Ontology as manifold: towards symbolic and numerical artificial embedding

Chloé Mercier, Frédéric Alexandre, Thierry Viéville

Ontology as manifold:

towards symbolic and numeric artificial embedding

Chloé Mercier, Frédéric Alexandre, Thierry Viéville

How to model human cognition with regard to...

Psychologically inspired symbolic reasoning

Biologically plausible spiking neural networks

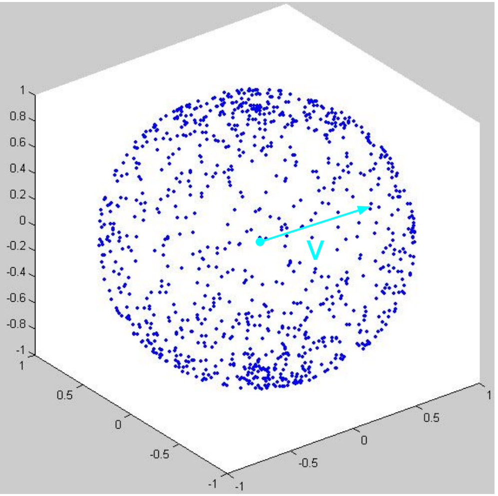
We propose a geometric mapping of ontologies onto neuronal manifolds

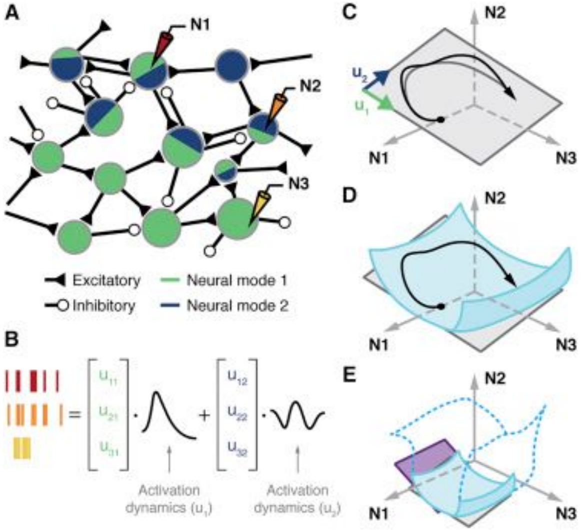
RDF(S): Resource Description Framework (Schema)
OWL: Web Ontology Language

VSA: Vector Symbolic Architecture
SPA: Semantic Pointer Architecture
NEF: Neural Engineering Framework

Vector Symbolic Architecture

- Each symbol encoded as a vector sampled from the unit hypersphere
- High-dimensional vector space => random vectors approximately orthogonal





The neural manifold hypothesis (Gallego et al, 2017)

Vector Symbolic Architecture

- Each symbol encoded as a vector sampled from the unit hypersphere
- High-dimensional vector space => random vectors approximately orthogonal
- **Dot product** as a **similarity measure**:

compares the semantic meaning of 2 vectors.

Vector Symbolic Architecture

- Each symbol encoded as a vector sampled from the unit hypersphere
- High-dimensional vector space => random vectors approximately orthogonal
- **Dot product** as a **similarity measure**:

compares the semantic meaning of 2 vectors.

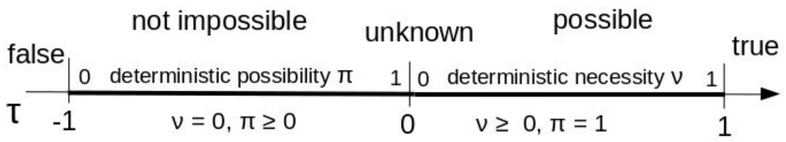
Closed-world reasoning	Open-world reasoning
Anything that cannot be stated true is false	Anything might be true, unless proven false
$\tau \in \{0,1\}$	$\tau \in \{-1, 0, 1\}$

Vector Symbolic Architecture

- Each symbol encoded as a vector sampled from the unit hypersphere
- High-dimensional vector space => random vectors approximately orthogonal
- **Dot product** as a **similarity measure**:

compares the semantic meaning of 2 vectors.

Closed-world reasoning	Open-world reasoning	Possibility theory
Anything that cannot be stated true is false	Anything might be true, unless proven false	Anything has a certain degree of possibility and necessity
$\tau \in \{0,1\}$	$\tau \in \{-1, 0, 1\}$	$\tau \in [-1,1]$



Structuring the universe of discourse

- symbols: ***resources***
  - classes (concepts)
  - individuals (instances)
  - properties (roles)
- facts: (weighted) ***triples***

(subject, predicate, object), τ

Structuring the universe of discourse

- symbols: ***resources***
  - classes (concepts)
  - individuals (instances)
  - properties (roles)
- facts: (weighted) ***triples***

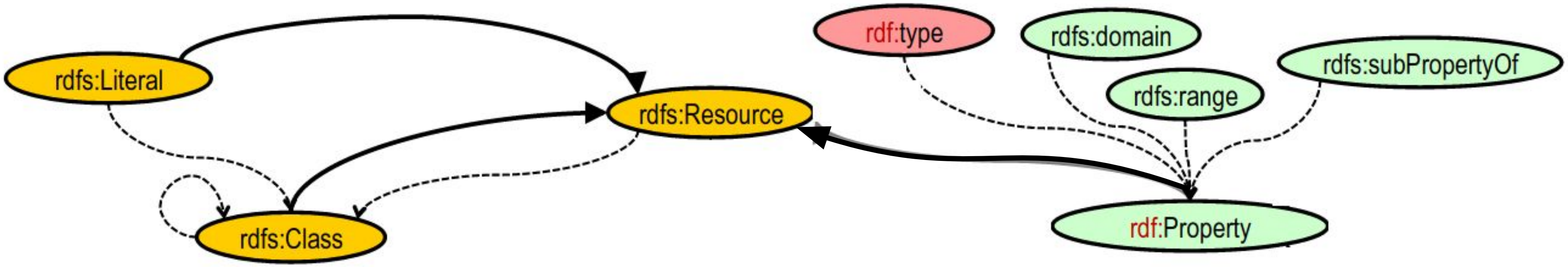
(subject, predicate, object), τ

vectors

s p o

scalar

RDF/RDFS: Resource Description Framework (Schema)



- **RDF: Resource Description Framework**
  - ➞ syntax: (subject, predicate, object) triples
- **RDFS: RDF Schema**
  - ➞ metamodel of RDF, written in RDF
  - ➞ allows to write (lightweight) **ontologies** and draw **inferences**

A triplestore using associative memories

A Semantic Pointer triplestore

(subject, predicate, object), τ

vectors: s p o

scalar

key	value
s	$\to$ τ $\mathcal{B}$(o,p)

A Semantic Pointer triplestore

(subject$_1$, predicate$_1$, object$_1$), τ_1

key	value
s_1 $\rightarrow$	τ_1 $\mathcal{B}(o_1,p_1)$

A Semantic Pointer triplestore

(subject$_1$, predicate$_1$, object$_1$), τ_1

(subject$_2$, predicate$_2$, object$_2$), τ_2

key	value
s_1 →	$\tau_1 \mathcal{B}(o_1,p_1)$
s_2 →	$\tau_2 \mathcal{B}(o_2,p_2)$

A Semantic Pointer triplestore

(subject$_1$, predicate$_1$, object$_1$), τ_1

(subject$_2$, predicate$_2$, object$_2$), τ_2

(subject$_2$, predicate$_3$, object$_3$), τ_3

key	value
s_1	$\rightarrow \tau_1\ \mathcal{B}(o_1,p_1)$
s_2	$\rightarrow \tau_2\ \mathcal{B}(o_2,p_2) + \tau_3\ \mathcal{B}(o_3,p_3)$

Inria Mnemosyne

A Semantic Pointer triplestore

(subject$_1$, predicate$_1$, object$_1$), τ_1

(subject$_2$, predicate$_2$, object$_2$), τ_2

(subject$_2$, predicate$_3$, object$_3$), τ_3

key	value
s_1	$\rightarrow \tau_1\, \mathcal{B}(o_1,p_1)$
s_2	$\rightarrow \tau_2\, \mathcal{B}(o_2,p_2) + \tau_3\, \mathcal{B}(o_3,p_3)$

Need to introduce:

- scalar multiplication a = τ a
- vector superposition c = a + b
- vector binding c = $\mathcal{B}$(a,b)

⇨ Semantic Pointer Architecture
(Eliasmith et al, 2013)

A Semantic Pointer triplestore

(subject$_1$, predicate$_1$, object$_1$), τ_1

(subject$_2$, predicate$_2$, object$_2$), τ_2

(subject$_2$, predicate$_3$, object$_3$), τ_3

key	value
s_1	$\rightarrow \tau_1\, \mathcal{B}(o_1,p_1)$
s_2	$\rightarrow \tau_2\, \mathcal{B}(o_2,p_2) + \tau_3\, \mathcal{B}(o_3,p_3)$

Need to introduce:
- scalar multiplication a = τ a
- vector superposition c = a + b
- vector binding c = $\mathcal{B}$(a,b)

⇨ Semantic Pointer Architecture
(Eliasmith et al, 2013)

Vector-derived Transformation Binding

(Gosmann and Eliasmith, 2019)

- Square dimensionality d with d'² = d
- Binding operation defined as:

$$\mathcal{B}(\mathbf{x},\mathbf{y}) = \mathbf{B}_\mathbf{y}\mathbf{x} = \begin{bmatrix} \mathbf{B}'_\mathbf{y} & 0 & 0 \\ 0 & \mathbf{B}'_\mathbf{y} & 0 \\ 0 & 0 & \mathbf{B}'_\mathbf{y} \end{bmatrix} \mathbf{x} \quad \text{where} \quad \mathbf{B}'_\mathbf{y} = d^{\frac{1}{4}} \begin{bmatrix} y_1 & y_2 & \cdots & y_{d'} \\ y_{d'+1} & y_{d'+2} & \cdots & y_{2d'} \\ \vdots & \vdots & \ddots & \vdots \\ y_{d-d'+1} & y_{d-d'+2} & \cdots & y_d \end{bmatrix}$$

- Non commutative, non associative
- Distributive and bilinear
- Admits right identity and right inverse

A Semantic Pointer triplestore

(subject$_1$, predicate$_1$, object$_1$), τ_1

(subject$_2$, predicate$_2$, object$_2$), τ_2

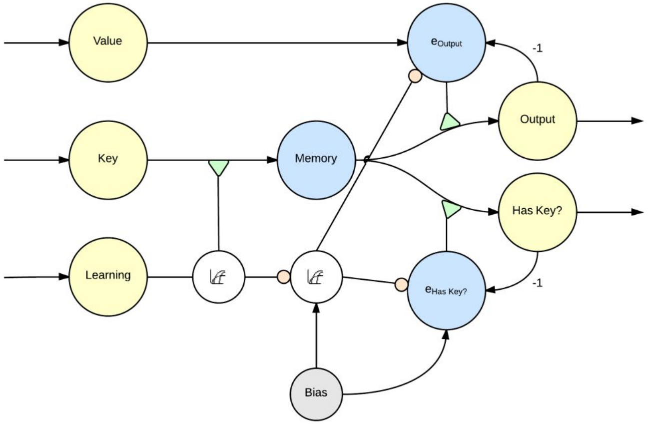
(subject$_2$, predicate$_3$, object$_3$), τ_3

key	value
s_1	$\rightarrow \tau_1\ \mathcal{B}(o_1,p_1)$
s_2	$\rightarrow \tau_2\ \mathcal{B}(o_2,p_2) + \tau_3\ \mathcal{B}(o_3,p_3)$

Need to introduce:

- scalar multiplication a = τ a
- vector superposition c = a + b
- vector binding c = $\mathcal{B}$(a,b)

⇨ Semantic Pointer Architecture
(Eliasmith et al, 2013)



NEF Associative Memory
(Voelker et al, 2014)

yellow circles: passthrough nodes
blue circles: neuron ensembles
orange connections: inhibitory
green connections: modulatory

A Semantic Pointer triplestore

$(\text{subject}_1, \text{predicate}_1, \text{object}_1)$

$(\text{subject}_2, \text{predicate}_2, \text{object}_2)$

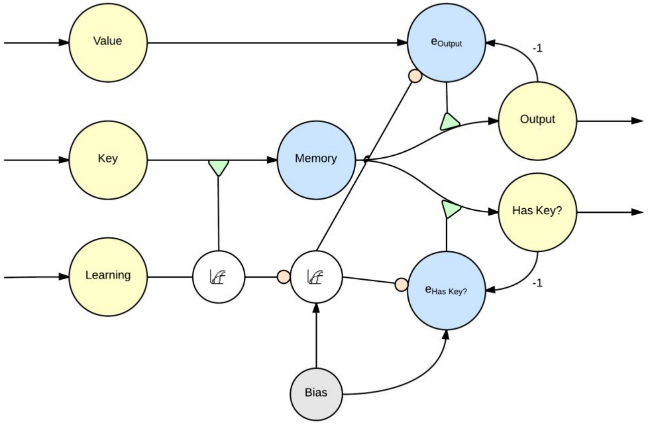
$(\text{subject}_2, \text{predicate}_3, \text{object}_3)$

key	value
s_1 →	$\mathcal{B}(o_1,p_1)$
s_2 →	$\mathcal{B}(o_2,p_2) + \mathcal{B}(o_3,p_3)$

Need to introduce:

- scalar multiplication a = τ a
- vector superposition c = a + b
- vector binding c = $\mathcal{B}$(a,b)

➪ Semantic Pointer Architecture
(Eliasmith et al, 2013)



NEF Associative Memory
(Voelker et al, 2014)

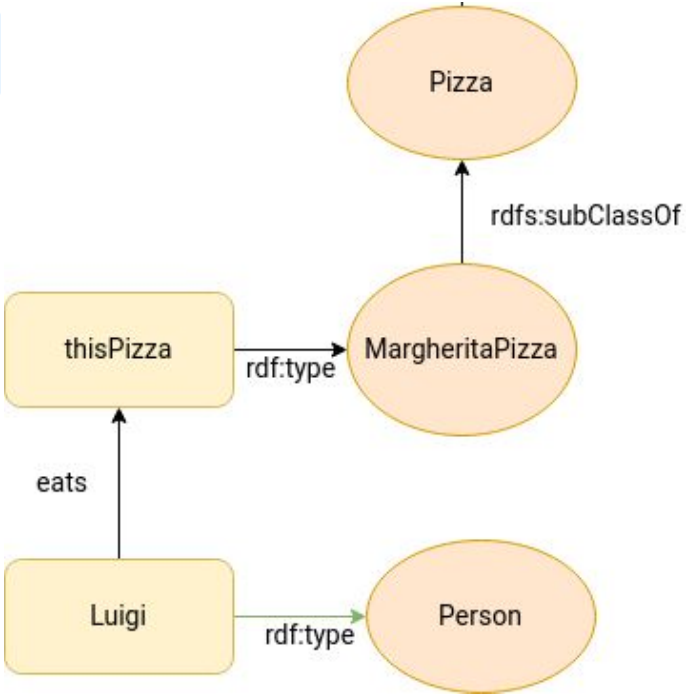
yellow circles: passthrough nodes
blue circles: neuron ensembles
orange connections: inhibitory
green connections: modulatory

inspired by
https://protege.stanford.edu/ontologies/pizza/pizza.owl

Example of inference

Inheritance inference rule:

X `rdf:type` C & C `rdfs:subClassOf` C' ⇒ X `rdf:type` C'



inspired by
https://protege.stanford.edu/ontologies/pizza/pizza.owl

Example of inference

Inheritance inference rule:

x `rdf:type` c & c `rdfs:subClassOf` c' $\Rightarrow$ x `rdf:type` c'

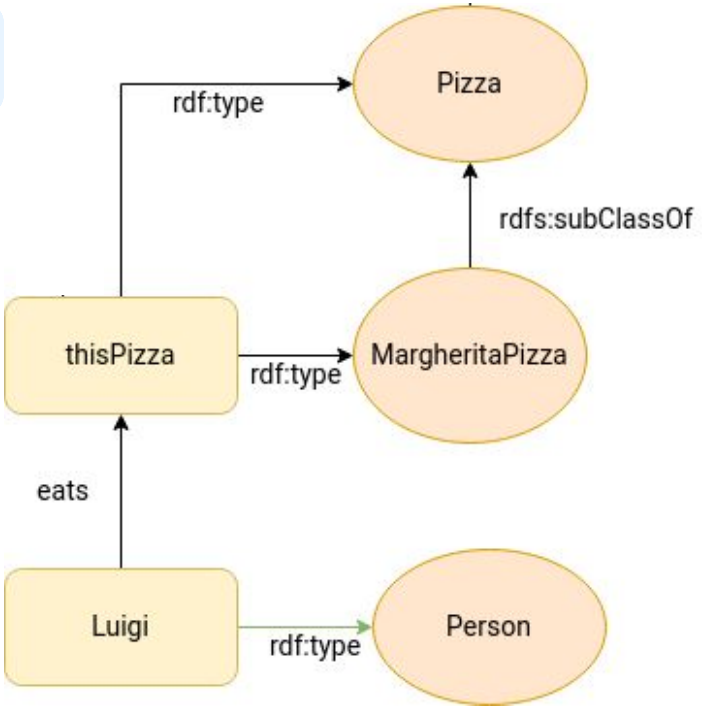
Sufficient condition:

$\mathbf{B}_{\text{type}}\, \mathbf{B}_{\text{subclass}} \longrightarrow \mathbf{B}_{\text{type}}$

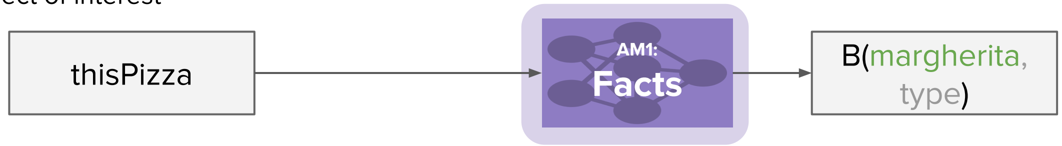
"inference rule" stored into a 2nd AM:

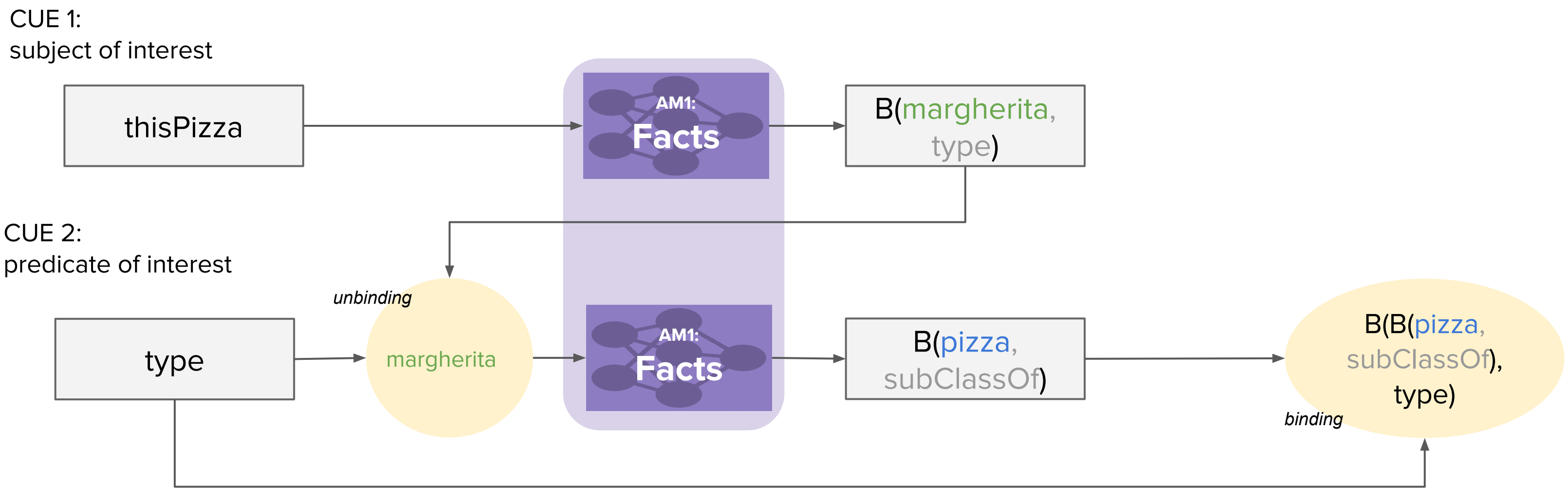
type ⊘ subclass → type

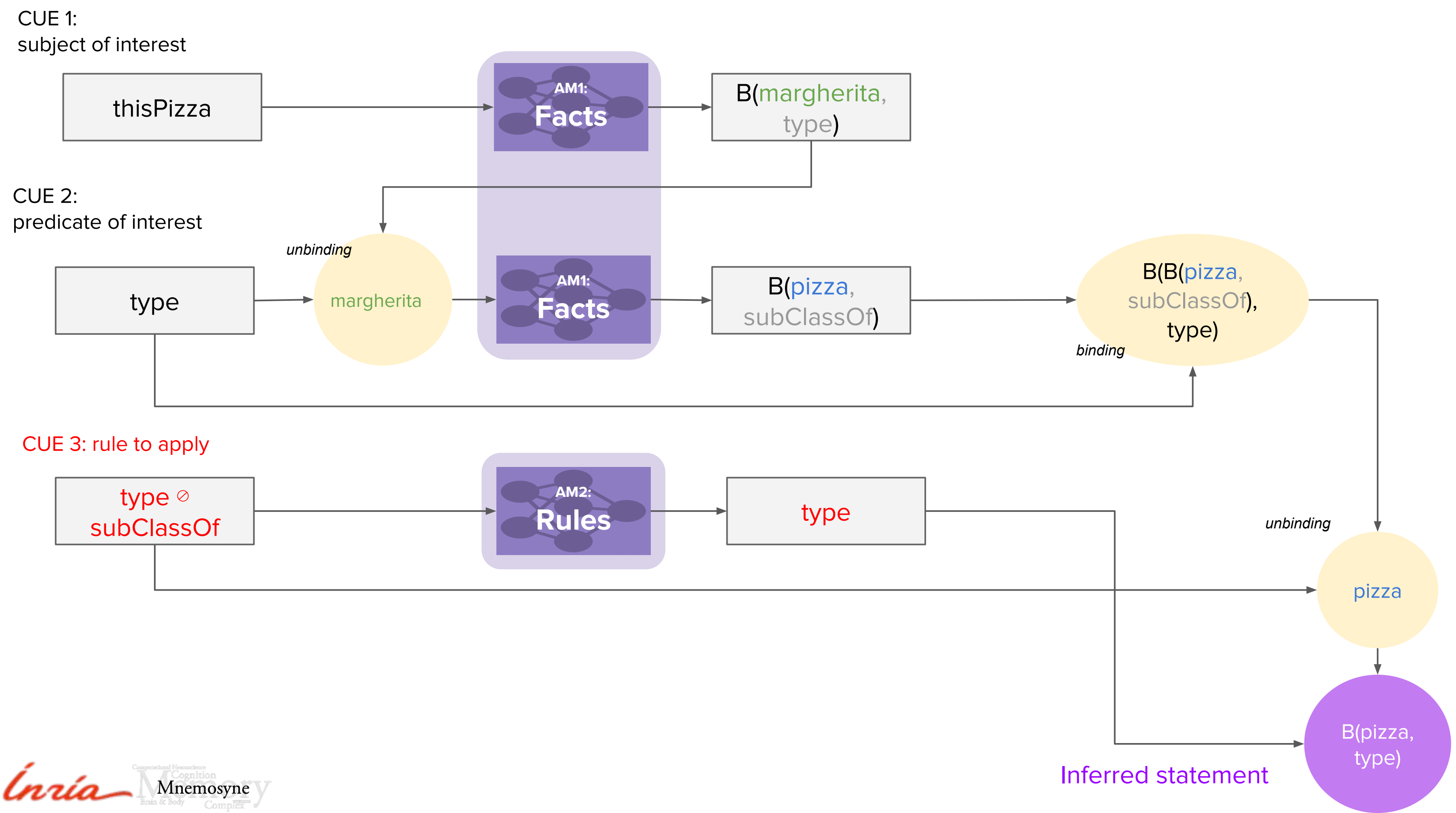
where x ⊘ y is such that $B_{x \oslash y} = B_x\, B_y$



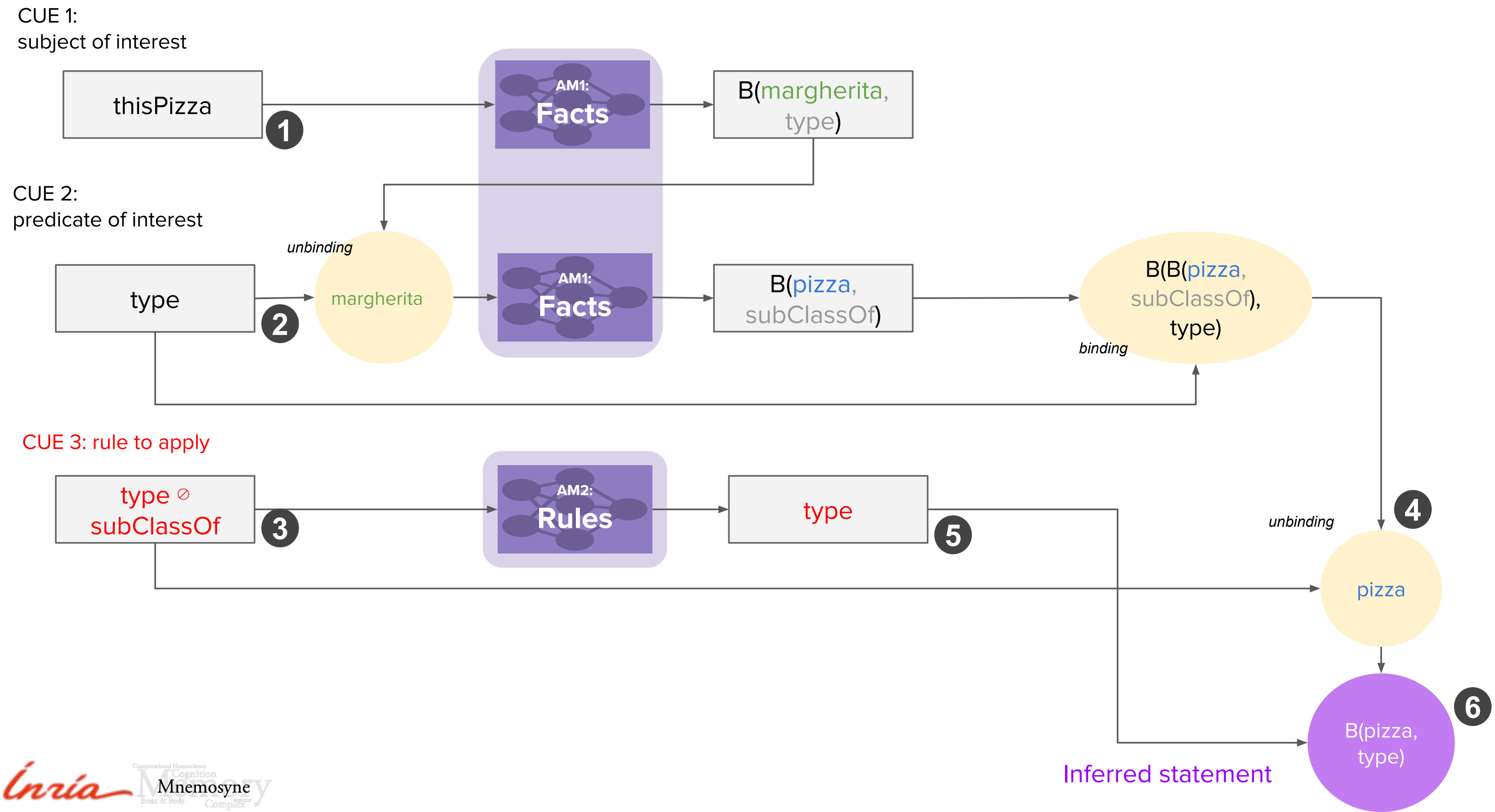
CUE 1:
subject of interest



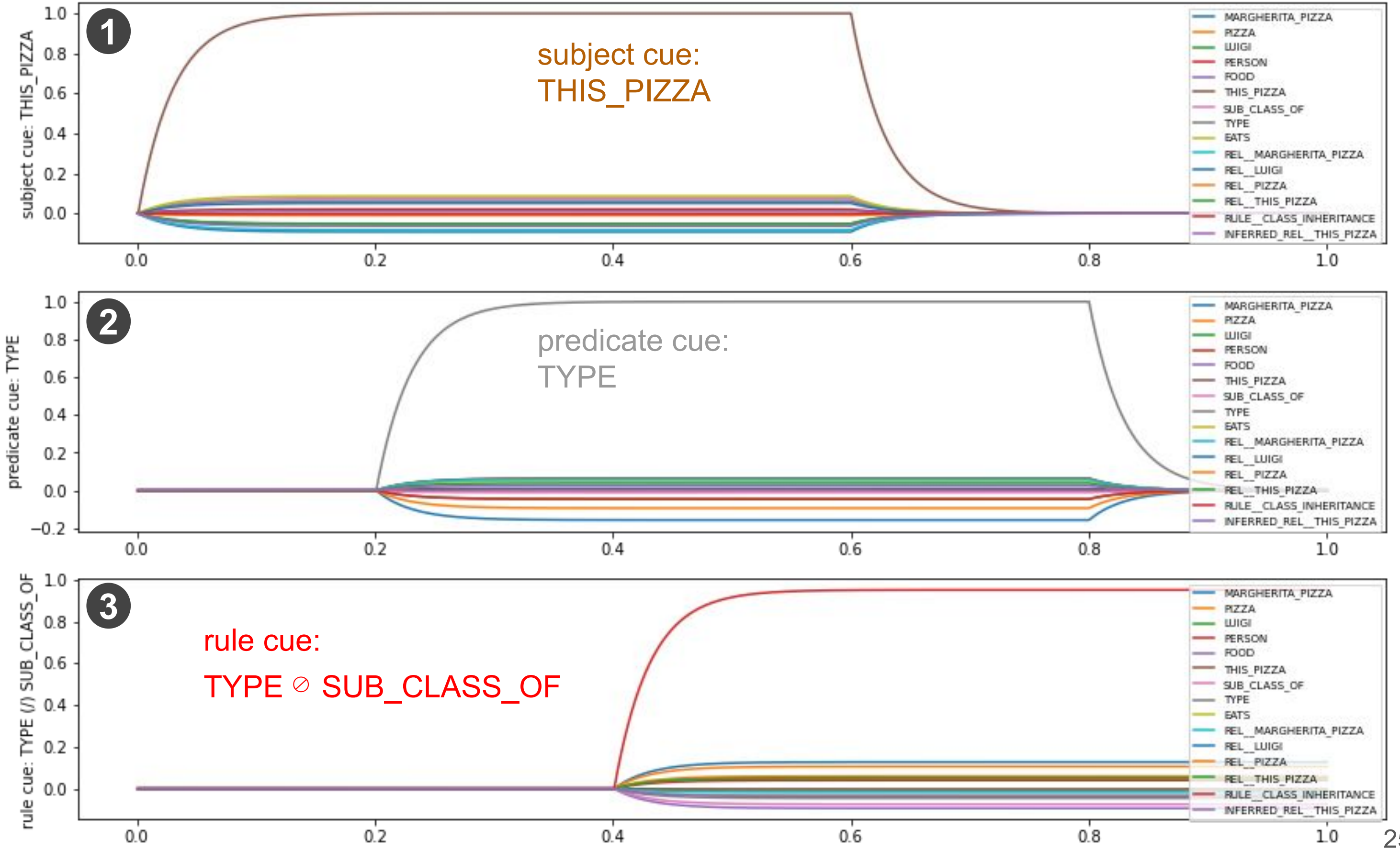

CUE 1:
subject of interest
thisPizza
AM1:
Facts
B(margherita, type)
CUE 2:
predicate of interest
unbinding
type
margherita
AM1:
Facts
B(pizza, subClassOf)
B(B(pizza, subClassOf), type)
binding



Inria Mnemosyne

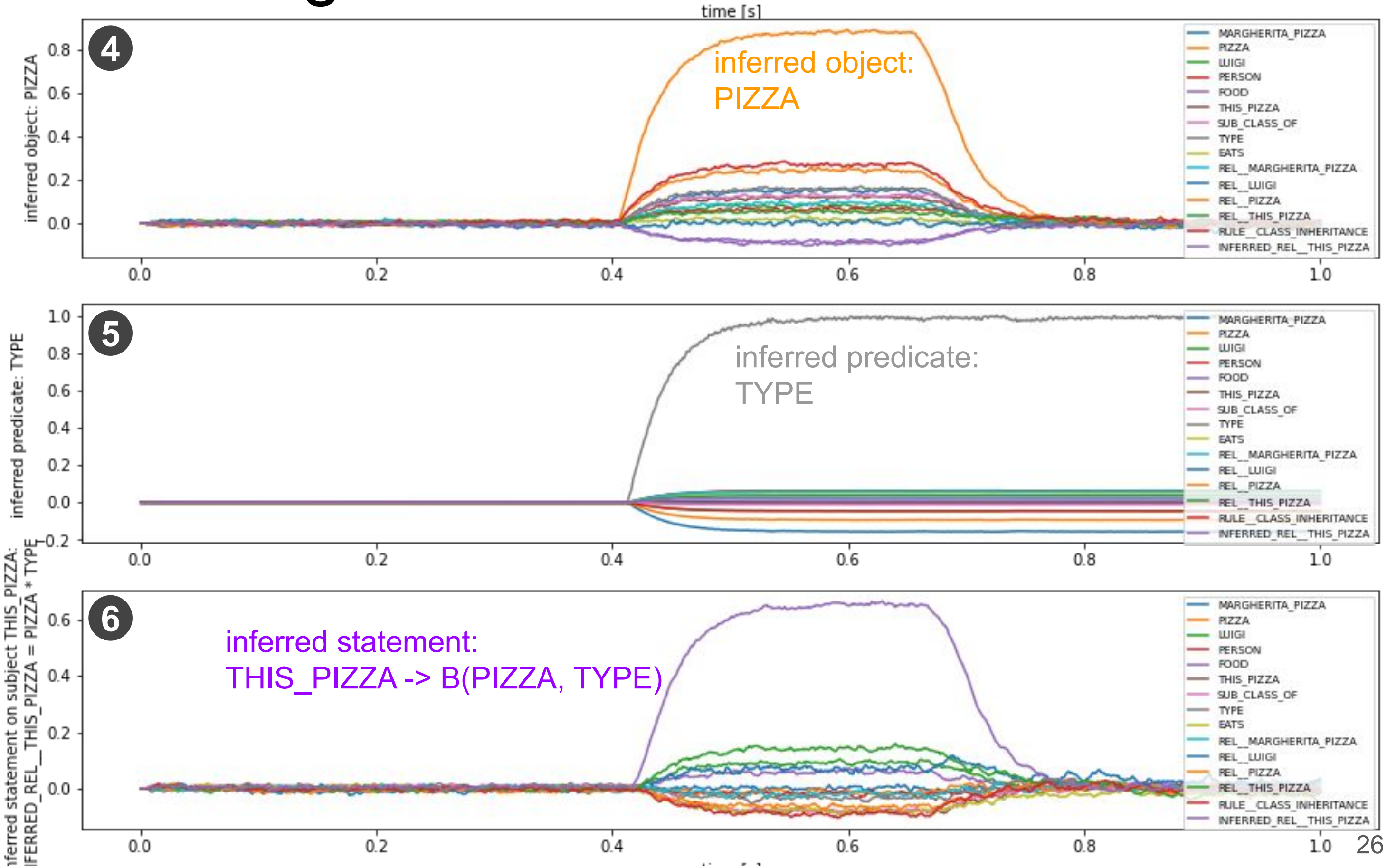

CUE 1:
subject of interest
thisPizza
1
AM1:
Facts
B(margherita, type)
CUE 2:
predicate of interest
type
2
unbinding
margherita
AM1:
Facts
B(pizza, subClassOf)
B(B(pizza, subClassOf), type)
binding
CUE 3: rule to apply
type ⊘ subClassOf
3
AM2:
Rules
type
5
unbinding
4
pizza
6
B(pizza, type)
Inferred statement

Simulation on Nengo



Plotted here are the input cue signals: accounting respectively for the subject, the predicate and the rule to apply.

Simulation on Nengo



Plotted here are the outputs given by the model: the object associated to the given subject and predicate, the simplified predicate obtained from the rule (subClass * type -> type, from the 2nd AM) and the final statement inferred.

Underlying mathematics

Vector-derived Transformation Binding (VTB)

(Gosmann and Eliasmith, 2019)

- Square dimensionality d with d’² = d
- Binding operation defined as:

$$\mathcal{B}(\mathbf{x},\mathbf{y}) = \mathbf{B_y}\mathbf{x} = \begin{bmatrix} \mathbf{B}'_\mathbf{y} & 0 & 0 \\ 0 & \mathbf{B}'_\mathbf{y} & 0 \\ 0 & 0 & \mathbf{B}'_\mathbf{y} \end{bmatrix} \mathbf{x} \quad \text{where} \quad \mathbf{B}'_\mathbf{y} = d^{\frac{1}{4}} \begin{bmatrix} y_1 & y_2 & \cdots & y_{d'} \\ y_{d'+1} & y_{d'+2} & \cdots & y_{2d'} \\ \vdots & \vdots & \ddots & \vdots \\ y_{d-d'+1} & y_{d-d'+2} & \cdots & y_d \end{bmatrix}$$

- Non commutative, non associative
- Distributive and bilinear

VT unbinding

(Gosmann and Eliasmith, 2019)

- VTB admits right approximate inverses:
  - the right approximate inverse **y~** for **y** is such that:

$$\forall \mathbf{x}, \mathcal{B}(\mathcal{B}(\mathbf{x}, \mathbf{y}), \mathbf{y}^{\sim})) = \mathbf{B}_{\mathbf{y}^{\sim}} \mathbf{B}_{\mathbf{y}} \mathbf{x} = \mathbf{x}$$

  - **B**$_{\mathbf{y}}$ almost orthogonal: $\mathbf{B}_{\mathbf{y}^{\sim}} = \mathbf{B}_{\mathbf{y}}^{\top}$

    => just permute the elements of y to get y~

- Right identity vector **i**$_{\mathbf{B}}$ such that $\forall x, \mathcal{B}(\mathbf{x}, \mathbf{i_B}) = \mathbf{x}$ ie $\mathbf{B}_{\mathbf{i_B}} = \mathbf{I}$

VT unbinding

Check if a property is a predicate for a given resource and, if so, retrieve the corresponding object:

$$\mathbf{x} \rightarrow \mathbf{z} = \mathcal{B}(\mathbf{y}, \mathbf{p})$$

- Unbind p: $\mathcal{B}(\mathbf{z}, \mathbf{p}^{\sim}) = \mathcal{B}(\mathcal{B}(\mathbf{y}, \mathbf{p}), \mathbf{p}^{\sim}) = \mathbf{y}$
- Check if the result belongs to the vocabulary: $\exists \mathbf{y}' \in \text{voc}, \mathbf{y} \cdot \mathbf{y}' = 1$?
  (if similarity is -1 < τ < 1, the statement is possible to some extent)

VT unbinding

What if want to check if two resources are linked to each other by a predicate and, if so, retrieve the corresponding property (left unbinding)?

We can flip the order of the operands using a well-chosen matrix $\mathbf{B}_{\leftrightarrow}$: $\mathcal{B}(\mathbf{y},\mathbf{p}) = \mathbf{B}_{\leftrightarrow}\mathcal{B}(\mathbf{p},\mathbf{y})$

=> Check if two resources are linked to each other by a predicate and, if so, retrieve the corresponding property:

- Flip the operands (multiply by $\mathbf{B}_{\leftrightarrow}$)
- Unbind y (multiply by the transpose of B_y)
- Check if the result is semantically close to a word from the vocabulary (compute the dot product)

About the operator $\oslash$

We introduce a vector composition operator making explicit the composition of two VTB operations, namely:

$$\mathbf{B_v} = \mathbf{B_y}\,\mathbf{B_x} \Leftrightarrow \mathbf{v} \stackrel{\text{def}}{=} \mathbf{y} \oslash \mathbf{x}$$

computable in $O\left(d^{\frac{3}{2}}\right)$

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

It commutes with the approximate inverse:(vérifier formulation)

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

where

$$\mathbf{x}^{\sim} \oslash \mathbf{x} \simeq \mathbf{i_B}$$

Towards more reasoning power

Beyond the type inheritance inference

- Covering all RDFS entailments reduced to the rule schema developed here, e.g.:
    - property inference (e.g., if Bob is the son of Ali then Bob is of the same family as Ali)
    - range inference (e.g., if Bob is the son of Ali, then Bob is a male)
    - domain inference (e.g., if Bob is the son of Ali, then Ali is a human)

  etc…

- To what extent could it be applicable to description logics and OWL entailment rules?

- SWRL rules (i.e., inference rules expressed directly as rewriting rules on variables) stored in associative memories?

All RDFS entailments

- Everything is a resource
  - IF `x p y` THEN
    - `x rdf:type rdfs:Resource`
    - `y rdf:type rdfs:Resource`
- Any class is a subclass of rdfs:Resource
  - IF `c rdf:type rdfs:Class` THEN
    - `c rdfs:subClassOf rdfs:Resource`
- Type propagation
  - IF `c2 rdfs:subClassOf c1`
  - AND `x rdf:type c2`
    - THEN `x rdf:type c1`
- Subsumption reflexivity
  - IF `c rdf:type rdfs:Class`
    - THEN `c rdfs:subClassOf c`
- Subsumption transitivity
  - IF `c2 rdfs:subClassOf c1`
  - AND `c3 rdfs:subClassOf c3`
    - THEN `c3 rdfs:subClassOf c1`
- Property propagation
  - IF `p2 rdfs:subPropertyOf p1`
  - AND `x p2 y`
    - THEN `x p1 y`
- Property subsumption reflexivity
  - IF `p rdf:type rdf:Property`
    - THEN `p rdfs:subPropertyOf p`
- Property subsumption transitivity
  - IF `p2 rdfs:subPropertyOf p1`
  - AND `p3 rdfs:subPropertyOf p2`
    - THEN `p3 rdfs:subPropertyOf p1`
- Type inference (domain)
  - IF `p rdfs:domain d` AND `x p y`
    - THEN `x rdf:type d`
- Type inference (range)
  - IF `p rdfs:range r` AND `x p y`
    - THEN `y rdf:type r`

inspired by https://protege.stanford.edu/ontologies/pizza/pizza.owl

Type inheritance

x `rdf:type` c & c `rdfs:subClassOf` c' ⇒ x `rdf:type` c'

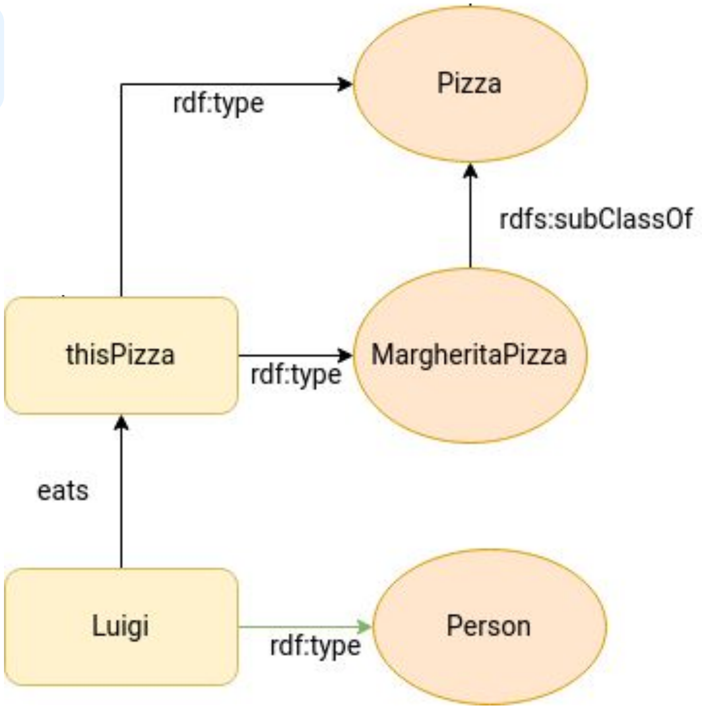
Sufficient condition:

$\mathbf{B}_{\text{type}}\, \mathbf{B}_{\text{subclass}} \longrightarrow \mathbf{B}_{\text{type}}$

“inference rule” stored into a 2nd AM:

type ⊘ subclass → type

where $x \oslash y$ is such that $B_{x \oslash y} = B_x B_y$

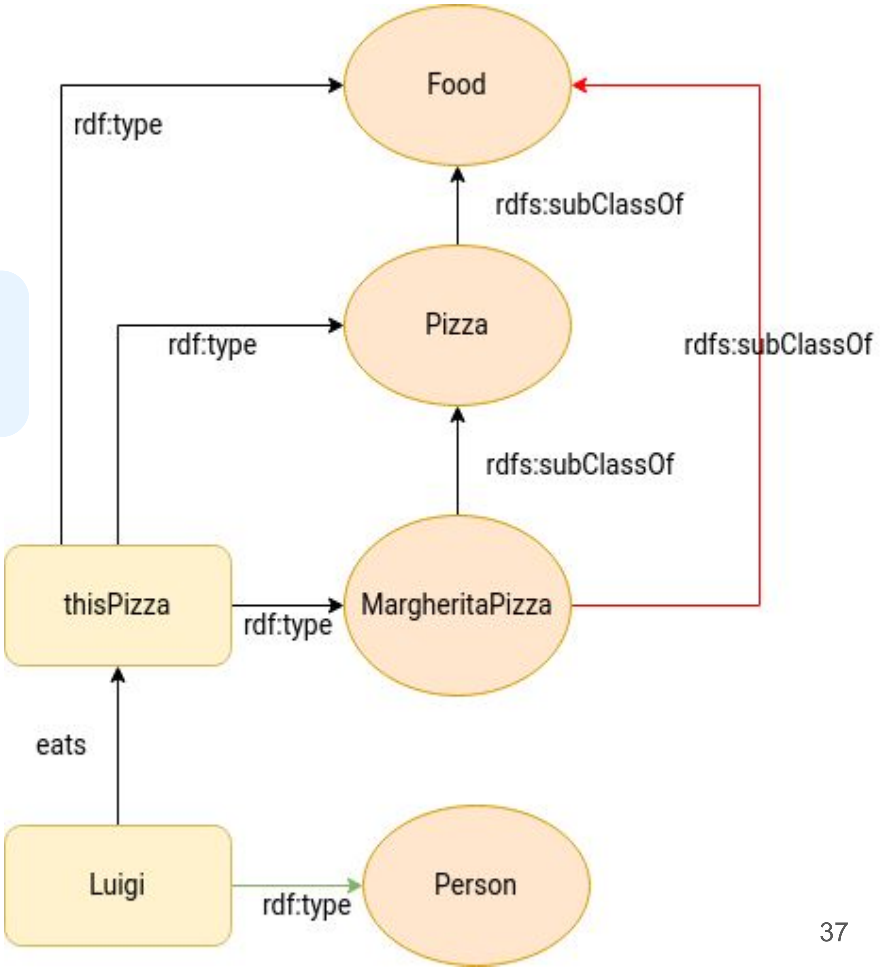


Subsumption transitivity

C rdfs:subClassOf C' & C' rdfs:subClassOf C''
⇒ C rdfs:subClassOf C''

Stored in 2nd AM as:

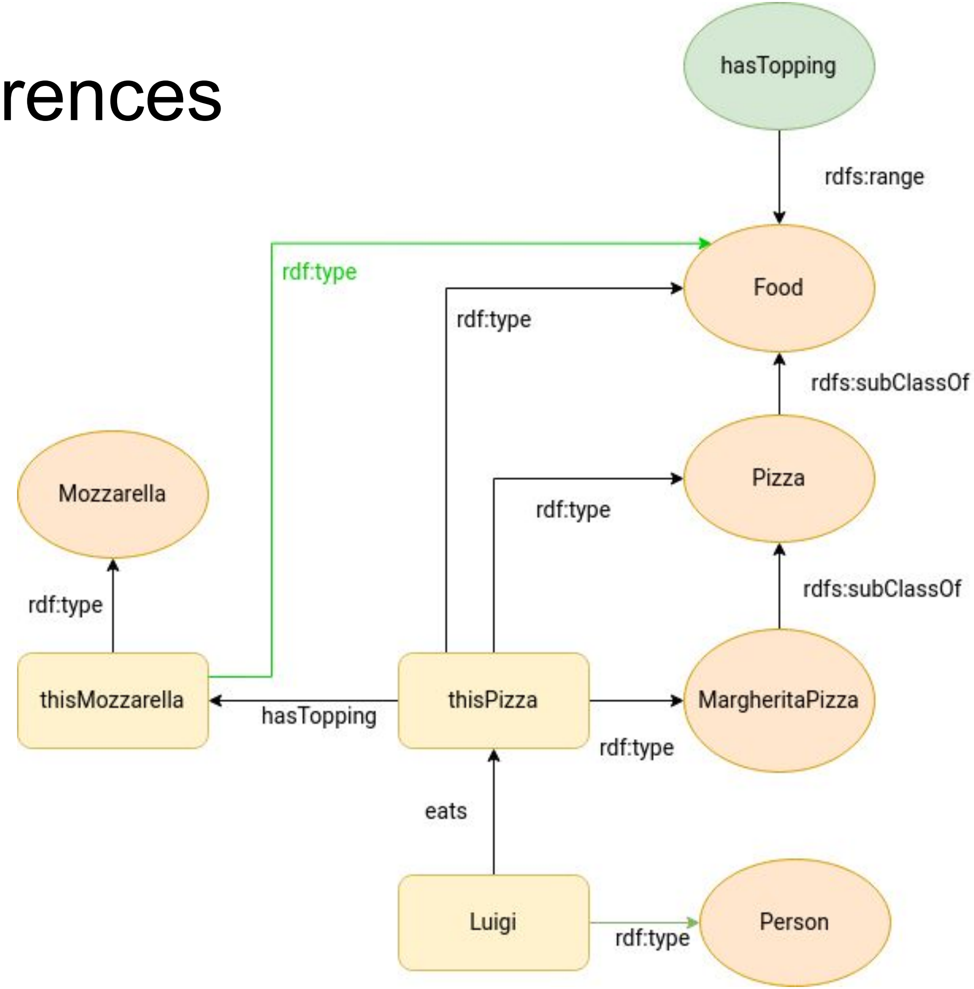
subClassOf ⊘ subClassOf → subClassOf



Domain and range inferences

p rdfs:domain c & x p y
⇒ x rdf:type c

p rdfs:range c & x p y
⇒ y rdf:type c



Domain and range inferences

p `rdfs:domain` c & x p y
⇒ x `rdf:type` c

p `rdfs:range` c & x p y
⇒ y `rdf:type` c

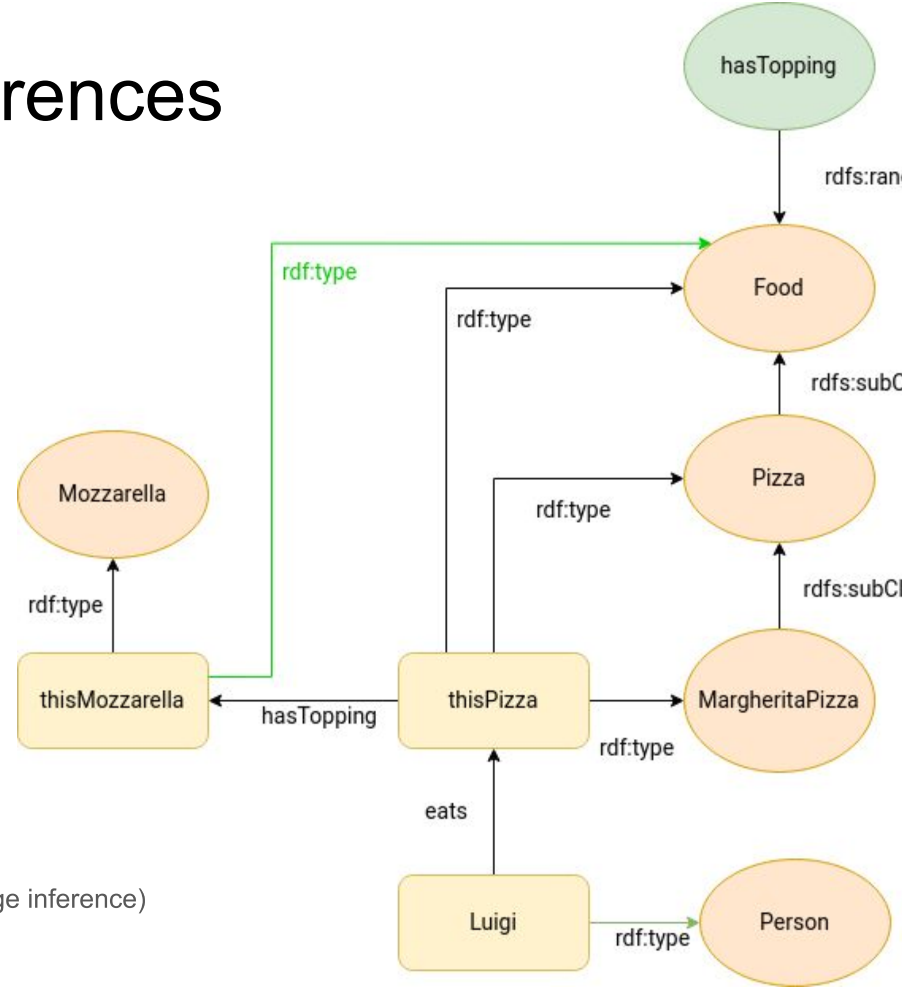
We introduce an UNKNOWN vector Υ (accounting for OWA) such that:

$$\mathbf{B}_{\mathbf{B}_{\mathbf{domain}} c} \Upsilon \Rightarrow \mathbf{B}_{\mathbf{type}} c$$

AM1
$$p \to \mathbf{B}_{\mathbf{domain}} c$$
$$x \to \mathbf{B}_{\mathbf{p}} \Upsilon$$

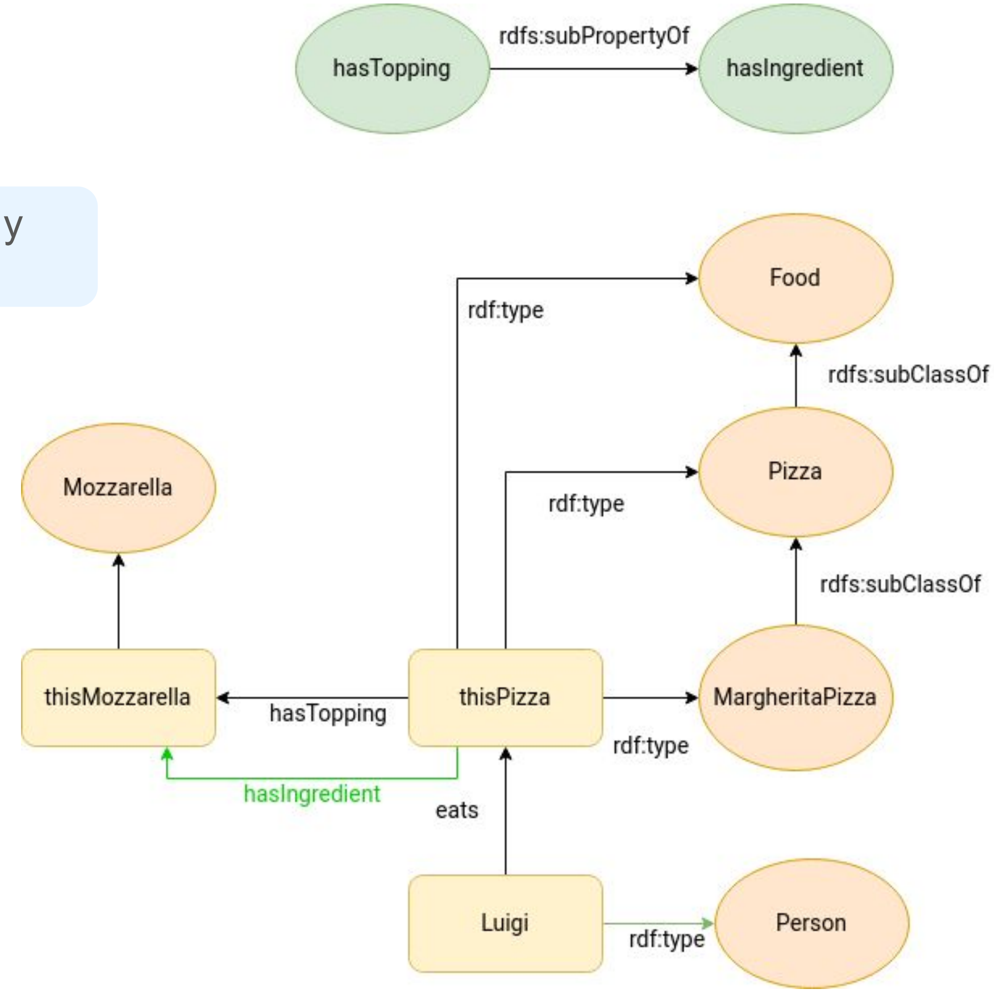
AM2
$$x \to \mathbf{B}_{\mathbf{type}} c$$

(same with range inference)



Inria Mnemosyne

Property propagation



p `rdfs:subPropertyOf` q & x p y
⇒ x q y

Using our previous architecture, this kind of inference is only possible when

$$\textbf{subPropertyOf} = i_B$$

Is this consistent?
Why not do the same for all “is_a” relationships (subClassOf, type) ?...
=> ongoing work

Gabriel's internship: subproperty inference

$$ {}^{po}_{s}\mathcal{B}\ {}^{po}_{s}\mathcal{B}\mathbf{subPropertyOf} =^{po}_{s} \mathcal{B} $$

would mean $$\mathbf{subPropertyOf} = i_B$$

Is this consistent? Why not do the same for all "is_a" relationships (subClassOf, type) ?... => ongoing work (requires a slightly different architecture)

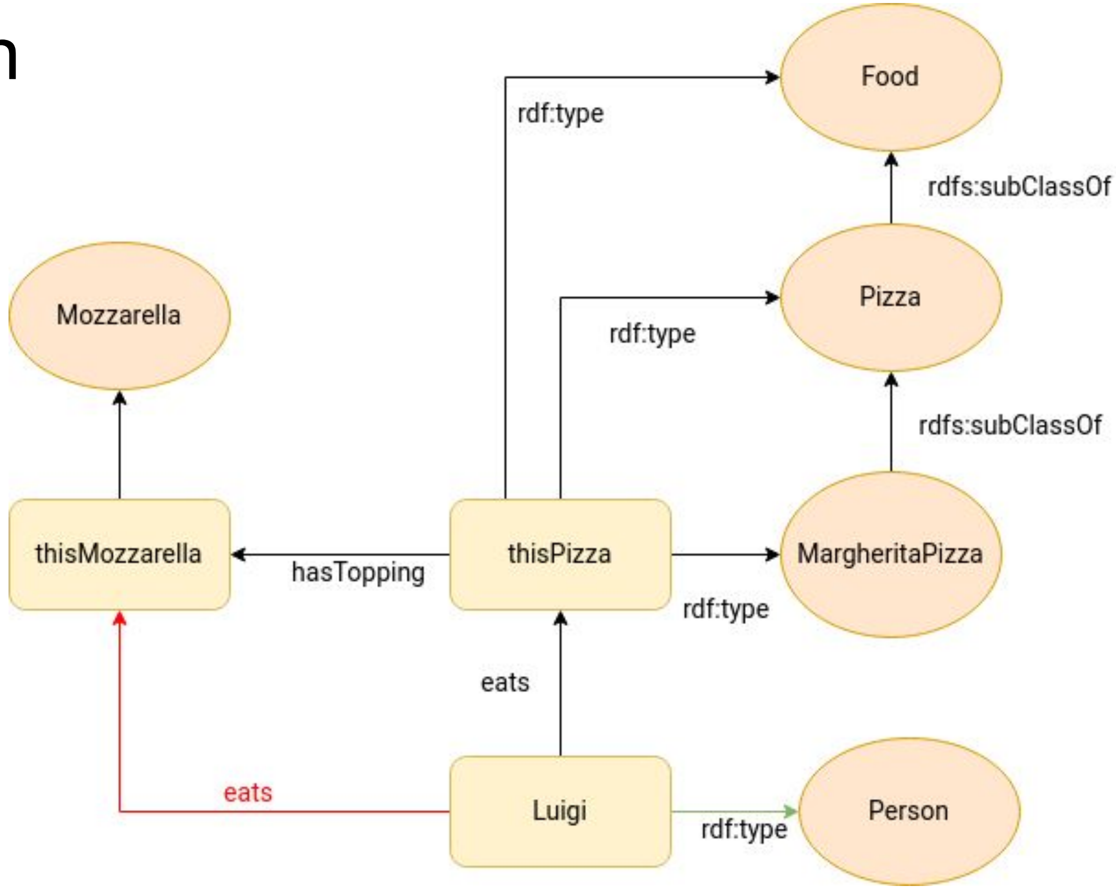
Property composition

Rules such as

eats ⊘ hasTopping → eats

to infer relational compositions

(not part of RDFS entailments, but similar to OWL *property chains*)

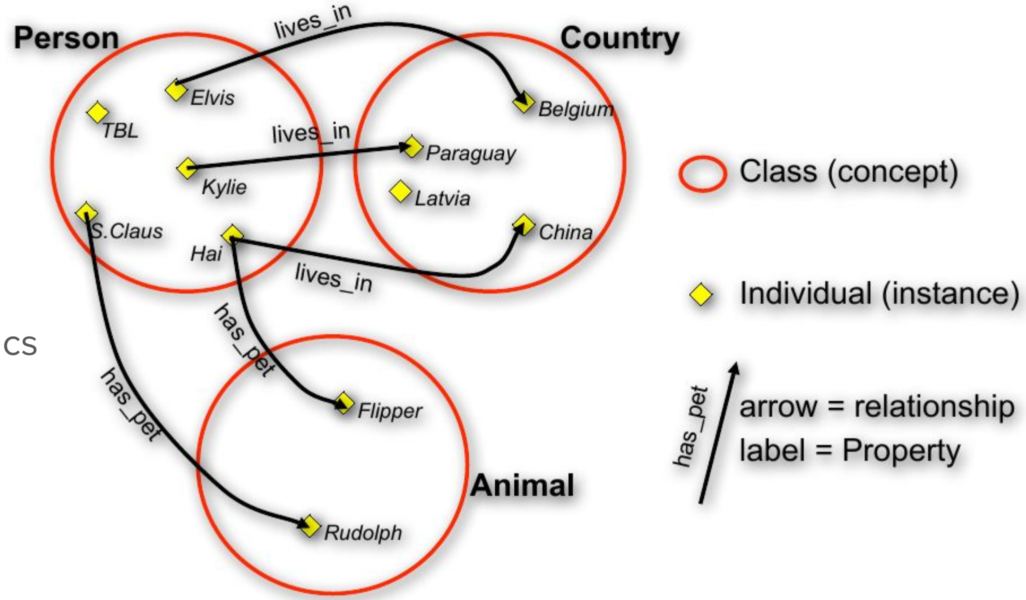


OWL: another representation for knowledge

Web Ontology Language

semantics based on description logics
=> fragment of 1st-order logic

more reasoning capabilities,
computability and decidability
under constraints



[A Practical Introduction to Protégé OWL]

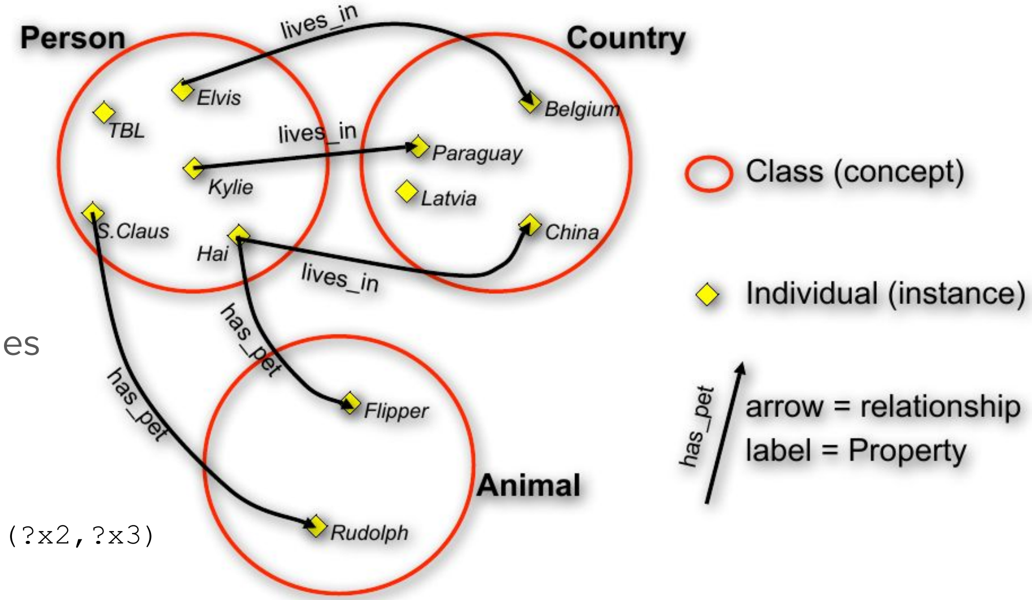
OWL + SWRL

SWRL:
a Semantic Web Rule Language
combining OWL and RuleML

Completes OWL with implication rules (written with variables, e.g. ?x):

```
hasParent(?x1,?x2)  ∧ hasBrother(?x2,?x3)

⇒ hasUncle(?x1,?x3)
```



A Practical Introduction to Protégé OWL

Alternate implementation:
a compressed triplestore vector

An alternate implementation of entailment closure

All triplets are memorized in a store that is merely a sum of vectors:

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

allowing to enumerate all (subject, object) of a property, i.e., select by predicate, also by predicate and subject or by predicate and object.

Given an entailment rule of the form:

$$(\$s_1 \; \$p_1 \; \$o_1.) \text{ and } (\$s_2 \; \$p_2 \; \$o_2.) \Rightarrow (\$s_0 \; \lambda \, \$p_0 \; \$o_0.)$$

where $\lambda \in [0, 1]$ allows to consider approximate inference.

Inria Mnemosyne

Note that for an efficient retrieval, when storing triples, we can index them either by subject, object or property:

$${}^{po}_{s}\mathcal{B}(p,y) = \mathcal{B}(p,y) = \mathcal{B}_y p \qquad \text{(indexing triples by subject)}$$

$${}^{so}_{p}\mathcal{B}(x,y) = \mathcal{B}_y^T x = \mathcal{B}_{y^\sim} x \qquad \text{(indexing triples by property)}$$

$${}^{sp}_{o}\mathcal{B}(x,p) = \mathcal{B}_p^T x^\sim = \mathcal{B}_{p^\sim} x^\sim \qquad \text{(indexing triples by object)}$$

$${}^{op}_{s}\mathcal{B}(y,p) = \mathcal{B}(p,y) = \mathcal{B}_y p$$

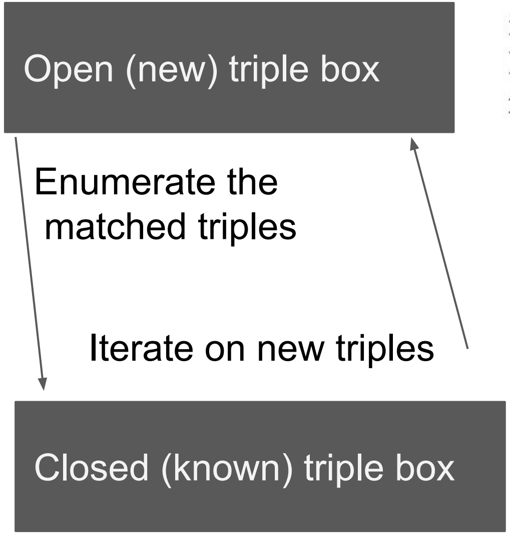
$${}^{os}_{p}\mathcal{B}(y,x) = \mathcal{B}_y^T x = \mathcal{B}_{y^\sim} x$$

$${}^{ps}_{o}\mathcal{B}(p,x) = \mathcal{B}_p^T x^\sim = \mathcal{B}_{p^\sim} x^\sim$$

B is a bilinear form.

The class inheritance example:

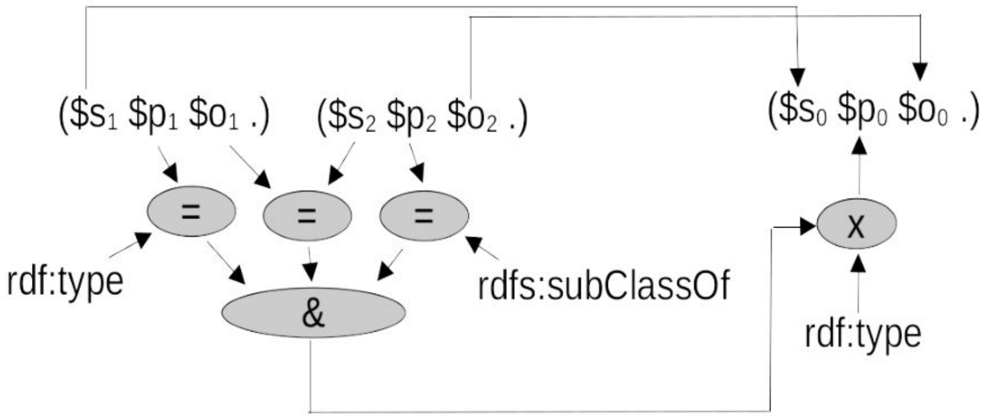
$$(\$s\ \texttt{rdf:type}\ \$c_1.)\ \text{and}\ (\$c_1\ \texttt{rdfs:subClassOf}\ \$c_2.) \Rightarrow (\$s\ \texttt{rdf:type}\ \$c_2.)$$



input A new triple $(\$s_0\ \$p_0\ \$o_0.)$ and a closed set of triples $\{(\$s_i\ \$p_i\ \$o_i.)\cdots\}$.
let $\{(\$s_0\ \$p_0\ \$o_0.)\}$ 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 $\$p_0 = \texttt{rdf:type}$ **then**
    for all $(\$s_i\ \$p_i\ \$o_i.), \$p_i = \texttt{rdfs:subClassOf}\ \&\ \$s_i = \$o_0$, in the closed triple set **do**
      add $(\$s_0\ \texttt{rdf:type}\ \$o_i.)$ to the open triple set.
    end for
  else if $\$p_0 = \texttt{rdf:subClassOf}$ **then**
    for all $(\$s_i\ \$p_i\ \$o_i.), \$p_i = \texttt{rdf:type}\ \&\ \$o_i = \$s_0$, in the closed triple set **do**
      add $(\$s_i\ \texttt{rdf:type}\ \$o_0.)$ 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

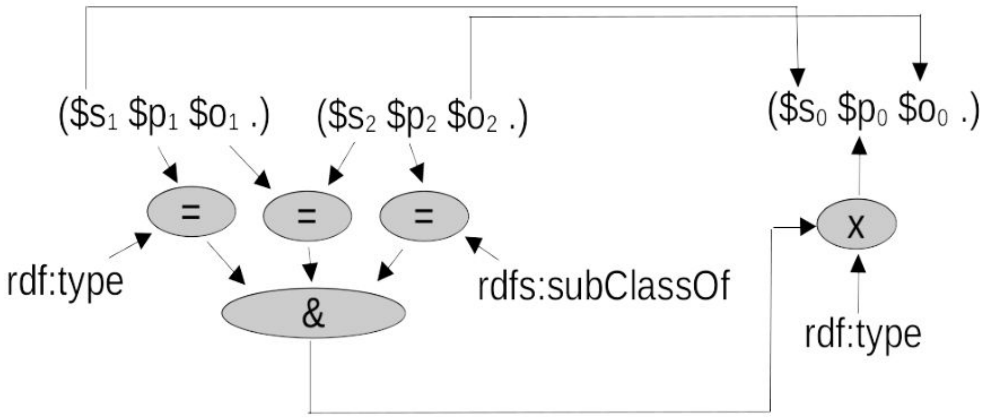
The class inheritance example:

- The rule corresponds to a hardwiring of a 2 input 1 output operator
- Approximate equality (i.e. similarity) is intrinsic to the calculus
- Approximately true property is intrinsic to the calculus
- The ‘&’ operator can be implemented as a product (but not only).



The class inheritance example:

- The rule corresponds to a hardwiring of a 2 input 1 output operator
- Approximate equality (i.e. similarity) is intrinsic to the calculus
- Approximately true property is intrinsic to the calculus
- The ‘&’ operator can be implemented as a product (but not only).



- For “negative” inferences (i.e. λ < 0) a different (symmetric) rule is required :

$$(\$s \neg\texttt{rdf:type}\ \$c_1.)\ \text{and}\ (\$c_2\ \texttt{rdfs:subClassOf}\ \$c_2.) \Rightarrow (\$s \neg\texttt{rdf:type}\ \$c_2.)$$

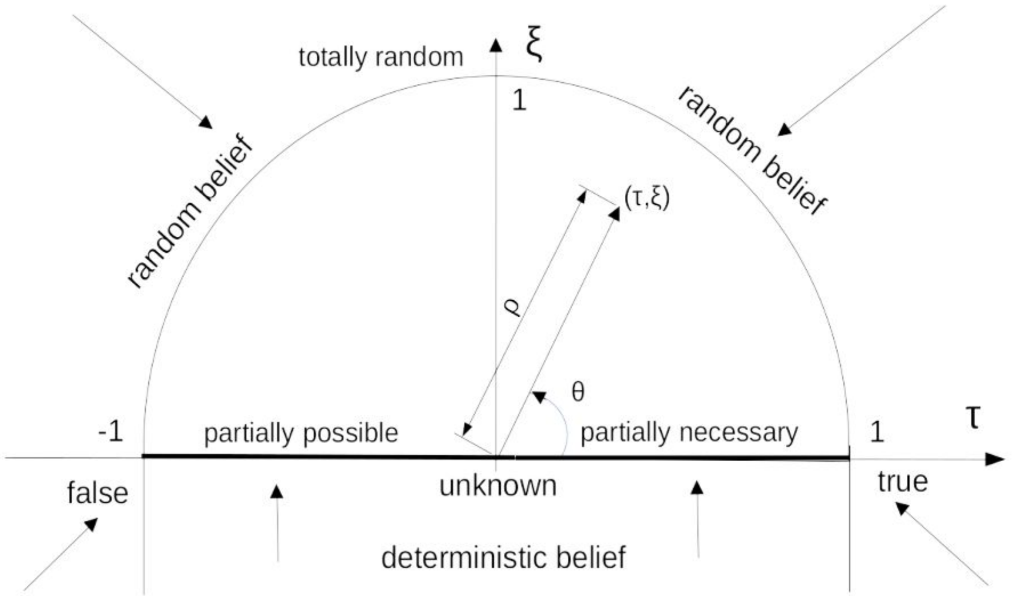
Accounting for uncertainty: possibility, necessity

Linking with possibility π / necessity ν theory

$$0 \leq \nu(E) \leq p(E) \leq \pi(E) \leq 1$$

$$\begin{aligned} \tau &= \pi + \nu - 1 \in [-1, 1] \\ \rho &= \nu - \pi + 1 \in [0, 1] \end{aligned}$$

A new interpretation of **necessity** and **possibility** as probability bounds



Linking with possibility π / necessity ν theory

Given a set of a-priori knowledge $\mathcal{K}$ and observation O, (π,ν) is properly defined as:

$$\begin{array}{rcl} \nu(E) & = & \min_{\kappa \in \mathcal{K}} P_\kappa(E|O) \\ \pi(E) & = & \max_{\kappa \in \mathcal{K}} P_\kappa(E|O) \end{array}$$

Given a possibility/necessity distribution (π,ν) we can define

- a probability distribution of maximal entropy
- for "minimally corrected" compatible distributions $\tilde{\nu} \leq \nu, \pi \leq \tilde{\pi}$

Linking with possibility π / necessity ν theory

Compatibility with Boolean and three-value logic calculus can be derived

Partially obtained by distribution monotonicity

$$E \subseteq E' \Rightarrow \nu(E) \leq \nu(E') \text{ and } \pi(E) \leq \pi(E')$$

Fully compatible in the “deterministic case” (vanilla theory)
Requires additional constraints in the 2D case for 0 or ± 1 values

In an open world (i.e. if we do not consider an exhaustive universe),
logical derivation related to the the complement of an event are not possible.

The correct model underneath is modal logic.

Linking with possibility π / necessity ν theory

In VSA (π, ν) could be implemented either as a **pair of real numbers**, or as a **complex number**.

- Similarity can be defined as the dot product $Re(xy^*)$ in the hermitian space $\mathbb{C}^d$ with the same interpretation
- VTB algebra and related operators seem generalizable

However there is no real “gain” to consider complex number apart from using the complex product instead of $2d$ matrix product.

Conclusion/Perspectives

- Distributive, tightly coupled approach, drawing inspiration from biology and psychology
- 2 associative memories:
  - AM1 to store the ontological "facts"
  - AM2 to store the inference rules => used to disclose implicit relationships/memberships
- Alternate implementation: compressed triplestore (computing the transitive closure)
- Achievable inferences:
  - RDFS entailments:
    - concept inclusion an inheritance
    - property domains and ranges
  - extension to OWL ontologies: class complement, intersection, etc

+ Scalability, completeness, soundness on larger databases?
+ How to manage uncertainty? (possibility-necessity approach?)
+ How to update the associative memory to add the inferred relationships?
+ How to learn the inference rules from patterns discovered in AM1 ?

Nengo associative memory is read-only (though some work is aiming to implement learning within AMs, cf Voelker et al 2014)

Thank you!

Special thanks to:
Hugo Chateau-Laurent
Théophane Valleys
Gabriel Doriath Döhler
Terrence Stewart

References: Alexandre 2019; Allemang, Hendler, and Gandon 2020; Ayadi et al. 2019; Bekolay et al. 2014; Choo 2018; Chui and Mhaskar 2018; Cohen, Yang, and Mazaitis 2017; Crawford, Gingerich, and Eliasmith 2016; Denœux, Dubois, and Prade 2020; Eidoon, Yazdani, and Oroumchian 2008; Eliasmith 2013; Eliasmith and Anderson 2002; Eliasmith et al. 2012; Gallego et al, 2017, Garcez and Lamb 2020; Gayler 2003; Gosmann 2018; Gosmann and Eliasmith 2019; Graves, Wayne, and Danihelka 2014; Grosof et al. 2003; Guo et al. 2016; Hohenecker and Lukasiewicz 2020; Horridge 2011; Horst 2005; Jiménez, Elizalde, and Raj 2018; Komer et al. 2019; Lallement, Hilario, and Alexandre 1995; Levesque 1986; Levy and Gayler 2008; Mandler 2011; McClelland and Rogers 2003; Mercier et al. 2021; Nallapu 2019; Petrucci, Ghidini, and Rospocher 2016; Phan et al. 2017; Plate 1995; Pulvermüller 2013; Riegel et al. 2020; Romero, David, and Lille 2019;Rusawuk 2018; Sajjad, Docherty, and Tyshetskiy 2019; Sauerwald and Zanetti 2019; Schlegel, Neubert, and Protzel 2020; Shi et al. 2020; Simpkin et al. 2018; Smith 1994; Stewart, Choo, and Eliasmith 2010; Stewart, Tang, and Eliasmith 2011; Sun and Alexandre 2013; Tettamanzi, Zucker, and Gandon 2017; Tous and Delgado 2006; Voelker, Crawford, and Eliasmith 2014; Wang, Qiu, and Wang 2021; Xiao, Huang, and Zhu 2015; Zhu et al. 2017