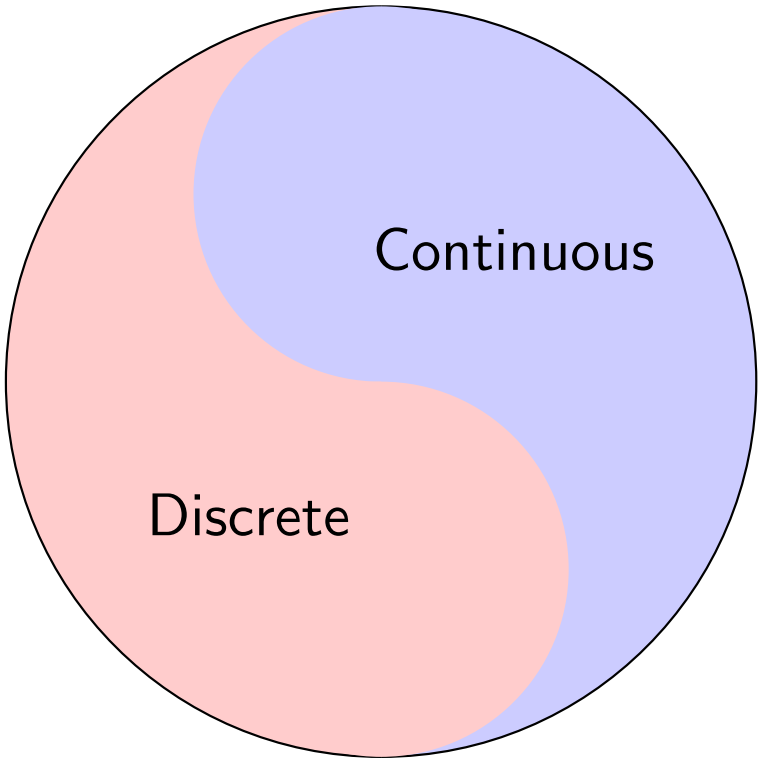
Learning and Inference in Hybrd Models

Luciano Serafini

Fondazione Bruno Kessler
Trento, Italy

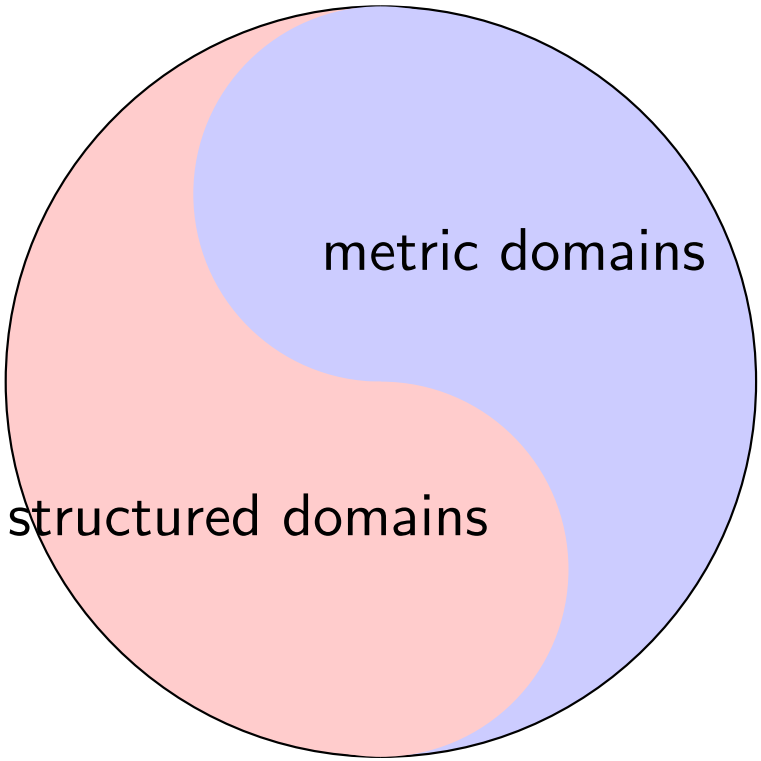
XAI seminars
Imperial College London UK

Hybrid Model



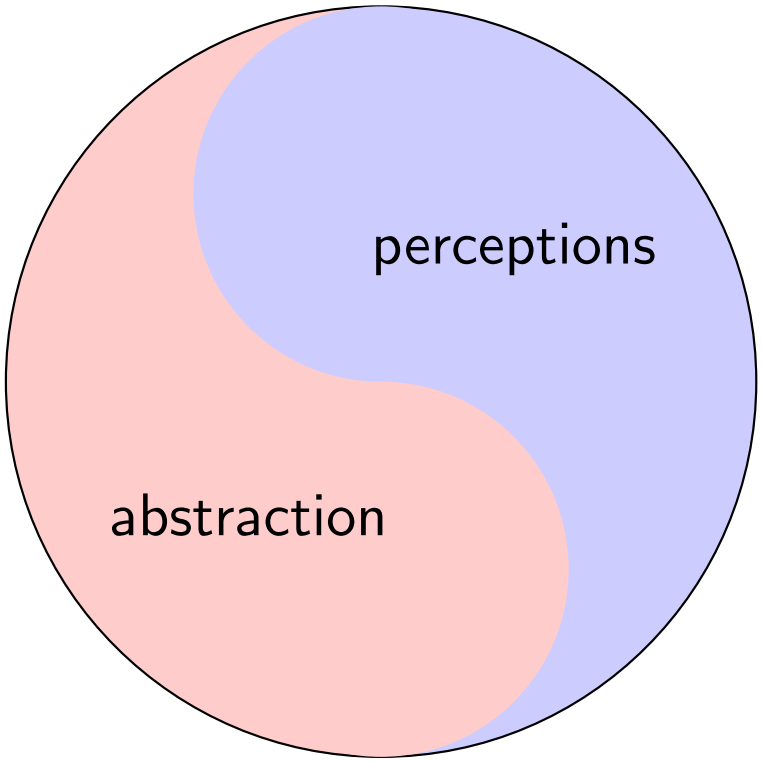
- design hybrid models;
- learning with hybrid models;
- inference with hybrid models.

Hybrid Model



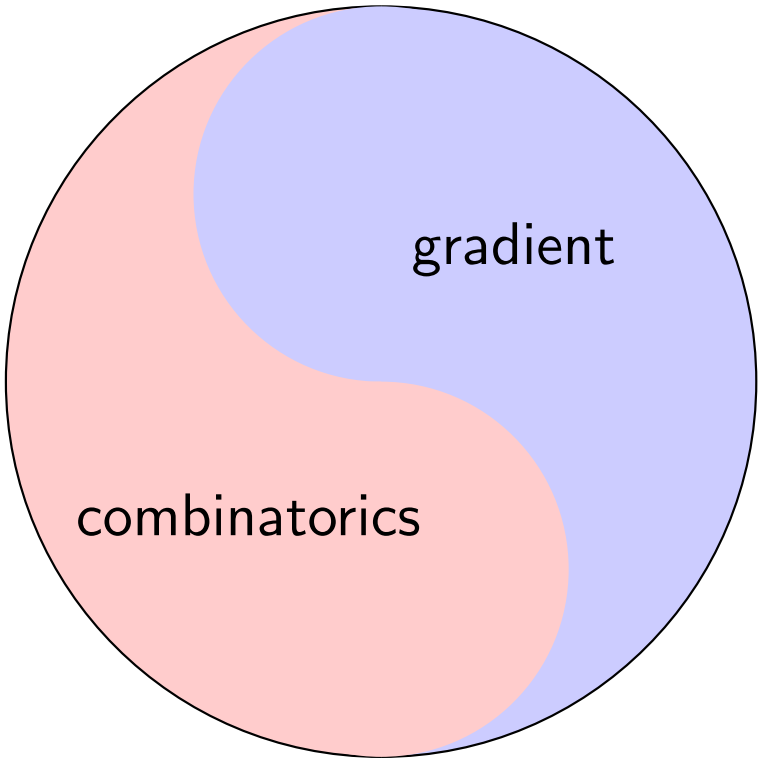
- design hybrid models;
- learning with hybrid models;
- inference with hybrid models.

Hybrid Model



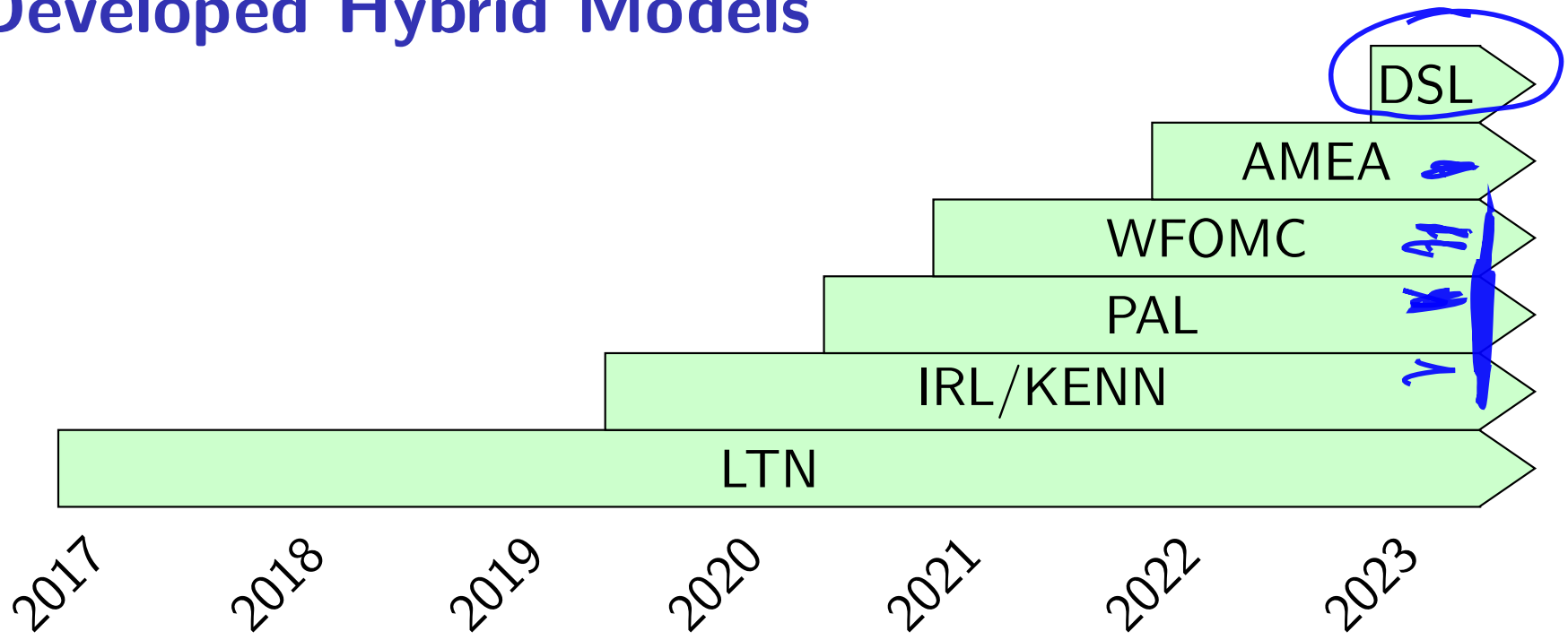
- design hybrid models;
- learning with hybrid models;
- inference with hybrid models.

Hybrid Model



- design hybrid models;
- learning with hybrid models;
- inference with hybrid models.

Developed Hybrid Models



LTN Logic Tensor Networks

IRL/KENN Incremental Refinemen Layer/Knowedge Enhanced Neural Netowrk

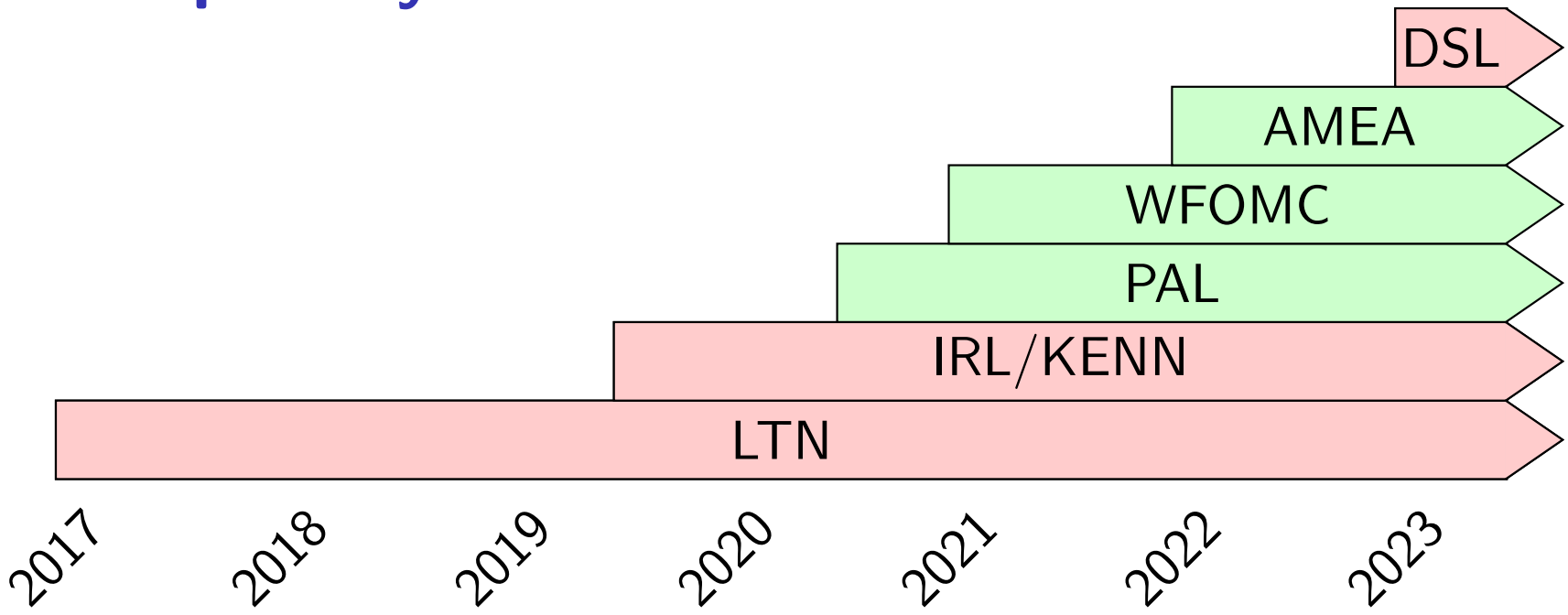
PAL Planning Acting and Learning

WFOMC Weighted First Order Model Counting

AMEAI Abstract Model for Embodied Agents

DSL Deep Symbolic Learning

Developed Hybrid Models



LTN Logic Tensor Networks

IRL/KENN Incremental Refinemen Layer/Knowedge Enhanced Neural Netowrk

PAL Planning Acting and Learning

WFOMC Weighted First Order Model Counting

AMEAI Abstract Model for Embodied Agents

DSL Deep Symbolic Learning

Three integration modalities

There is not a unique way to integrate sub-symbolic and symbolic logical knowledge in a unique hybrid model. Three examples are the following:

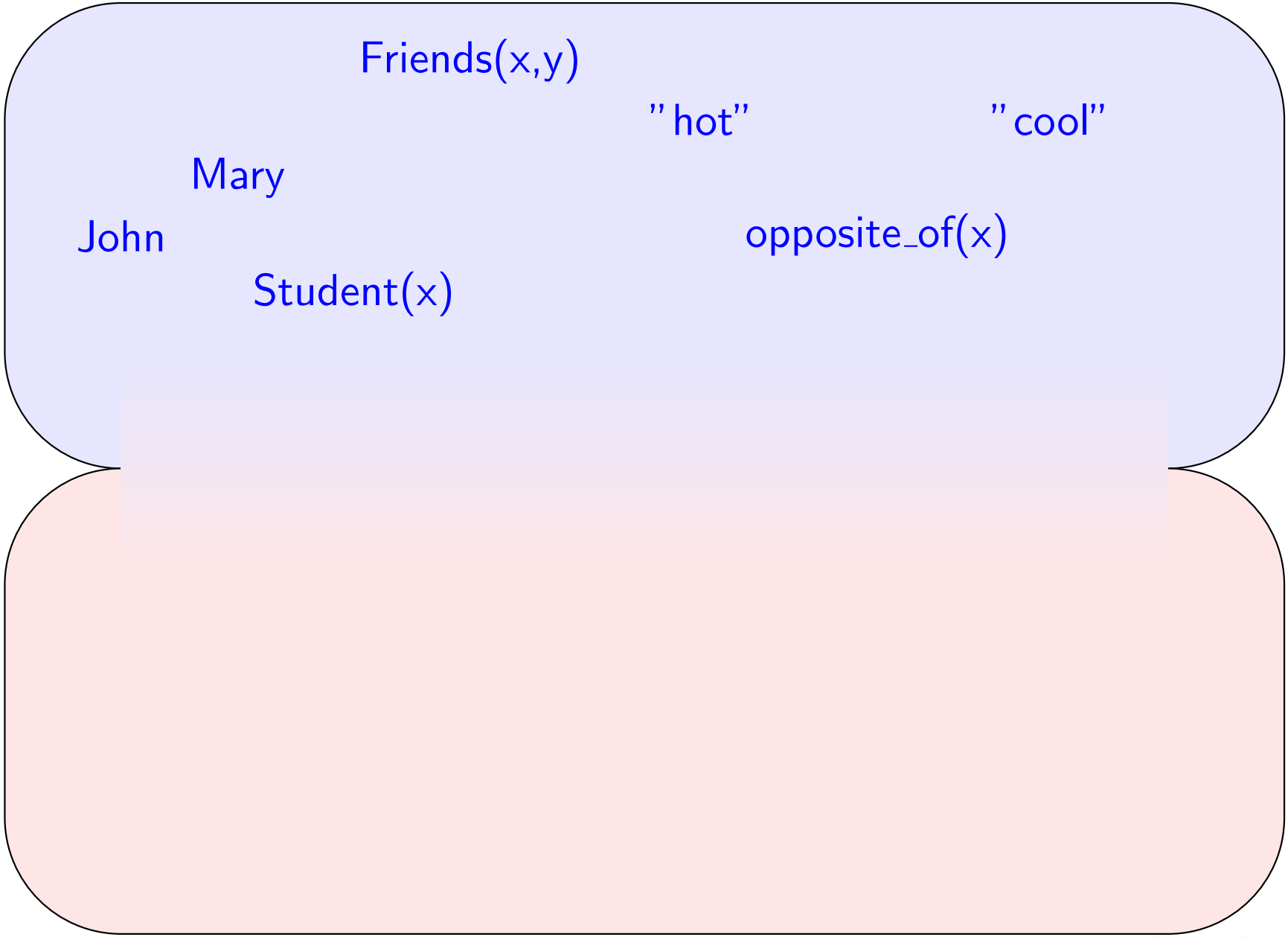
1. subsymbolic models provides the semantics of logical symbols (LTN)
2. A symbolic system is used to revise the predictions done by a sub-symbolic system (IRL)
3. the model is the composition of two function a sub-symbolic function and a symbolic function (DSL)

LTN: Logic Tensor Network

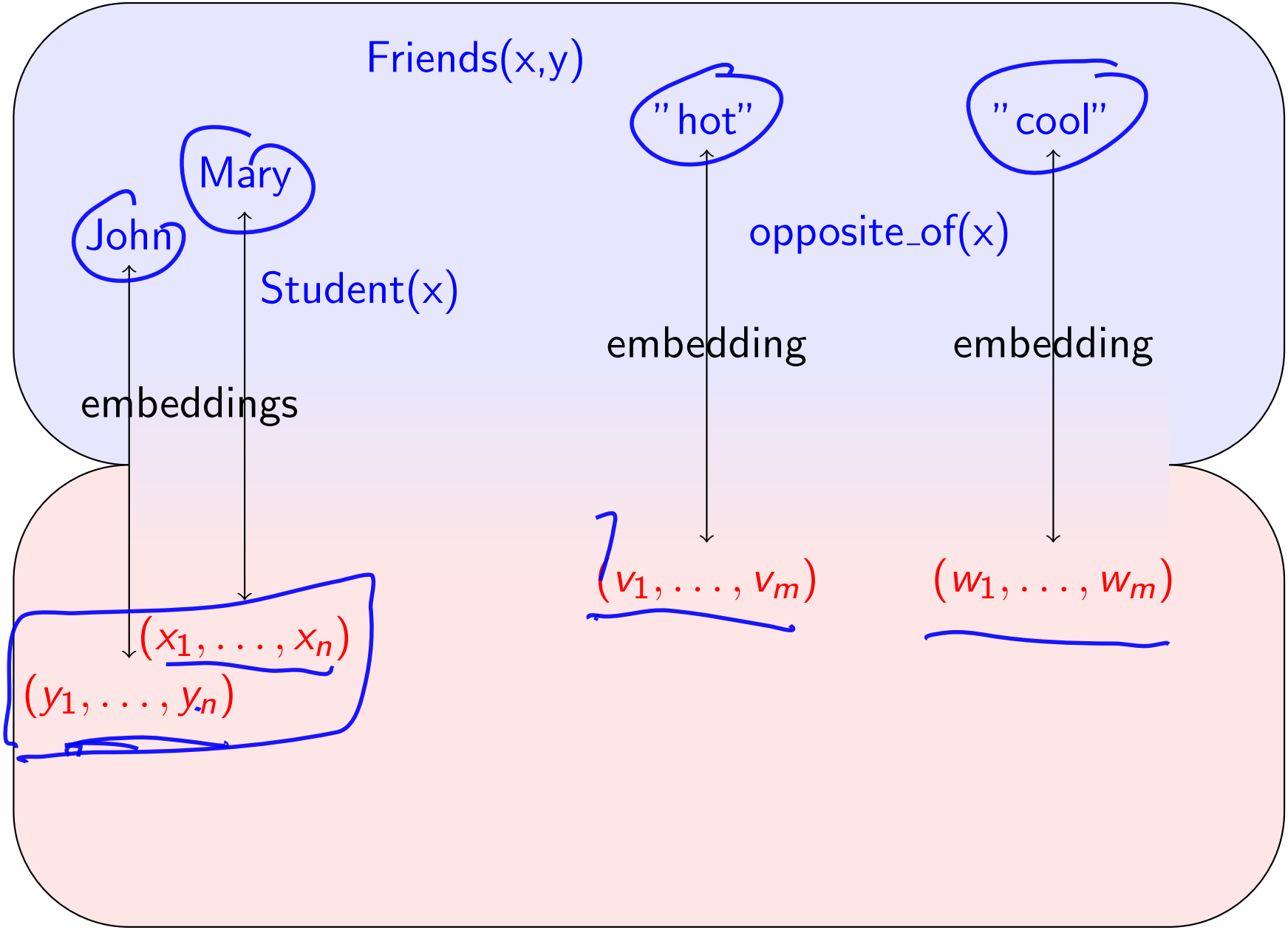
- inspired by the work by Gori and Diligenti on Semantic Regularization
- Logical symbols are grounded to real value functions (e.g., neural networks, embeddngs, generative models)
- view logical formula as constraints that need to be satisfied by a base neural network
- Training data and constraints are encoded in logical formula
- Logical formulas $\Rightarrow$ loss function
- Training $\Rightarrow$ maximum satisfiability of logical formulas

Samy Badreddine et al. "Logic tensor networks". In: *Artificial Intelligence* 303 (2022), p. 103649

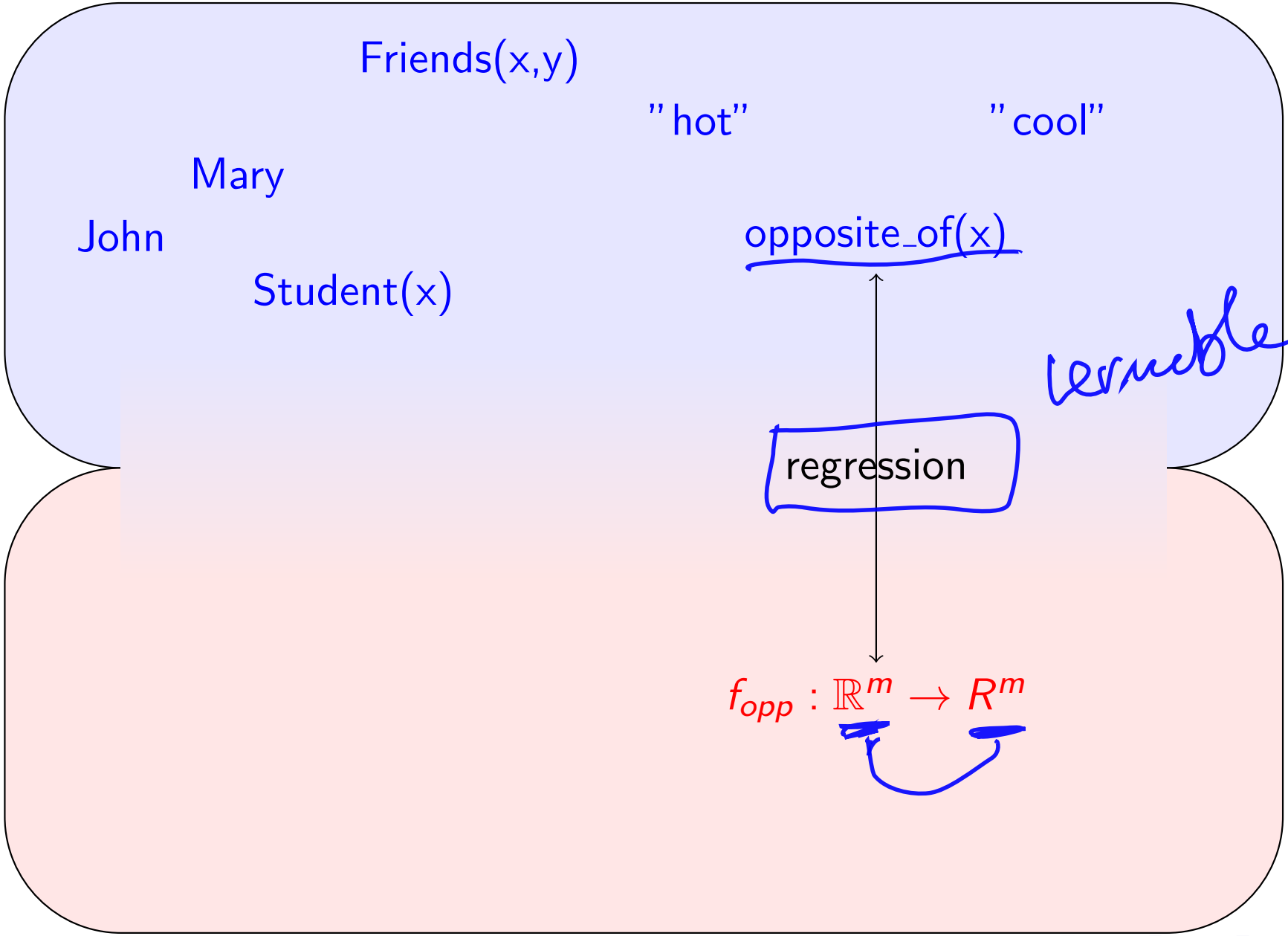
Grounding FO Signature to sub-symbolic models



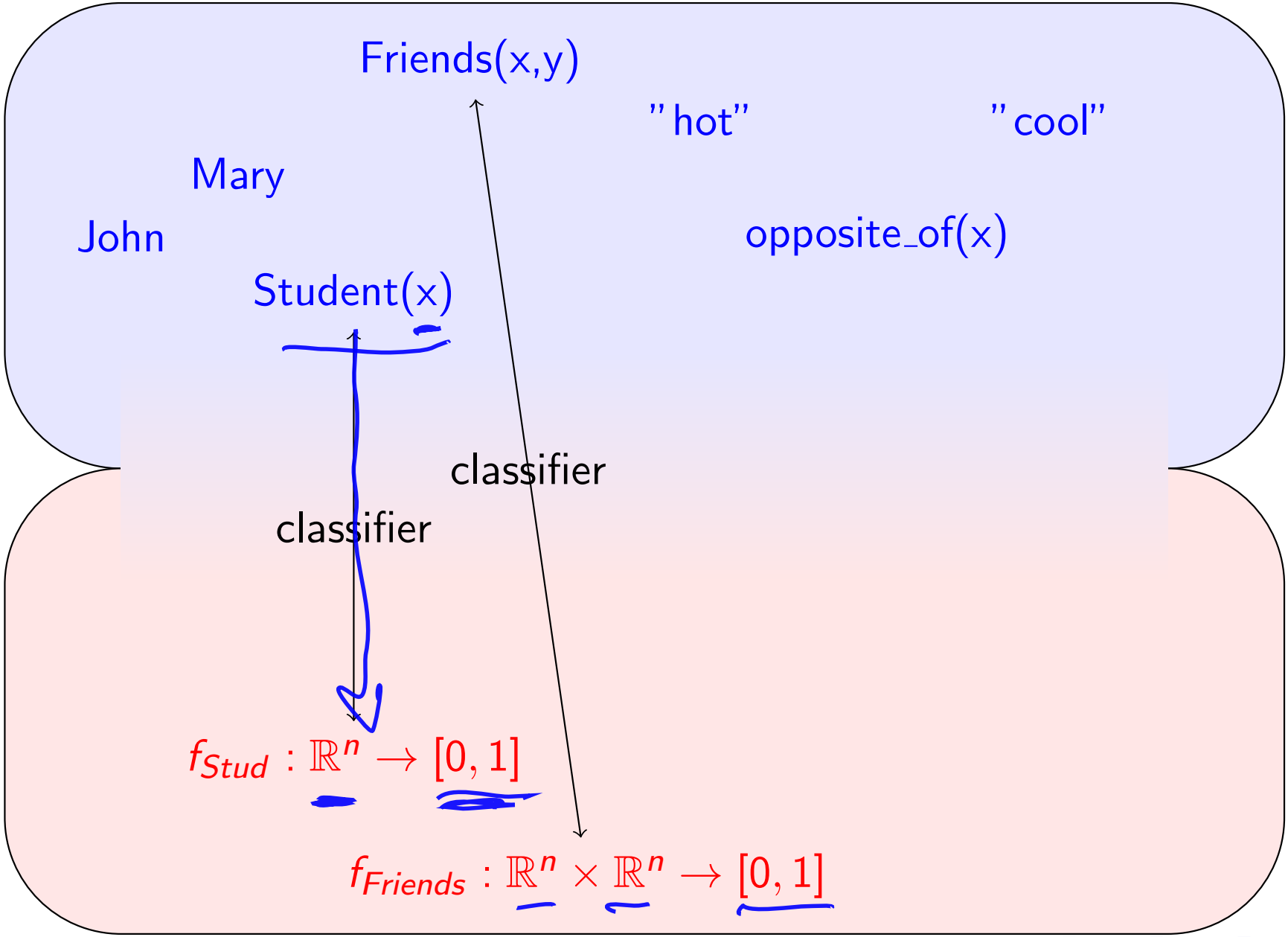
Grounding FO Signature to sub-symbolic models



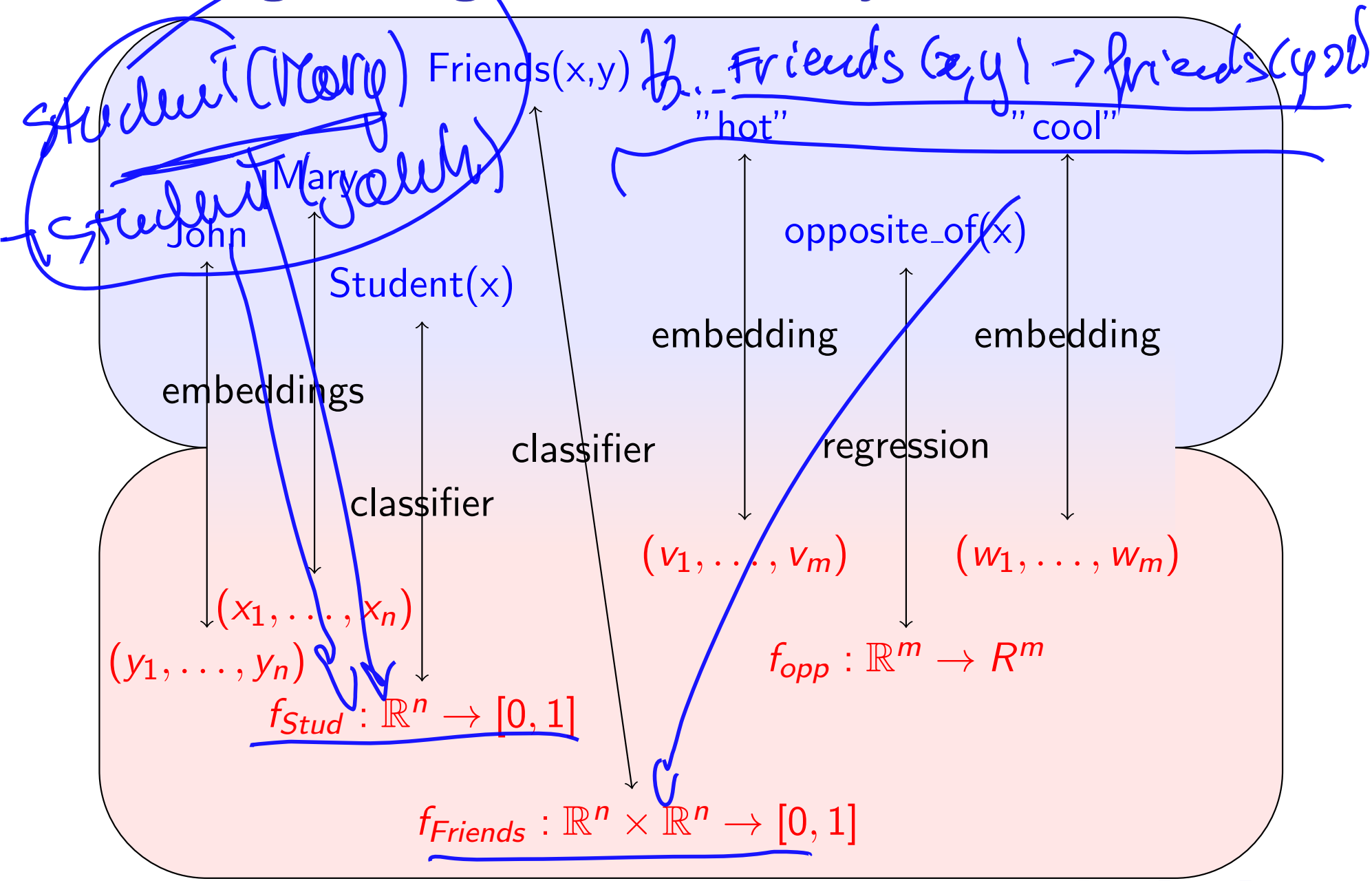
Grounding FO Signature to sub-symbolic models



Grounding FO Signature to sub-symbolic models



Grounding FO Signature to sub-symbolic models



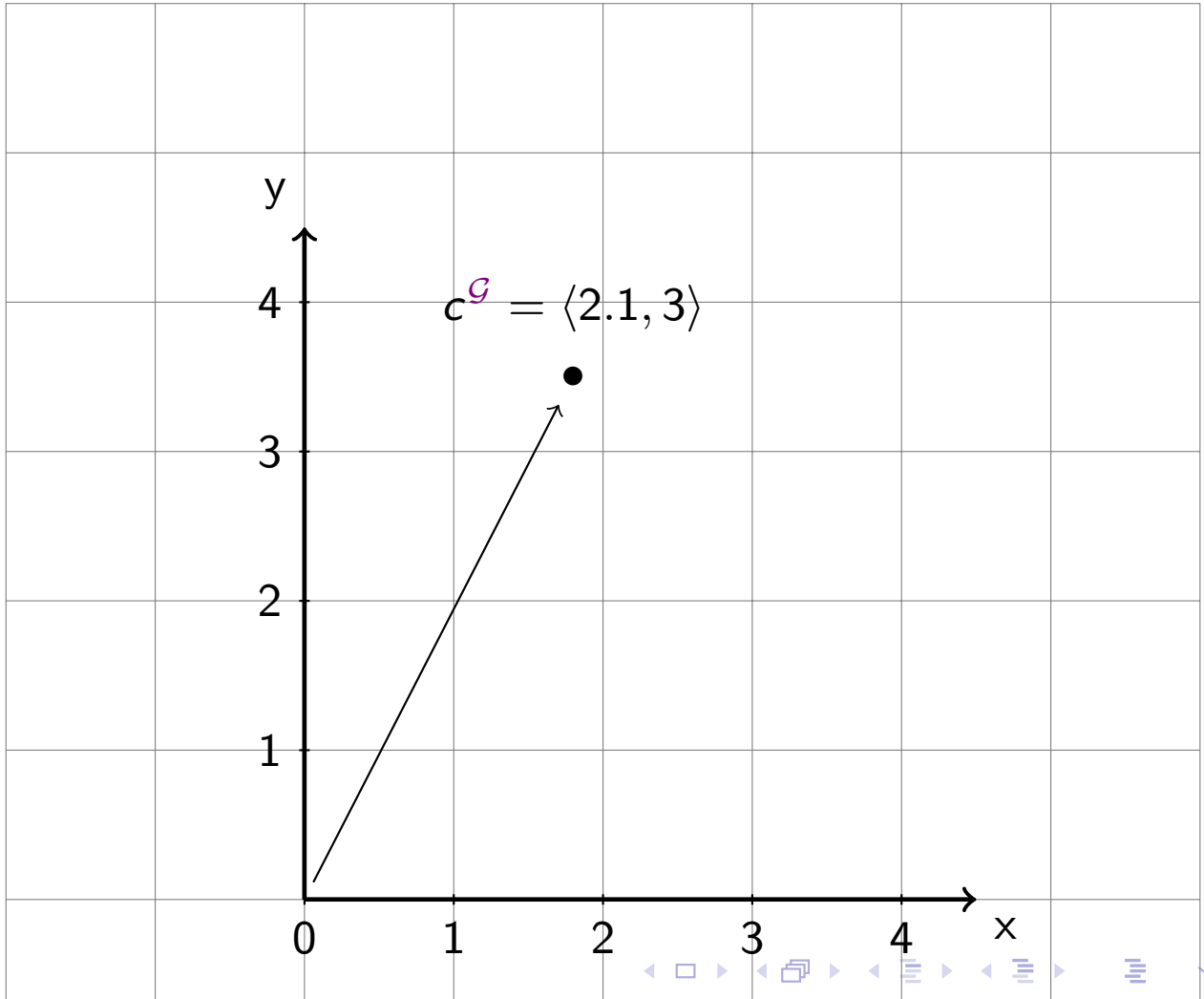
Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

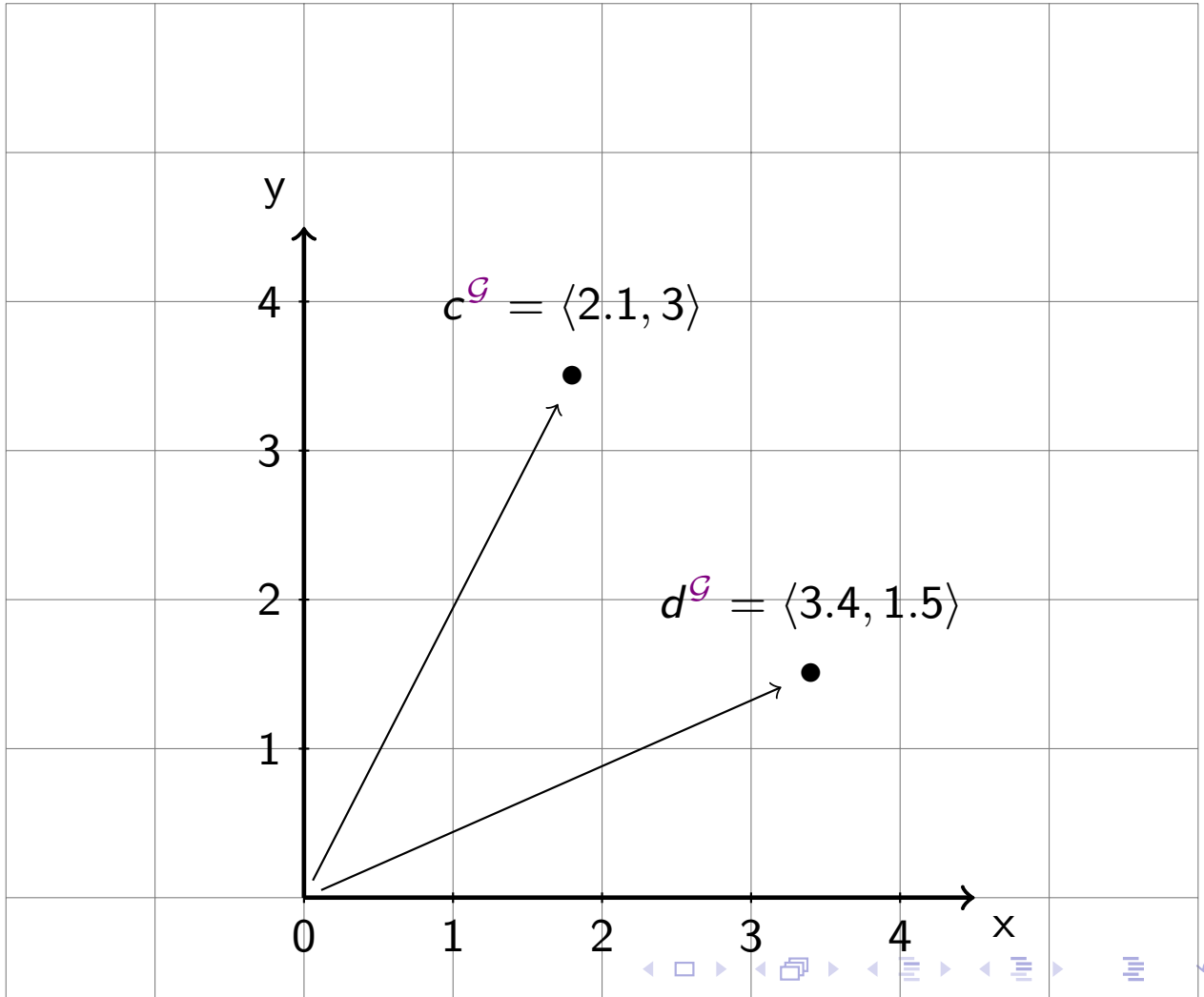
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

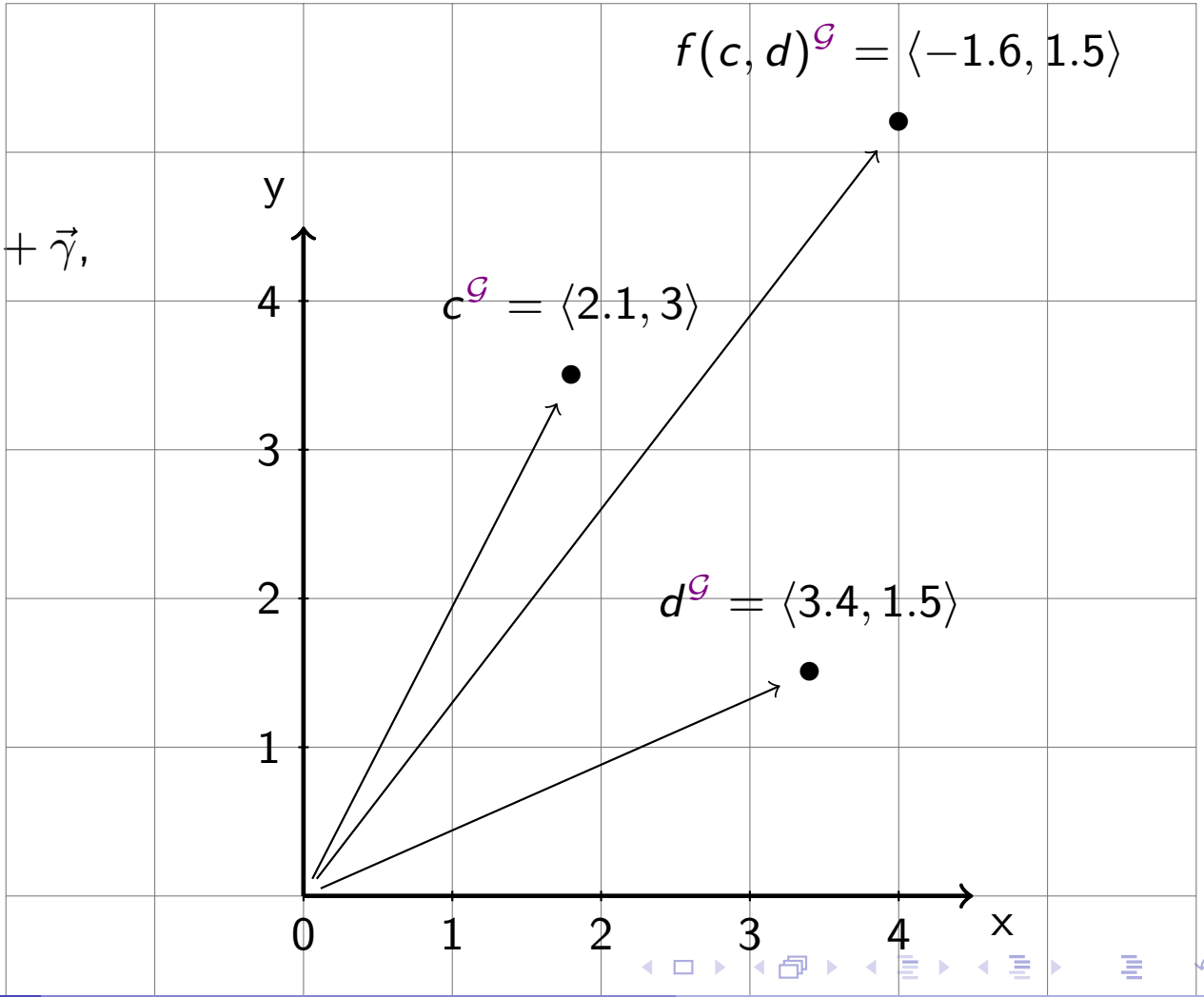
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

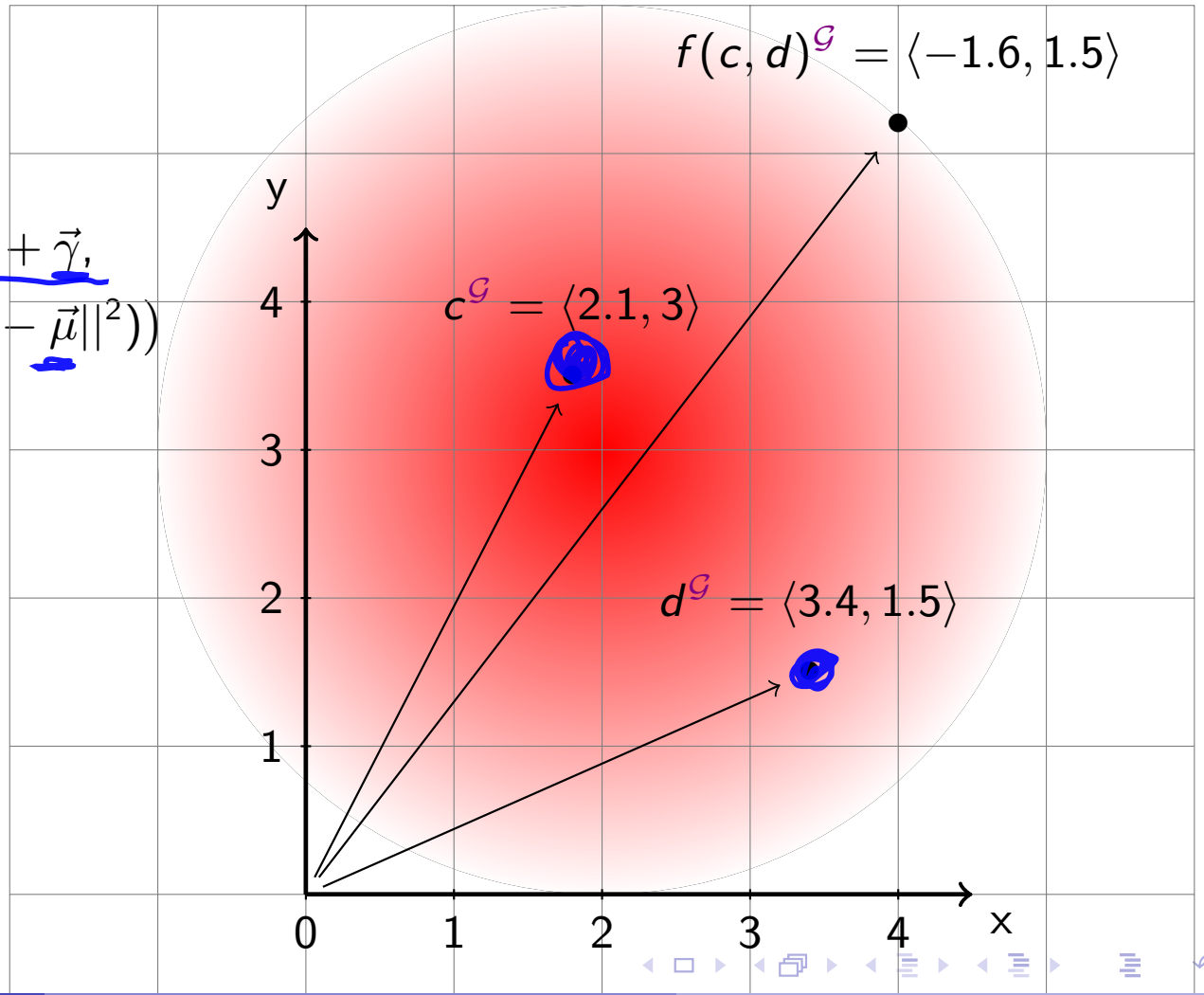
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \alpha\vec{x} + \beta\vec{y} + \vec{\gamma}$,



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

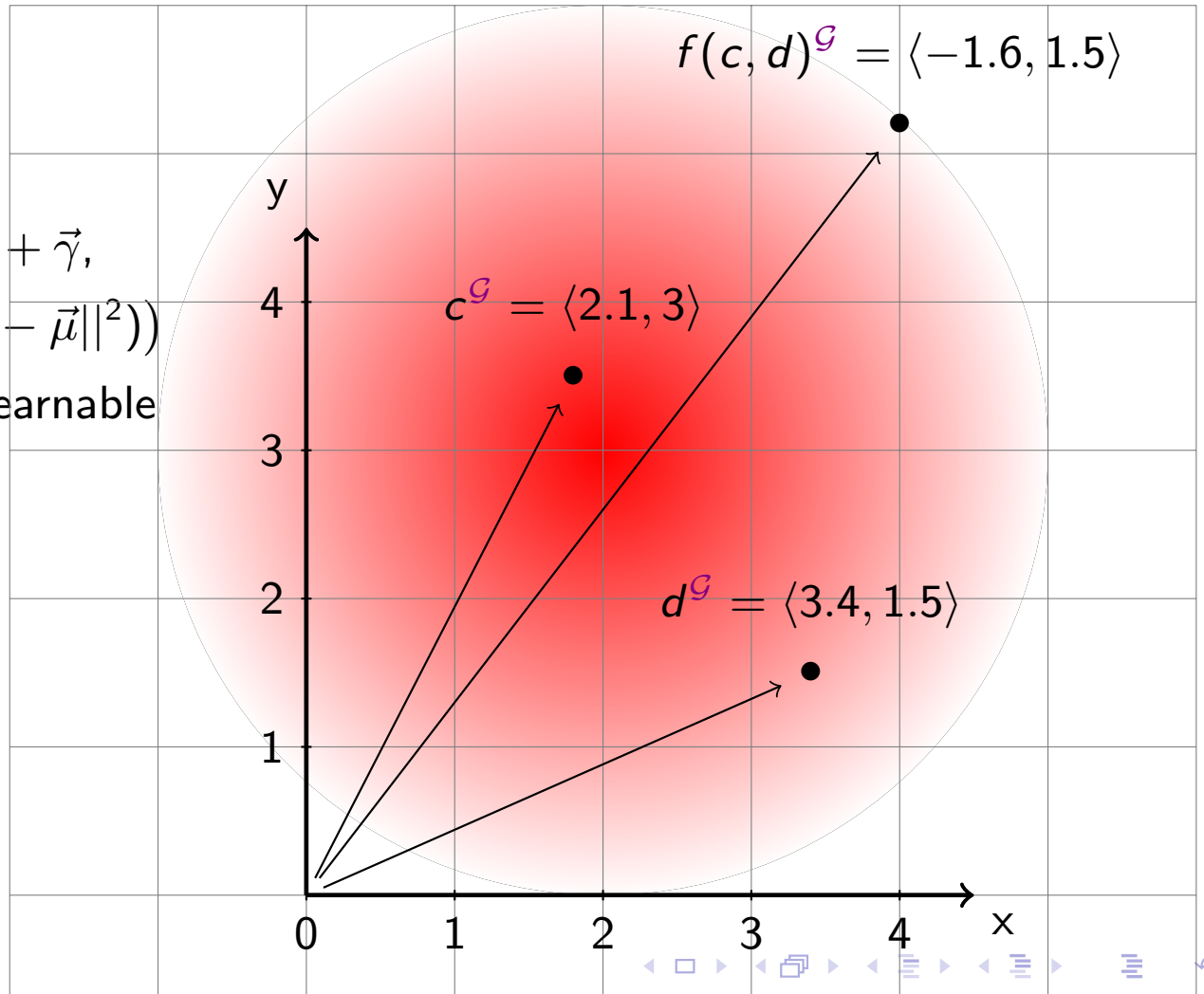
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \alpha\vec{x} + \beta\vec{y} + \vec{\gamma}$,
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

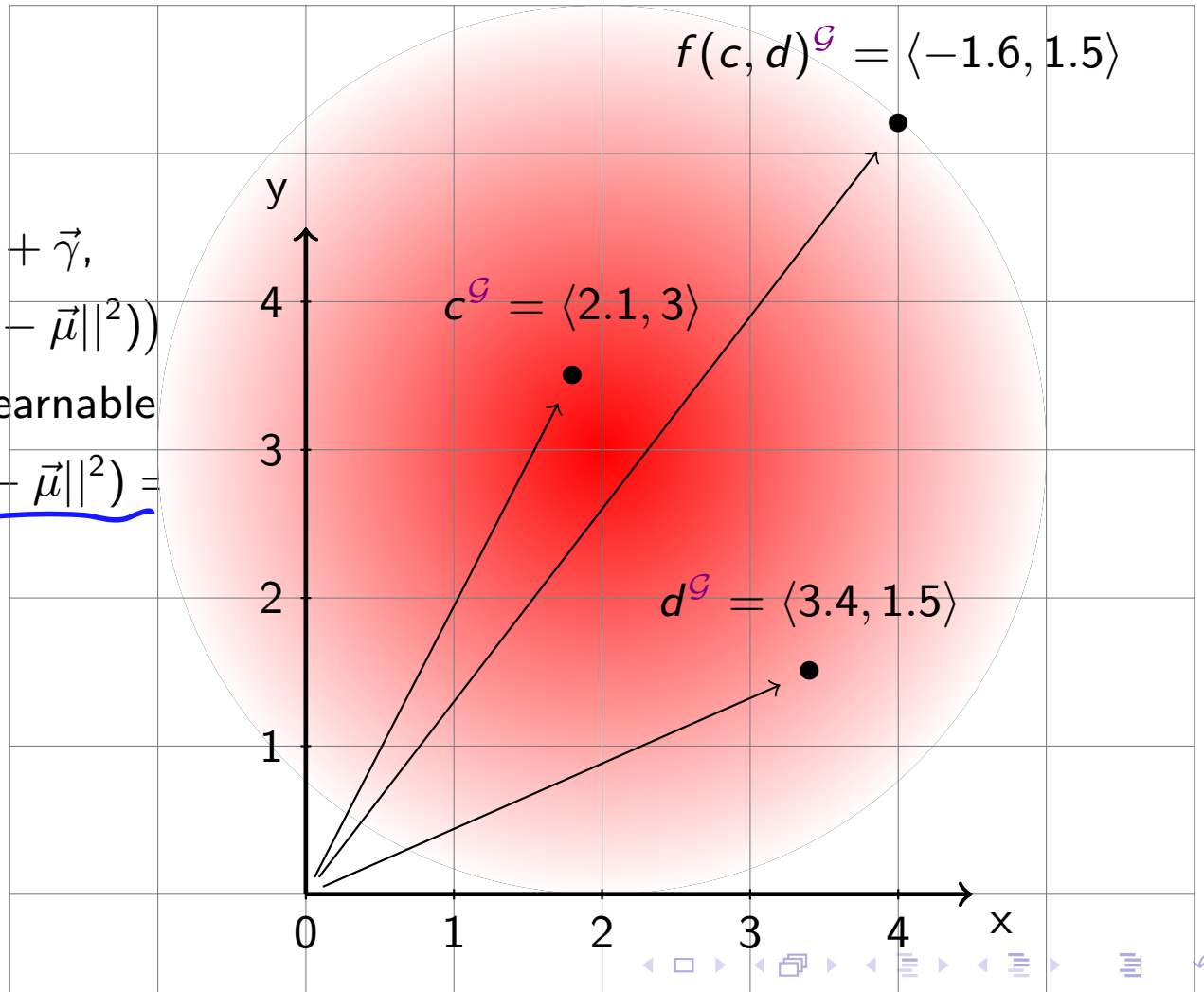
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \alpha\vec{x} + \beta\vec{y} + \vec{\gamma}$,
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$
- $\alpha, \beta, \gamma, \delta$ and μ are learnable



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

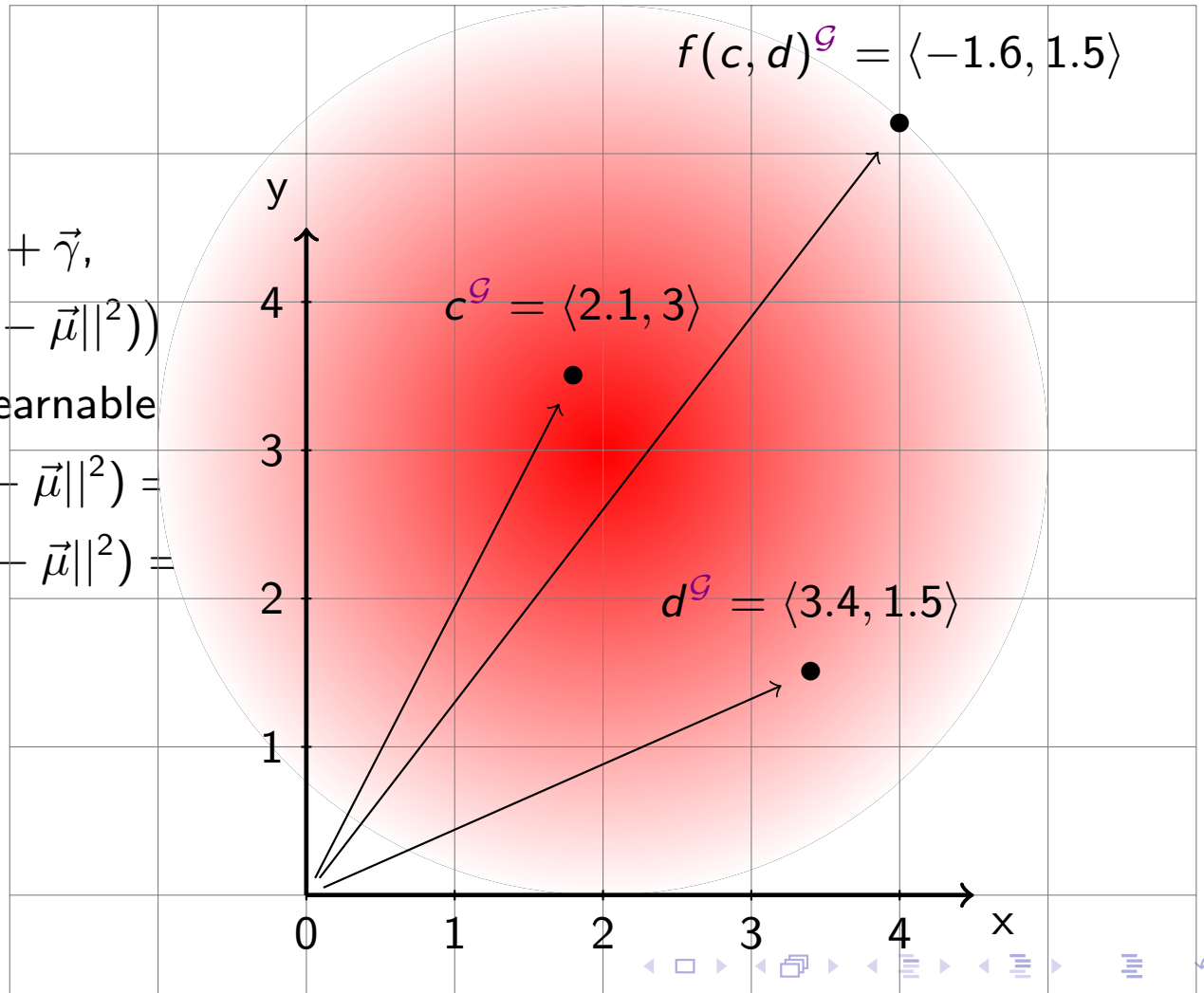
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \alpha\vec{x} + \beta\vec{y} + \vec{\gamma}$,
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$
- $\alpha, \beta, \gamma, \delta$ and μ are learnable
- $P(c)^{\mathcal{G}} = exp(-||c^{\mathcal{G}} - \vec{\mu}||^2) =$



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

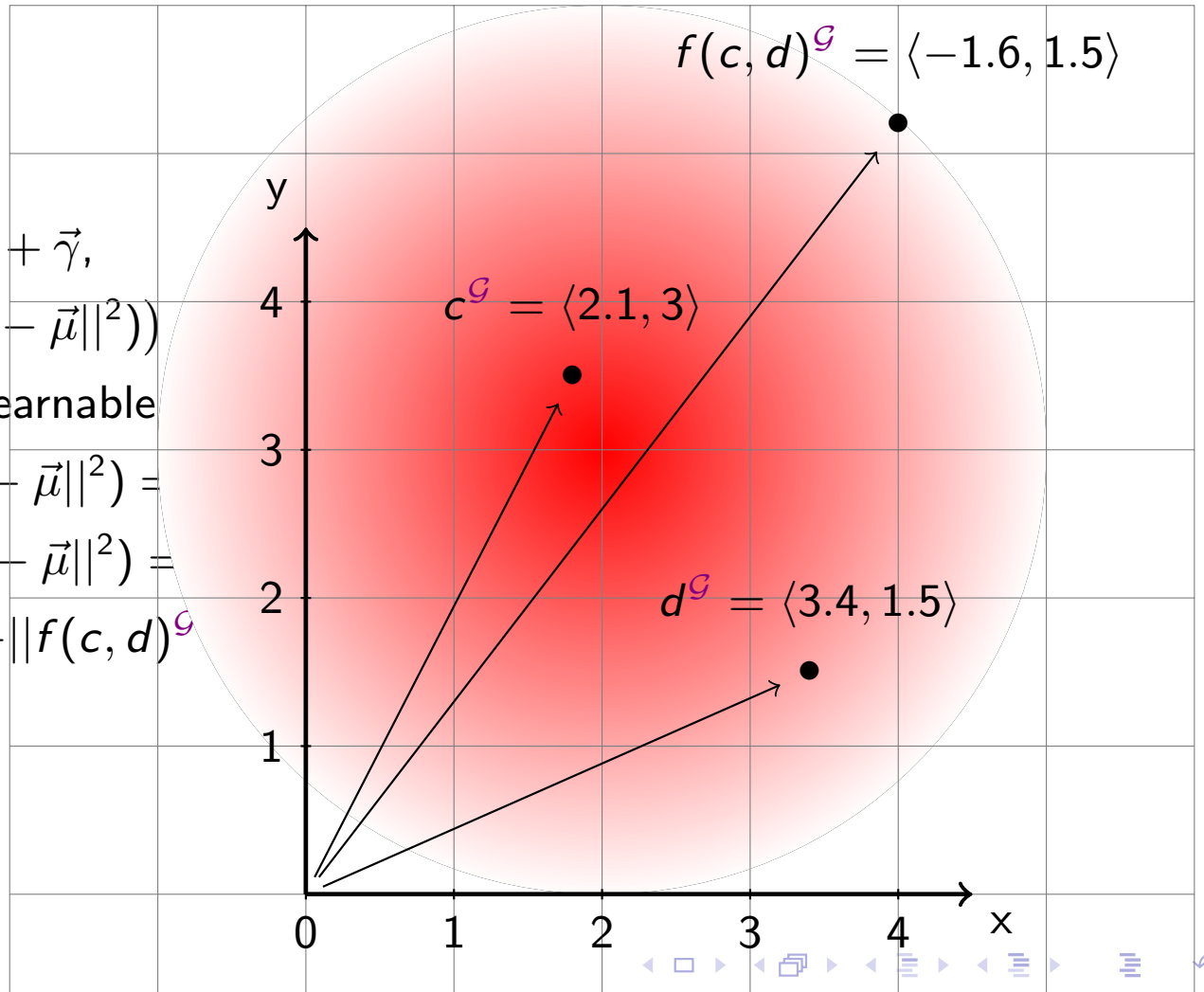
- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \alpha\vec{x} + \beta\vec{y} + \vec{\gamma}$,
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$
- $\alpha, \beta, \gamma, \delta$ and μ are learnable
- $P(c)^{\mathcal{G}} = exp(-||c^{\mathcal{G}} - \vec{\mu}||^2) =$
- $P(d)^{\mathcal{G}} = exp(-||d^{\mathcal{G}} - \vec{\mu}||^2) =$



Grounding FOL Signature in $\mathbb{R}^2$

$$\Sigma = (c, d, f, P) \tag{1}$$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \vec{\delta}$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \alpha\vec{x} + \beta\vec{y} + \vec{\gamma}$,
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$
- $\alpha, \beta, \gamma, \delta$ and μ are learnable
- $P(c)^{\mathcal{G}} = exp(-||c^{\mathcal{G}} - \vec{\mu}||^2) =$
- $P(d)^{\mathcal{G}} = exp(-||d^{\mathcal{G}} - \vec{\mu}||^2) =$
- $P(f(c,d))^{\mathcal{G}} = exp(-||f(c,d)^{\mathcal{G}}$
  $1.95 * 10^{-6}$



Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$
- $(\phi \rightarrow \psi)^{\mathcal{G}} = min(1 - \phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$
- $(\phi \rightarrow \psi)^{\mathcal{G}} = min(1 - \phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$
- $(\phi \vee \psi)^{\mathcal{G}} = \min(\phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$
- $(\phi \rightarrow \psi)^{\mathcal{G}} = min(1 - \phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$
- $(\phi \vee \psi)^{\mathcal{G}} = \min(\phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$
- $(\neg\phi)^{\mathcal{G}} = 1 - \phi^{\mathcal{G}}$

Alternative fuzzy operators

Examples:

$$(\phi \wedge \psi)^{\mathcal{G}} = \min(\phi^{\mathcal{G}}, \psi^{\mathcal{G}}) \qquad \text{(Gödel T-norm)}$$

$$(\phi \wedge \psi)^{\mathcal{G}} = \phi^{\mathcal{G}} \cdot \psi^{\mathcal{G}} \qquad \text{(Product T-norm)}$$

$$\ldots$$

Grounding FOL quantifiers

- The semantics of $\forall x \phi(x)$ and $\exists x \phi(x)$ is given in terms of aggregation functions $Agg : [0,1]^* \rightarrow [0,1]$
- Every individual variable x is grounded in a m-tuple $(x^{(1)}, \ldots, x^{(m)})$ of elements in $\mathbb{R}^n$
- $x^{\mathcal{G}}$ is obtained by sampling from $\mathbb{R}^n$ according to some known data distribution.

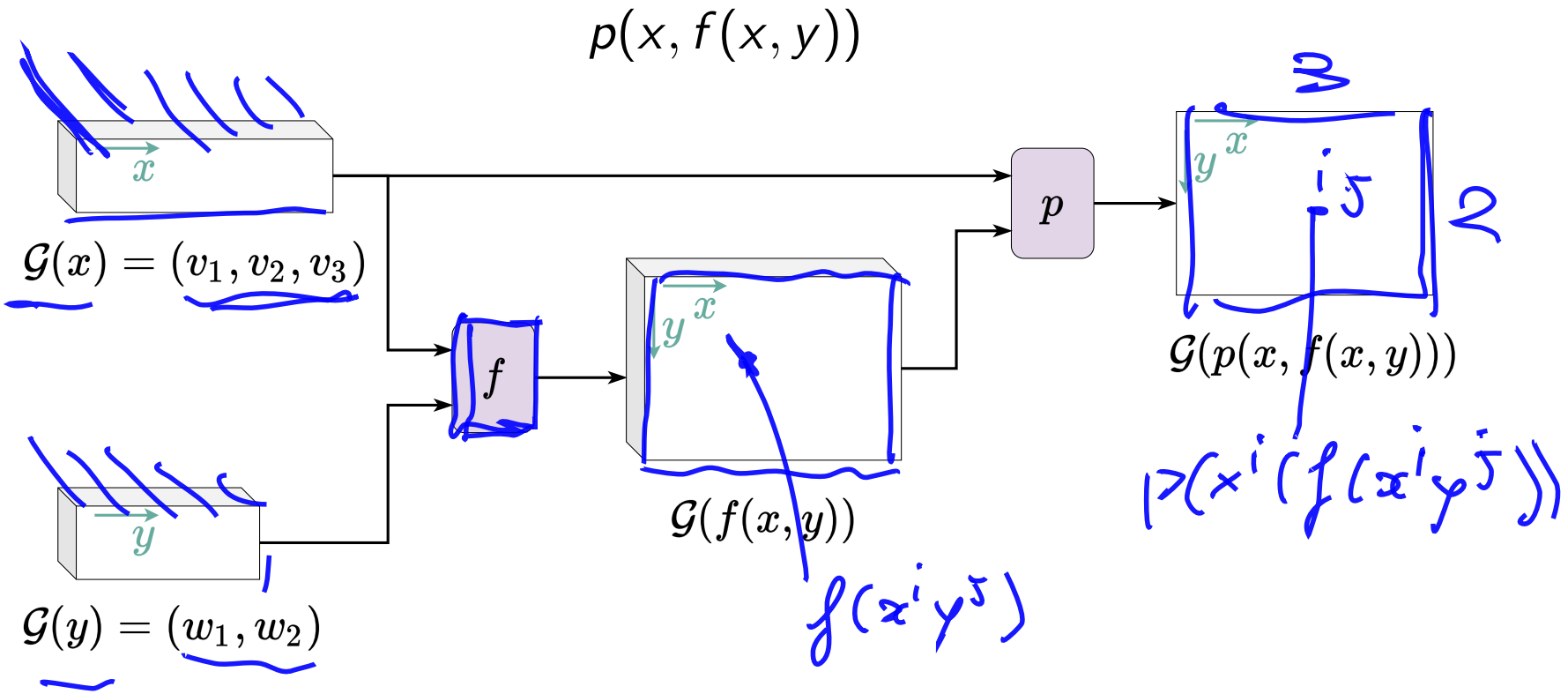
Standard fuzzy semantics for quantifiers

$$(\forall x \phi(x))^{\mathcal{G}} = \min_{i=1}^{m} \phi(x^{(i)})^{\mathcal{G}} \qquad (\exists x \phi(x))^{\mathcal{G}} = \max_{i=1}^{m} \phi(x^{(i)})^{\mathcal{G}}$$

Alternative parametrized semantics

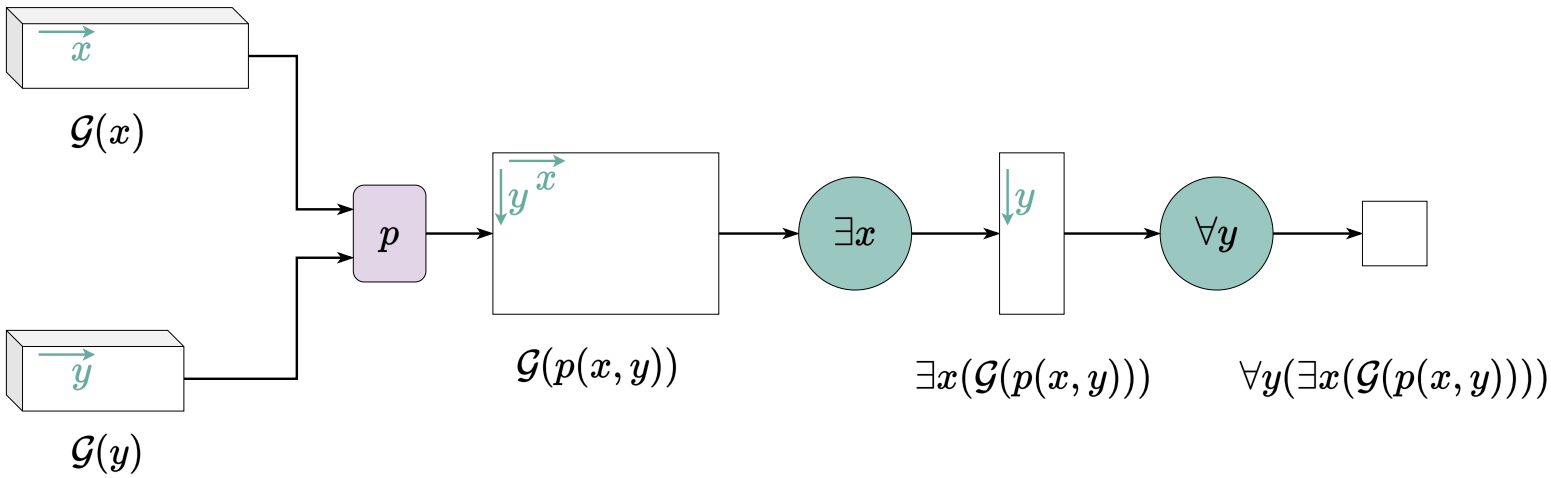
$$(Qx \phi(x))^{\mathcal{G}} = \left(\frac{1}{m} \sum_{i=1}^{m} (\phi(x^{(i)})^{\mathcal{G}})^p \right)^{\frac{1}{p}}$$

Computation graph of a formula ϕ



Computation graph of a formula ϕ

$$\forall y \exists x P(x, y)$$



Usage of LTN

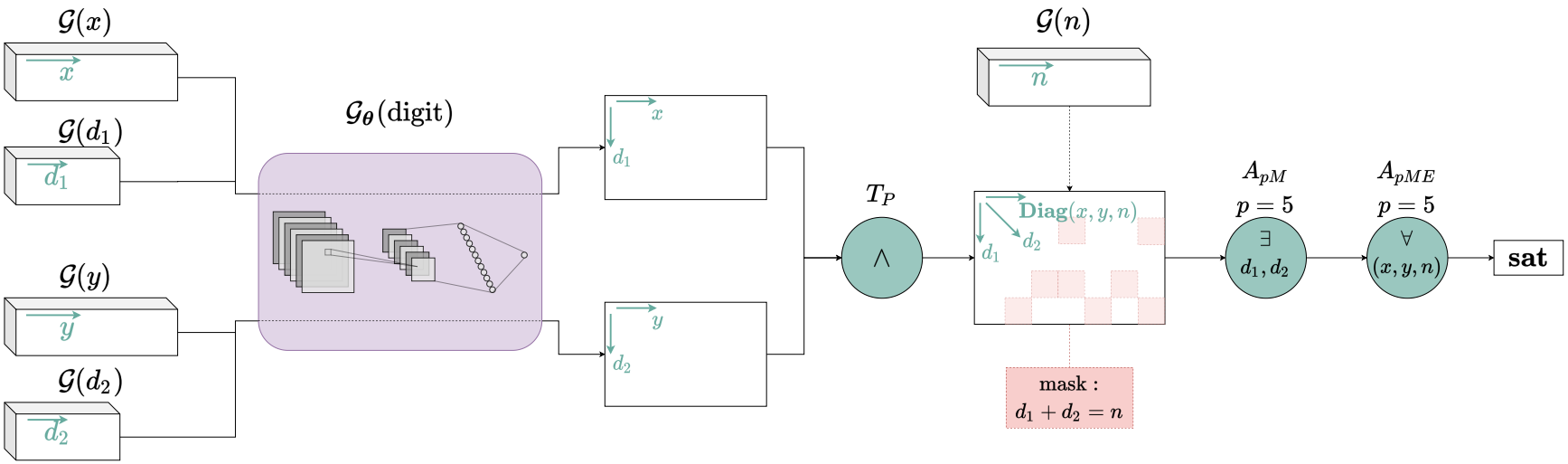
- specify your learning problem in terms of $\phi_1, \ldots, \phi_n$
- Create the computational circuits for $\phi_1, \ldots, \phi_n$;
- train this circuit by minimizing the loss function

$$1 - Agg(\phi_1^{\mathcal{G}}, \ldots, \phi_n^{\mathcal{G}})$$

where Agg is some aggregation operator for truth values.

Mnist sum

$$\forall x, y, n(\exists d_1, d_2((d_1 + d_2 = n) \wedge digit(x, d_1) \wedge digit(y, d_2)))$$

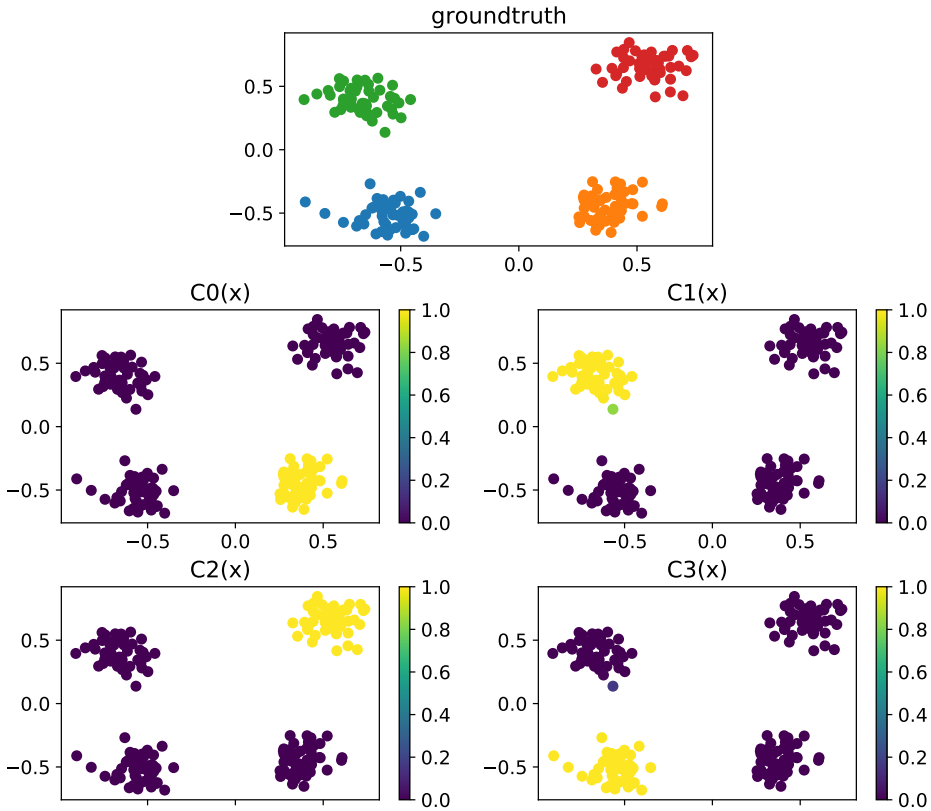


Custering

$$\forall x \exists c C(x,c) \qquad \forall c \exists x C(x,c)$$

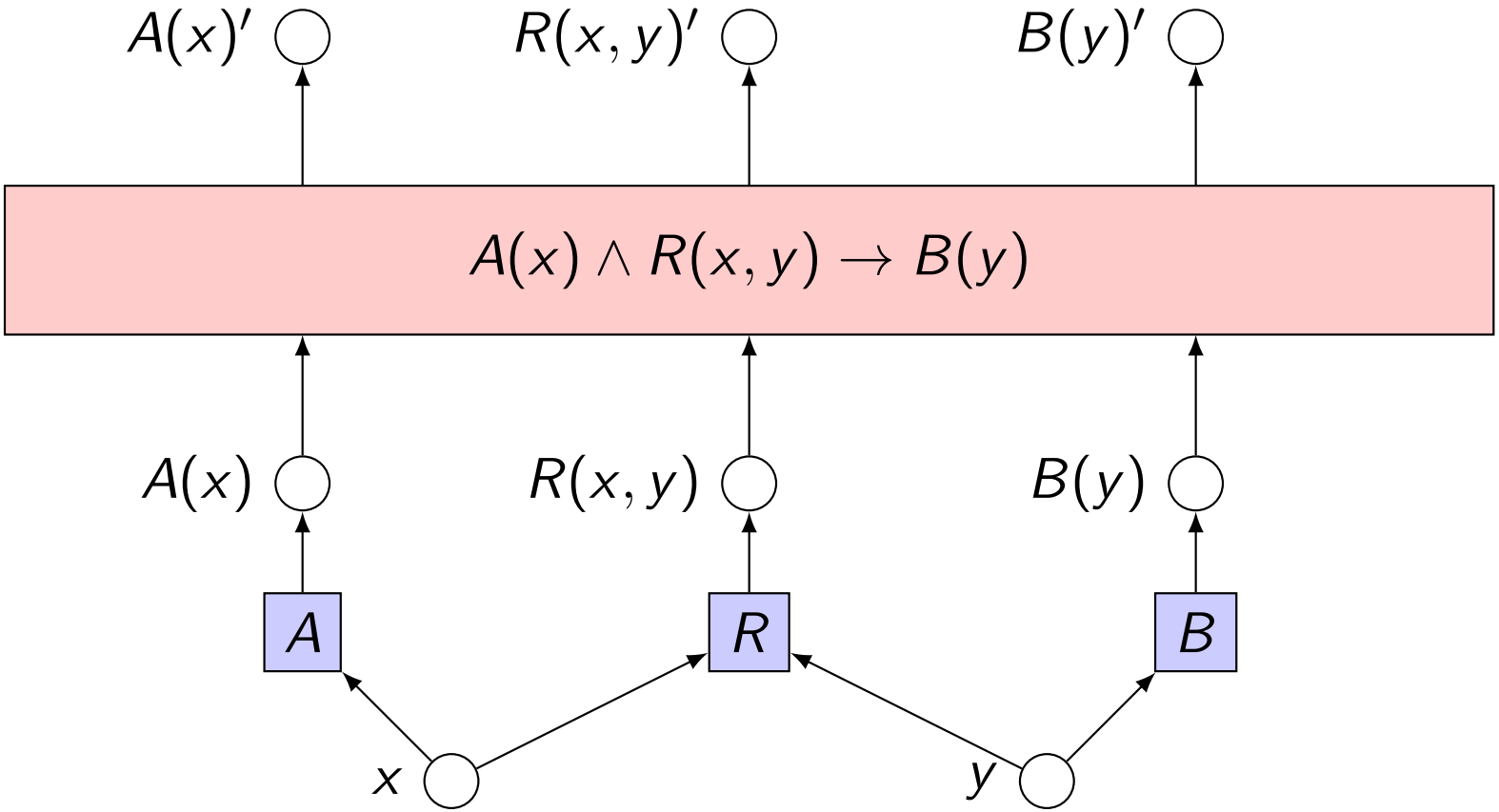
$$\forall x,y,c |x-y| < T_{close} : (C(x,c) \leftrightarrow C(y,c))$$

$$\forall x,y,c |x-y| > T_{dist} : \neg(C(x,c) \wedge C(y,c))$$



IRL: Incremental Refinement layer

- In IRL the logical formulas (in clausal form) is like any other layer of a neural network. It takes in imput the predictions of the predicates and return a modification of the prediction that better fits the formula



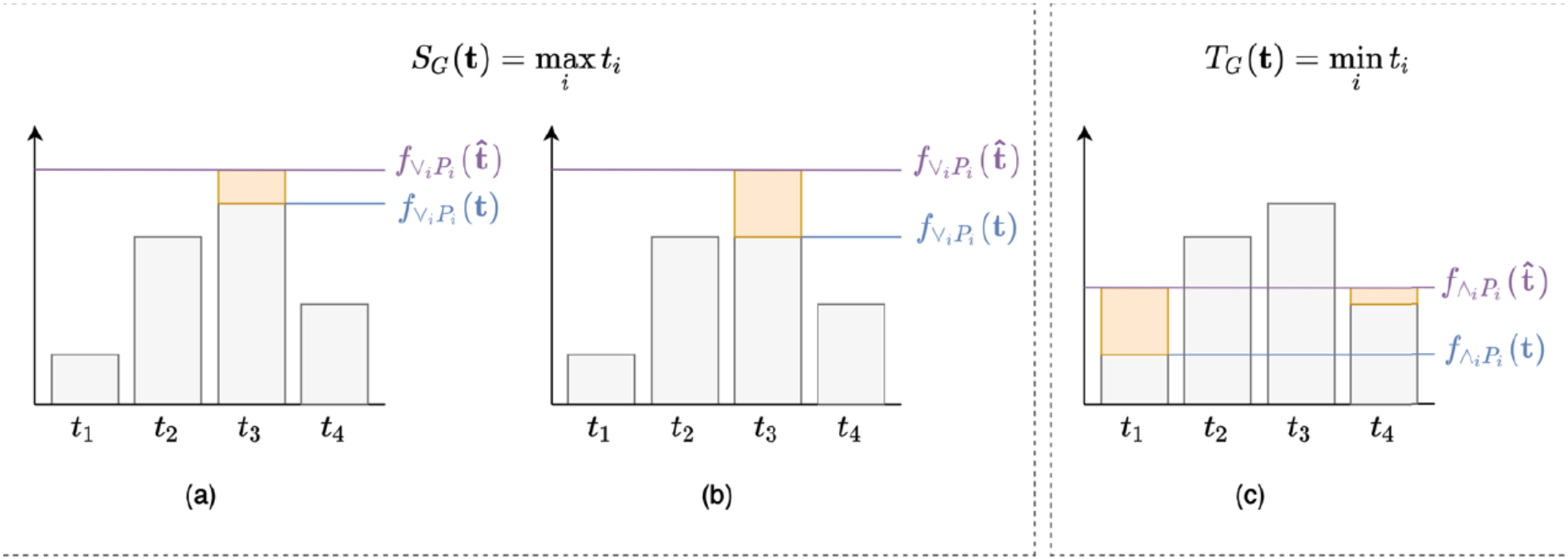
Alessandro Daniele et al. "Refining neural network predictions using background knowledge". In: *Machine Learning* (2023), pp. 1–39

Definition (Minimal Refinement function)

Let $C = (l_1, \ldots, l_n)$ be a clause. A minimal refinement function for a value $w \in [0, 1]$ is a function $f : [0.1]^n \to [0.1]^n$ such that

- $C(f(t_1, \ldots, t_n)) = w$
- every other $f : [0, 1]^n \to [0, 1]^n$ such that $C(f'(t_1 \ldots, t_n)) = w$

$$\|f(\boldsymbol{t}) - \boldsymbol{t}\| \leq \|f'(\boldsymbol{t}) - \boldsymbol{t}\|$$

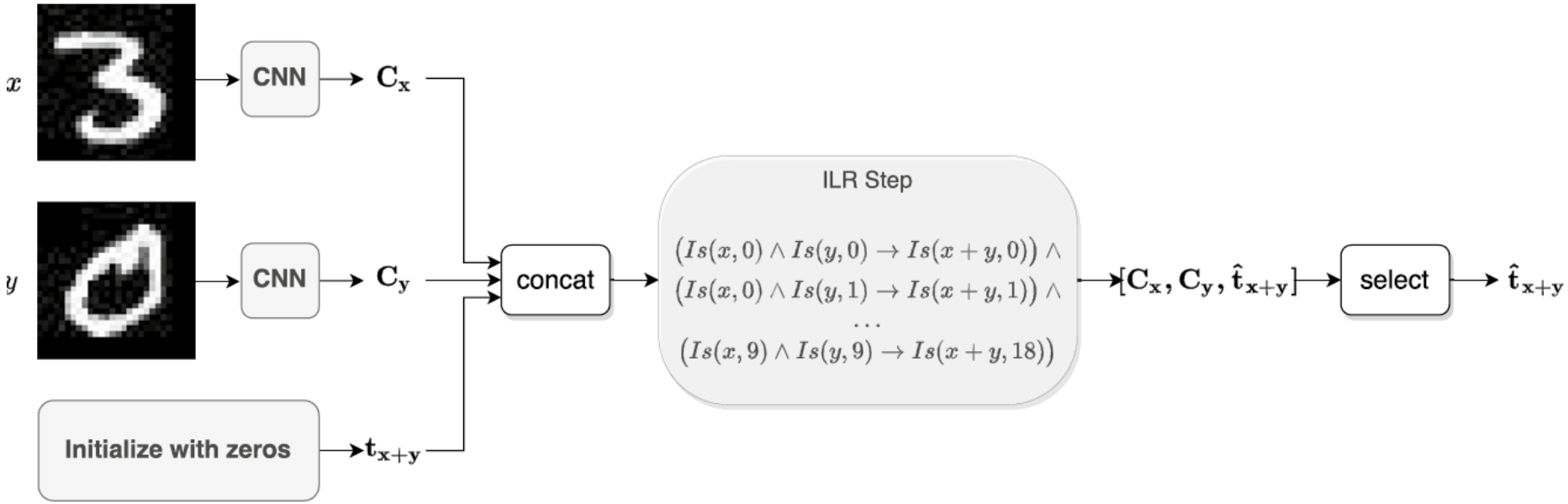


Minimal Refinement function - General case

- refinement functions are defined for other operators ($\wedge$,$\vee$,...)
- Refinement functions can be defined for every formula as a composition of the refinement functions for each single operator.
- Analitic form for refinement function has been provided for an entire classes of fuzzy semantics that includes Product, Gödel, and Łukasiewicz.

Daniele et al., "Refining neural network predictions using background knowledge"

Mnist Sum


x
CNN
$\mathbf{C_x}$
y
CNN
$\mathbf{C_y}$
Initialize with zeros
$\mathbf{t_{x+y}}$
concat
ILR Step
$(Is(x,0) \wedge Is(y,0) \rightarrow Is(x+y,0)) \wedge$
$(Is(x,0) \wedge Is(y,1) \rightarrow Is(x+y,1)) \wedge$
$\ldots$
$(Is(x,9) \wedge Is(y,9) \rightarrow Is(x+y,18))$
$[\mathbf{C_x}, \mathbf{C_y}, \hat{\mathbf{t}}_{\mathbf{x+y}}]$
select
$\hat{\mathbf{t}}_{\mathbf{x+y}}$

Deep Symbolic Learning

Learn end to end the composition of two functions

$$\mathsf{f} \circ \mathsf{g}$$

$$g(f_1(\boldsymbol{x}), f_2(\boldsymbol{y}))$$

- f is the perception function

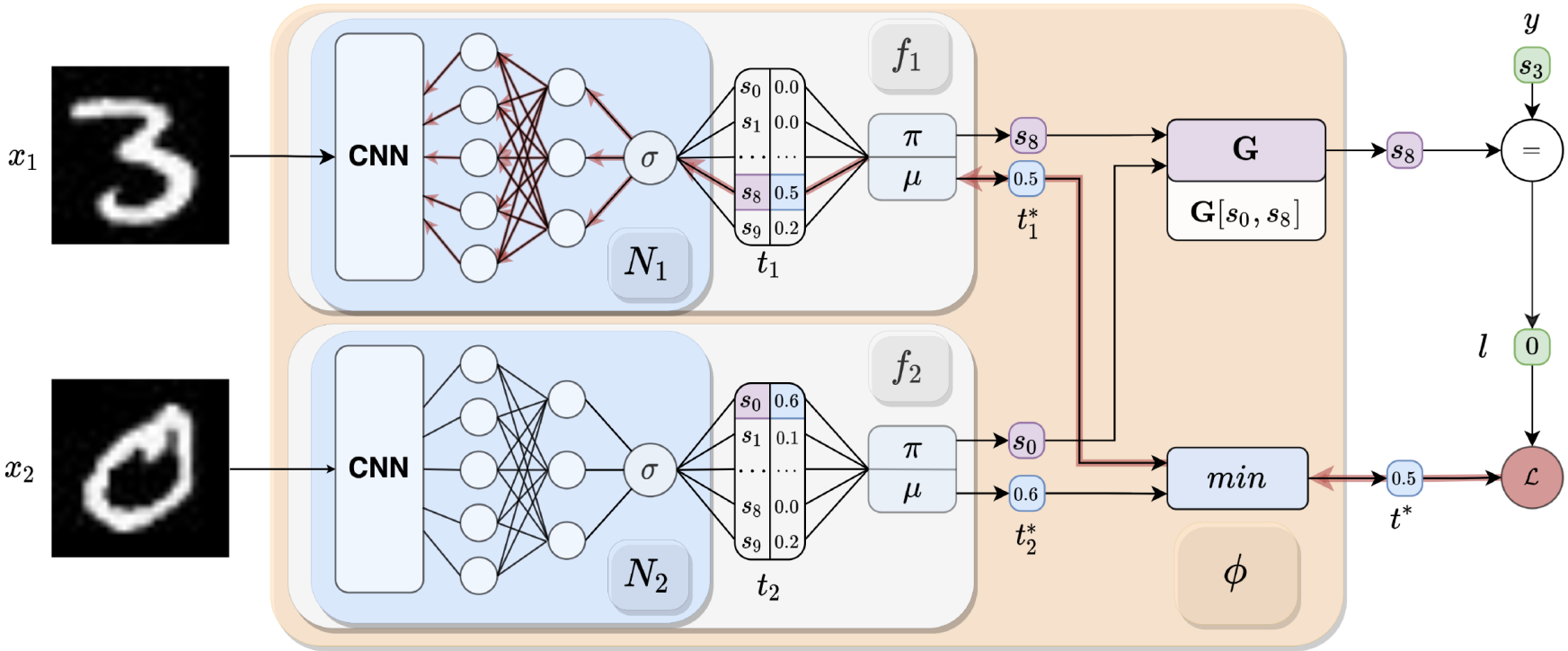
$$f : R^n \longrightarrow \{1, \ldots, m\}$$

- g is the discrete function

$$g : \{1, \ldots, m\}^k \longrightarrow \{1, \ldots, p\}$$

Alessandro Daniele et al. "Deep Symbolic Learning: Discovering Symbols and Rules from Perceptions". In: *arXiv preprint arXiv:2208.11561* (2022). To appear in IJCAI 2023

Deep Symbolic Learning



Conclusiont

- There is not a unique way to combine logic and neural networks Frank Van Harmelen and Annette ten Teije. "A boxology of design patterns for hybrid learning and reasoning systems". In: *arXiv preprint arXiv:1905.12389* (2019)
- I strongly believe that integrative approaches are better than encoding logic into non symbolic systems
- In this talk we have seen different approaches that have been used to solve the same class of problems (with similar results)
- As NeSy community we should develop benchmarks to evaluate and compare different approaches