021. The role of definitions in coordinating on perceptual meanings. In Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2021).

Andy Lücking. 2016. Modeling co-verbal gesture perception in type theory with records. In Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, pages 383–392.

Andy Lücking and Jonathan Ginzburg. 2022. Referential transparency as the proper treatment for quantification. Semantics and Pragmatics, 15(4):1–56.

Andy Lücking and Jonathan Ginzburg. 2023. Leading voices: Dialogue semantics, cognitive science, and the polyphonic structure of multimodal interaction. Language and Cognition, 15(1):148–172.

Tony Plate. 1991. Holographic reduced representations: Convolution algebra for compositional distributed representations. In Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI'91, pages 30–35.

Tony Plate. 1997. A common framework for distributed representation schemes for compositional structure. Connectionist systems for knowledge representation and deduction, pages 15–34.

M. Purver. 2006. CLARIE: Handling clarification requests in a dialogue system. Research on Language & Computation, 4(2):259–288.

Kenny Schlegel, Peer Neubert, and Peter Protzel. 2022. A comparison of vector symbolic architectures. Artificial Intelligence Review, 55(6):4523–4555.

Lokendra Shastri. 1999. Advances in SHRUTI – a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. Applied Intelligence, 11(1):79–108.

Yichuan Tang and Chris Eliasmith. 2010. Deep networks for robust visual recognition. In Proceedings of the 27 th International Conference on Machine Learning, pages 1055–1062.

Guangyu Robert Yang and Xiao-Jing Wang. 2020. Artificial neural networks for neuroscientists: A primer. Neuron, 107(6):1048–1070.