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

Staffan Larsson[1] Robin Cooper[1] Jonathan Ginzburg[2] Andy Lücking[2,3]

[1] University of Gothenburg
[2] Université Paris Cité
[3] Goethe University Frankfurt

CLASP seminar 15/11 2023
Previously presented at NALOMA IV, IWCS 2023, Nancy

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR
- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Introduction I

- Questions
  - How is linguistic meaning related to perception?
  - How do we learn and agree on the meanings of our words?
- We are developing a formal *judgement-based semantics* where notions such as perception, classification, judgement, learning and dialogue coordination play a central role
- Key ideas:
  - modelling perceptual meanings as classifiers of real-valued perceptual data
  - modelling how agents learn and coordinate on meanings through interaction with other agents (semantic coordination)

Introduction II

- We formulate our account in TTR, a theory of types with records (Cooper, 2023)
- Several key aspects of semantic coordination and learning from interaction have been formalised using TTR.
- It is, at the same time, widely acknowledged that neural models have many attractive properties, including robustness against noise and (potentially) biological plausibility.

Introduction III

- Work on TTR 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.
- Here, we will explore the possibility of using Eliasmith's Semantic Pointer Architecture (SPA) (Eliasmith, 2013) for this purpose
- Our hope is that this will ultimately lead to an account of human language learning that is interactive, incremental, explainable, robust and biologically plausible

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR
- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

SPA

The Semantic Pointer Architecture (SPA) is

- a Vector Semantic Architecture (VSA) (Schlegel *et al.*, 2022)
- based on the Neural Engineering Framework (NEF) (Eliasmith and Anderson, 2003)
- implemented as *Spaun* (Eliasmith *et al.*, 2012) in *Nengo* (Bekolay *et al.*, 2014)

Semantic Pointers I

[. . .] 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)

- Hence, there are three *perspectives* on or *levels of description* for semantic pointers:
  (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.

Elements of VSAs and SPA

Schlegel *et. al.* (2022) in their survey of Vector Semantic Architectures (VSAs) offer a comparison of different approaches in terms of four distinct parameters:

- Hypervectors
- Similarity measurement
- Binding
- Bundling

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
- They have the useful property that two random vectors are with very high probability *quasi-orthogonal*.

Similarity measurement I

- VSAs use similarity metrics to evaluate vector representations, in particular, to assess whether the represented symbols have a related meaning.
- The *dot product* of two vectors A, B is standardly computed as the sum of the product of their components, as in (1a).
- Following most VSA approaches, we use cosine as a measure of similarity; given (1b), this reduces to the dot product when the vectors are normalised (i.e., of length 1).

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

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

Similarity measurement II

- If $A \cdot B \approx 1$, the vectors are (nearly) identical, $A \approx B$
- For any vector A, $A \cdot A \approx 1$

Bundling

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

Binding I

- 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 $C \times A^{-1} = B$

Binding II

- Here, we generally follow the approach known as *Holographic Reduced Representations* (HRR), first defined by Plate (1991), which is the approach utilised by Eliasmith and implemented in Nengo.
- Specifically, with respect to binding we use *circular convolution* $C = A \circledast B$ defined as follows in a space of dimension D:

$$(2) \quad c_j = \sum_{k=0}^{D-1} b_k a_{j-k (\mathrm{mod}\ D)} \text{ for } j \in \{0, \ldots, D-1\}$$

- Circular convolution approximates the standard tensor outer product by summing over all of its (wrap-around) diagonals.

Binding III

- Circular *correlation* provides an approximated inverse for circular convolution used for *unbinding*.
- The inverse is defined in (3a), exemplified in (3b), and its use for *unbinding* is given in (3c):

(3) a. $a_j^{-1} = a_{D-j (\text{mod } D)}$ 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. For example, $A \circledast B \circledast B^{-1} \approx A$

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

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR
- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

TTR fundamentals I

- $a : T$ is a judgement that a is of type T
- Types may be either *basic* or *complex*
- Some basic types in TTR:
  - *Ind*, the type of an individual
  - *Int*, the type of integers
  - *Real*, the type of real numbers

TTR fundamentals II

- Complex types are structured objects which have types or other objects introduced in the theory as components
- *ptypes* are constructed from a predicate and arguments of appropriate types as specified for the predicate.
- Examples are 'man(a)', 'see(a,b)' where $a, b : \mathit{Ind}$.
- The objects or *witnesses* of ptypes can be thought of as proofs in the form of situations, states or events in the world which instantiate the type.

Records and record types

- If
  - $a_1 : T_1$,
  - $a_2 : T_2(a_1)$,
  - $\ldots$,
  - $a_n : T_n(a_1, a_2, \ldots, a_{n-1})$,
  - where $T(a_1, \ldots, a_n)$ represents a type T which depends on the objects $a_1, \ldots, a_n$,
- ...the record to the left is of the record type to the right.

$$\begin{bmatrix} \ell_1 & = & a_1 \\ \ell_2 & = & a_2 \\ \ldots & & \\ \ell_n & = & a_n \\ \ldots & & \end{bmatrix} : \begin{bmatrix} \ell_1 & : & T_1 \\ \ell_2 & : & T_2(l_1) \\ \ldots & & \\ \ell_n & : & T_n(\ell_1, l_2, \ldots, l_{n-1}) \end{bmatrix}$$

- $\ell_1, \ldots \ell_n$ are *labels* which can be used elsewhere to refer to the values associated with them.

Records and record types

- A sample record and record type:

$$\begin{bmatrix} \text{ref} & = & \text{obj}_{123} \\ \text{c}_{\text{man}} & = & \text{prf1} \\ \text{c}_{\text{run}} & = & \text{prf2} \end{bmatrix} : \begin{bmatrix} \text{ref} & : & \mathit{Ind} \\ \text{c}_{\text{man}} & : & \text{man(ref)} \\ \text{c}_{\text{run}} & : & \text{run(ref)} \end{bmatrix}$$

- The record on the left is of the record type on the right provided
  - $\text{obj}_{123} : \mathit{Ind}$
  - $\text{prf1} : \text{man}(\text{obj}_{123})$
  - $\text{prf2} : \text{run}(\text{obj}_{123})$

Meet and merge I

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

(4) $$C_1 = \begin{bmatrix} \text{x}:\mathit{Ind} \\ \text{c}_{\text{man}}:\text{man(x)} \end{bmatrix}$$
$$C_2 = \begin{bmatrix} \text{x}:\mathit{Ind} \\ \text{c}_{\text{run}}:\text{run(x)} \end{bmatrix}$$

- 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 $a : T_2$.

Meet and merge II

- 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*.

$$(5) \quad C_1 \dot{\wedge} C_2 = \begin{bmatrix} \mathrm{x}:\mathit{Ind} \\ \mathrm{c_{man}}:\mathrm{man(x)} \\ \mathrm{c_{run}}:\mathrm{run(x)} \end{bmatrix}$$

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR
- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Relating SPA and TTR: The basic idea I

- We define a mapping, σ, from types in TTR to patterns (types) of neural activity represented as vectors in SPA.
- 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.
- Essentially, the correspondence we define characterises the brain activity of an agent when engaged in an act of making a type judgement

Relating SPA and TTR: The basic idea II

- Our aim 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 $\mathbf{T}$ to represent $\sigma(T)$.
- We will also often use $T \sim \mathbf{T}$ to mean $\sigma(T) = \mathbf{T}$.

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Basic types

- We will use semantic pointers to correspond to basic TTR types.
- For basic types, we assume a function β that provides a unique semantic pointer corresponding to each basic type and that the function σ is defined relative to β:

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

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

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

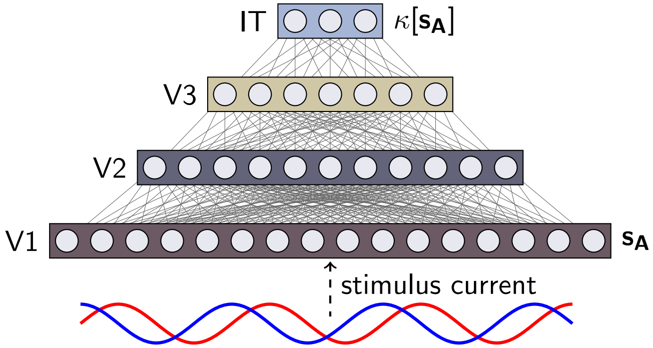
Judgements I

- 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
- 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.

Judgements II

- 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

Judgements III



Judgements IV

- 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 (7a) or equivalently (7b).

(7) 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 ':'.

Judgements V

- Type judgements can also be based on a witness cache.
- The witness cache in TTR is a function F that takes a type T and returns a set of objects so that $x : T$ if $a \in F(T)$.
- We can let **F** be a structure that binds types with a bundling of semantic pointers $\mathbf{a}_0 + \mathbf{a}_1 + \ldots + a_n$, for example

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

Judgements VI

- 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:

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

Judgements VII

- Given this, type checking can be done by looking up the witness cache in **F** and checking its similarity to the object:

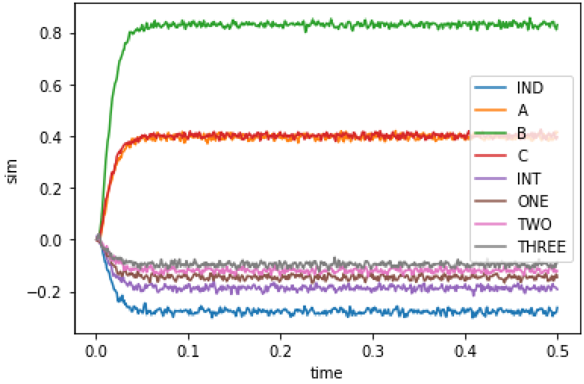
(10) $x : T$ if $\mathbf{F} \circledast \mathbf{T}' \approx \mathbf{x}$

where we use $\approx$ so that this means

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

- (11) says that the vector which results from unbinding **T** associated with type T from **F** is (approximately) identical to the semantic pointer **a**.
- For example, $a : \mathit{Ind}$ if $\mathbf{F} \circledast \mathbf{Ind}' \approx \mathbf{a}$

Judgements VIII



- Given an **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 **F** with **IND**′ is approximately (≊) similar to pointers **A**, **B** and **C**.

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Labelled sets I

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

(12)

$$\{\langle \ell_1, x_1 \rangle, \ldots, \langle \ell_n, x_n \rangle\} \sim \boldsymbol{\ell_1} \circledast \mathbf{x_1} + \ldots + \boldsymbol{\ell_n} \circledast \mathbf{x_n}$$

- 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.

Labelled sets II

- In both frameworks, given an item x and structure associating items, it is possible to retrieve the item y which x is associated with in the structure.
  - In TTR, this is done by finding a pair $\langle x, y \rangle$ in S, a set of ordered pairs of items
  - In SPA, this is done by unbinding y from a binding $x \circledast y$ in a vector $\mathbf{S}$, a bundle of bindings between pairs of items.

Labelled sets III

- 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 **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.

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Record types I

- 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.

$$(13) \quad \begin{bmatrix} \ell_1 & : & T_1 \\ \ldots & & \\ \ell_n & : & T_n \end{bmatrix} \sim \boldsymbol{\ell_1} \circledast \mathbf{T_1} + \ldots + \boldsymbol{\ell_n} \circledast \mathbf{T_n}$$

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Paths in record types I

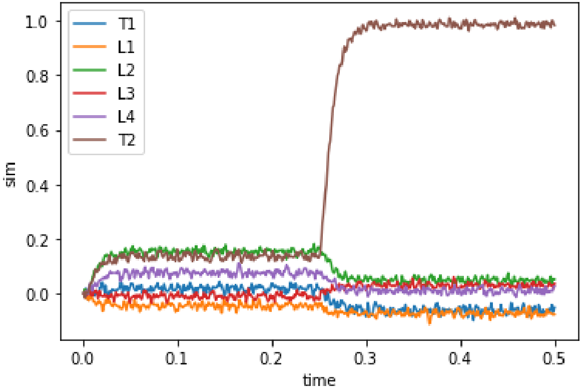
- In TTR, labels conjoined by '.' form paths in records and record types.
- We can use unbinding in SPA to achieve something similar.
- If
  - T_1 is a record type
  - T_2 is a type
  - $T_1.\ell_1.\ldots.\ell_m : T_2$
  - $T_1 \sim \mathbf{T_1},\ T_2 \sim \mathbf{T_2},\ \ell_i \sim \mathbf{L_i}(1 \leq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

(14) $\mathbf{T_1} \circledast \mathbf{L_1'} \circledast \ldots \circledast \mathbf{L_m'} \approx \mathbf{T_2}$

- We can recover $\mathbf{T_2}$ from $\mathbf{T_1}$ 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.

Paths in record types II

- Note that this retrieval is lossy:



- 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.

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Ptypes I

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

(15) a. run(a) $\sim$ (**pred** $\circledast$ **run** $+$ **arg1** $\circledast$ **a**)

b. hug(a,b) $\sim$ (**pred** $\circledast$ **hug** $+$ **arg1** $\circledast$ **a** $+$ **arg2** $\circledast$ **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 a situation s_A to be judged to be of a complex type involving properties and relations.

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Meet and Merge I

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

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

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

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

Meet and Merge II

- The SPA summing operation is distributive in the same way that $\dot{\wedge}$ is—'binding distributes over bundling' (Schlegel *et al.*, 2022)–, so that

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

corresponding to

(18) $\left[\ell_1 {:} T_1\right] \dot{\wedge} \left[\ell_1 {:} T_2\right] = \left[\ell_1 {:} T_1 \dot{\wedge} T_2\right]$

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

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR

- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Subtyping I

- 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:

(19) $T_1 \sqsubseteq T_2$ iff $T_1 \dot{\wedge} T_2 = T_1 \sim (\mathbf{T_1} + \mathbf{T_2}) \approx \mathbf{T_1}$

- For example,

(20) $$\sigma(\begin{bmatrix}\text{x}: T_a \\ \text{y}: T_b\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{x}: T_a\end{bmatrix}) = ((\mathbf{x} \circledast \mathbf{T_a} + \mathbf{y} \circledast \mathbf{T_b}) + (\mathbf{x} \circledast \mathbf{T_a})) \approx (\mathbf{x} \circledast \mathbf{T_a} + \mathbf{y} \circledast \mathbf{T_b})$$

Subtyping II

- However, the above solution does not work because (20) 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:

$$T_1 \sqsubseteq T_2 \text{ iff } T_1 \dot{\wedge} T_2 = T_1 \sim \max(\mathbf{T_1}, \mathbf{T_2}) \approx \mathbf{T_1} \tag{21}$$

- The similarity of the maximum is indeed larger than the (cosine) similarity of supertype and subtype.
- However, further work is needed to further specify and verify this proposal.

Outline

Introduction

SPA fundamentals

TTR fundamentals

Relating SPA and TTR
- The basic idea
- Basic types
- Judgements
- Labelled sets
- Record types
- Paths in record types
- Ptypes
- Meet and Merge
- Subtyping

Summary and conclusions

Summary and conclusions I

- 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.

Summary and conclusions II

- 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.

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

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

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

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

Eliasmith, Chris; Stewart, Terrence C; Choo, Xuan; Bekolay, Trevor; DeWolf, Travis; Tang, Yichuan; and Rasmussen, Daniel 2012.
A large-scale model of the functioning brain.
science 338(6111):1202–1205.

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

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

Plate, Tony 1991.
Holographic reduced representations: Convolution algebra for compositional distributed representations.

In Mylopoulos, John and Reiter, Ray, editors 1991, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, IJCAI'91.

30–35.

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

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