Symbolic Knowledge Injection via PSyKI

A tutorial

Giovanni Ciatto
giovanni.ciatto@unibo.it

Matteo Magnini
matteo.magnini@unibo.it

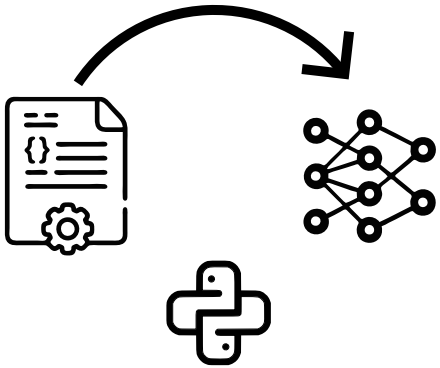
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna, Cesena, Italy

24^{th} International Conference on
Principles and Practice of Multi-Agent Systems
November 16, 2022

Next in Line...

What

PSyKI: a (Python) platform for symbolic knowledge injection



GitHub Repository

https://github.com/psykei/psyki-python
(please star us :)

Main papers

- [Magnini et al., 2022b]

Why SKI?

There are several benefits:

- prevent the predictor to become a black-box!;
- reduce learning time;
- reduce the data size needed for training;
- improve predictor's accuracy;
- build a predictor that behave as a logic engine.

Next in Line...

Symbolic Knowledge Injection I

Key insights:

- Altering ML predictors...
- ...to make they comply to user-provided knowledge...
- ...which is represented in symbolic form

Symbolic Knowledge Injection II

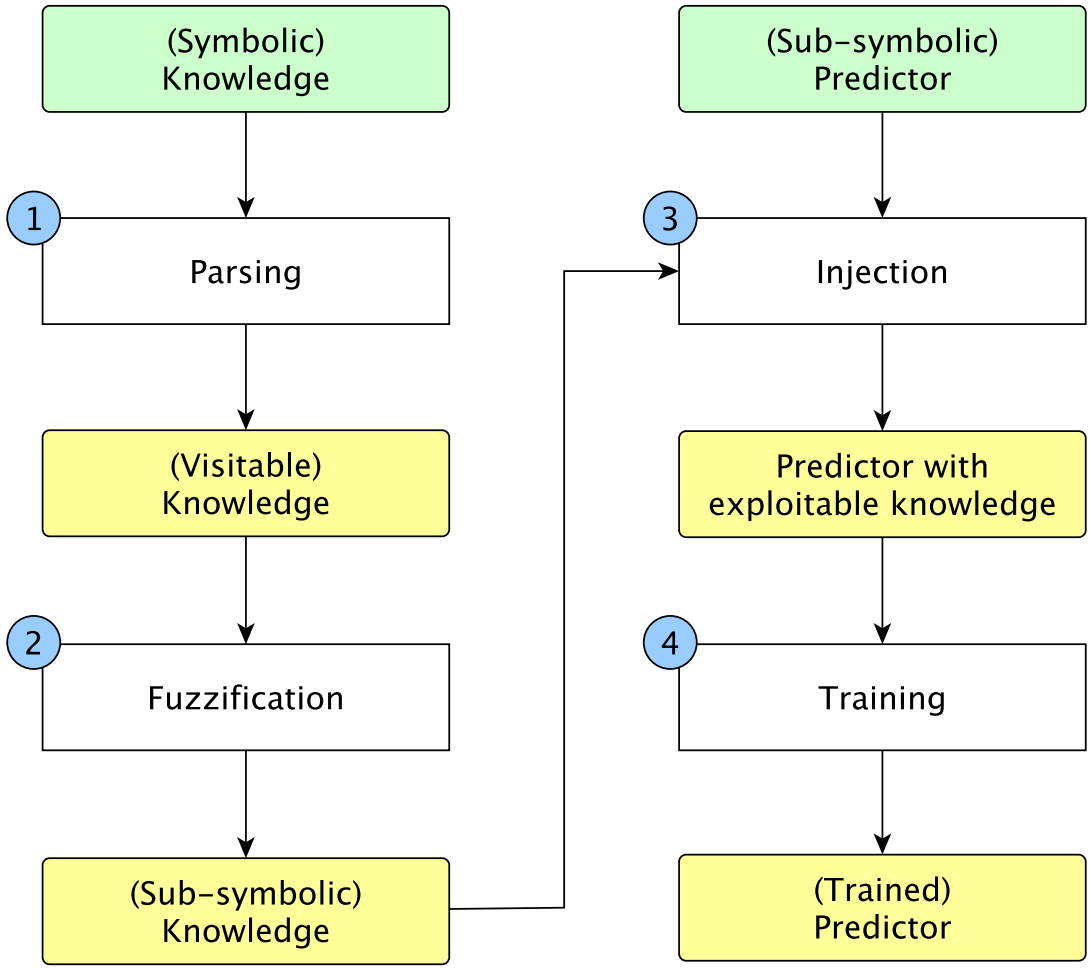
We define SKI as:

any algorithmic procedure affecting how sub-symbolic predictors draw their inferences in such a way that predictions are either computed as a function of, or made consistent with, some given symbolic knowledge*.

* a wide definition that includes the vast majority of the works surveyed in [Besold et al., 2017, Xie et al., 2019, Calegari et al., 2020].

Symbolic Knowledge Injection III

General workflow:



What does 'symbolic' actually mean? I

According to [van Gelder, 1990], symbolic representations of knowledge

- involves a set of symbols,
- which can be combined (e.g., concatenated) in (possibly) infinitely many ways,
- following precise syntactical rules, and
- where both elementary symbols and any admissible combination of them can be assigned with meaning

  ie each symbol can be mapped into some entity from the domain at hand.

Notable example

- formal logic

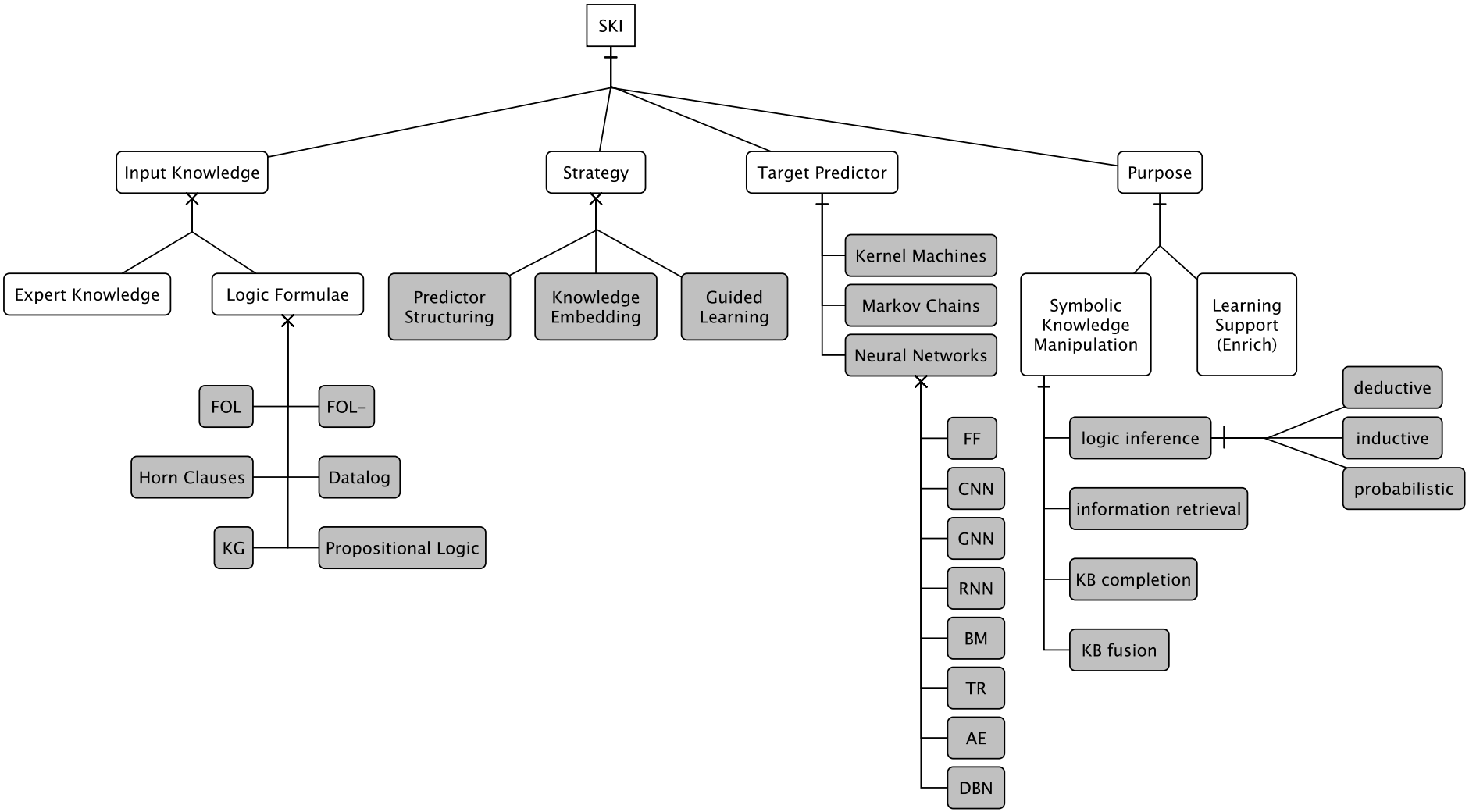
What does 'symbolic' actually mean? II

Opposite notion: **distributed** representations

- where symbols alone have no meaning
- unless it is considered along with its neighbourhood

  ie any other symbol which is close (according to some notion of closeness)

Taxonomy of SKI methods I



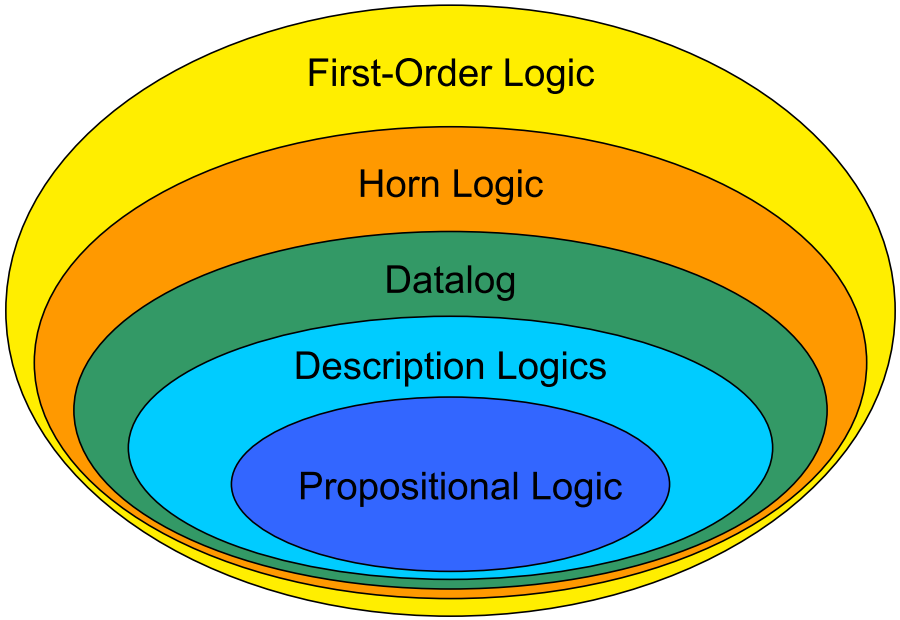
Taxonomy of SKI methods II

- input knowledge how is the knowledge to-be-injected represented?
  - commonly, some sub-set of first-order logic (FOL)
- target predictor which predictors can knowledge be injected into?
  - mostly, neural networks
- strategy how does injection actually work?
  - guided learning the input knowledge is used to guide the training process
  - structuring the internal composition of the predictor is (re-)structured to reflect the input knowledge
  - embedding the input knowledge is converted into numeric array form
- purpose why is knowledge injected in the first place?
  - knowledge manipulation improve / extend / reason about symbol knowledge—subsymbolically
  - learning support improve the sub-symbolic predictor (e.g. speed, size, etc.)

Next in Line...

About Logic I

How to represent knowledge?



- *expressiveness–tractability* trade-off [Levesque and Brachman, 1987, Brachman and Levesque, 2004]

About Logic II

In practice, virtually all SKI algorithms deal with:

- datalog;
- description logics (a.k.a. knowledge graph, KG);
- propositional logic (PL).

First Order Logic I

Overview

- FOL is extremely flexible and expressive
  - variables, quantifiers, structured terms, negation, logic connectives
- one can use recursion to define recursive structures;
  - possibly, intensionally—i.e. without extensively describing everything
- maybe too "powerful" for canonical NN
  - most NN are essentially DAG
  - training via backpropagation[Baldi and Sadowski, 2016] requires no cycles

  → recursion not supported

First Order Logic II

Example of FOL knowledge base (Peano numbers)

$$natural(\texttt{zero})$$
$$\forall X : natural(X) \rightarrow natural(\texttt{successorOf}(varX))$$

Horn Clauses (≈ Prolog) I

Overview

- sub-set of FOL with:
  - implicit quantifiers
  - limited set of logic connectives
- still supports recursion
- nice expressiveness–tractability trade-off
  - often exploited to design/realise automatic reasoning

Horn Clauses (≈ Prolog) II

Example of Horn clauses (Peano numbers)

$$\mathit{natural}(\texttt{zero})$$
$$\mathit{natural}(\texttt{successorOf}(\mathit{varX})) \leftarrow \mathit{natural}(X)$$

Datalog I

Overview

- sub-set of Horn clauses with no recursion
- good for SKI!

Peano numbers in Datalog

- cannot be represented!
  - (as they require recursion)

Description Logics ($\approx$ Knowledge Graphs) I

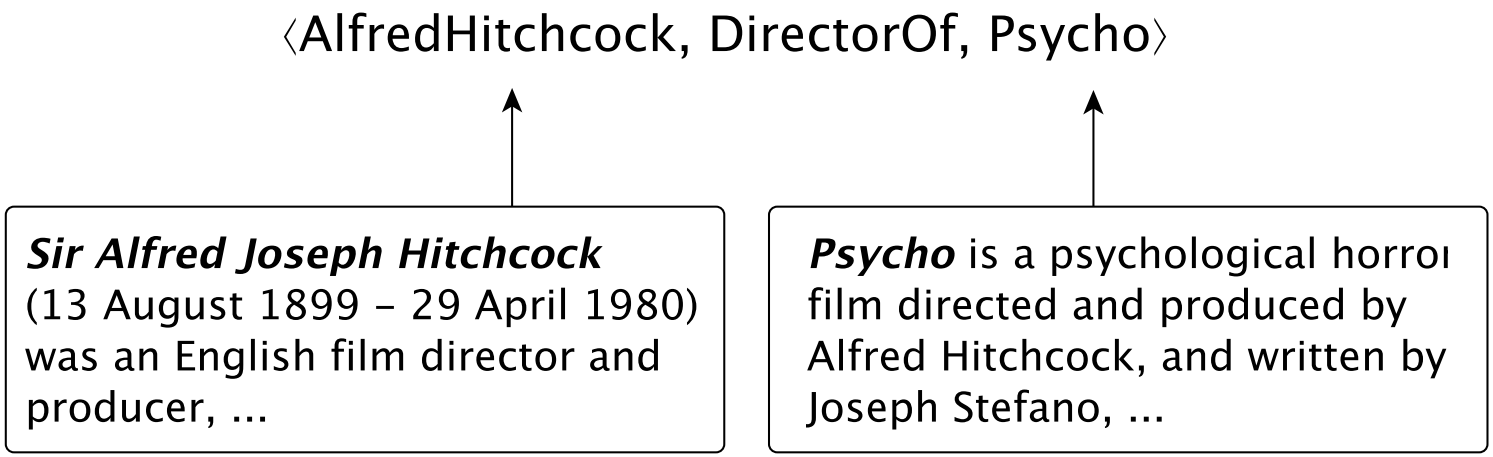
Overview

- Very restricted subset of FOL
  - only constants, variables and n-ary predicates with $n \leq 2$;
- Everything is represented via collections of triplets of the form:

$$\langle \mathtt{a}\ f\ \mathtt{b} \rangle \text{ or } f(\mathtt{a}, \mathtt{b})$$

  where $\mathtt{a}, \mathtt{b}$ are entities, and f is a (binary) relationship
- essentially, directed graph:
  - nodes (i.e. entities) represent individuals,
  - edges (i.e. relationships) represent relations among individuals;

Description Logics (≈ Knowledge Graphs) II



Propositional Logic I

Overview

- The simplest subset of FOL
  - no quantifiers, no terms, no n-ary predicates with $n > 0$
  - essentially, just Boolean algebra
- low expressiveness, but easy to work with.

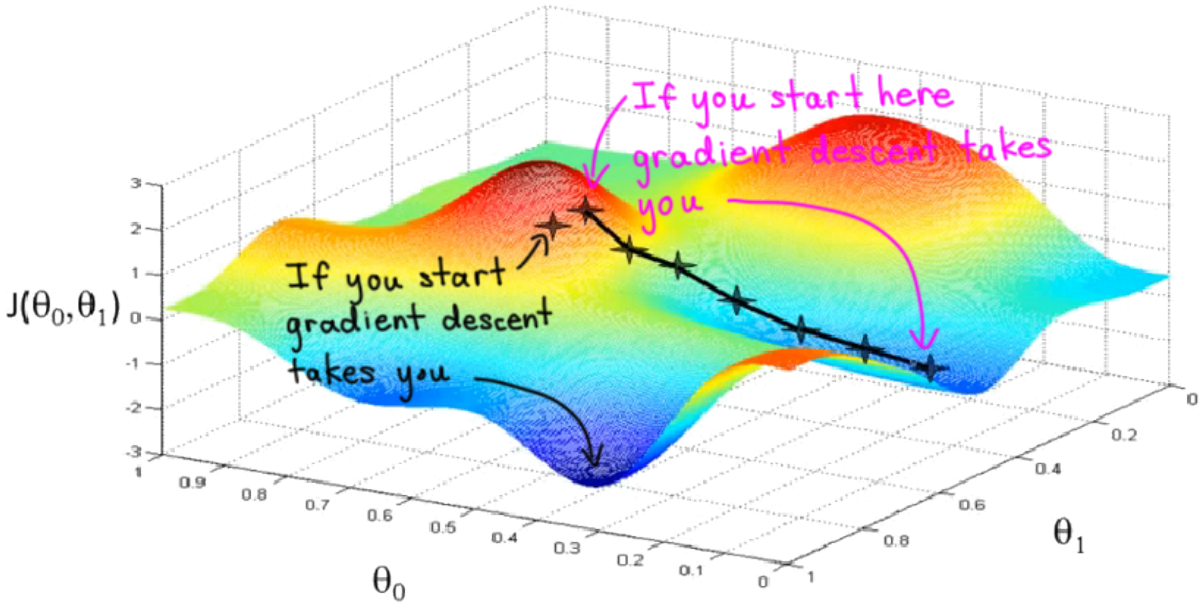
Propositional Logic II

Example

$$big\_petal \wedge average\_sepal \rightarrow \texttt{virginica}.$$
$$big\_petal \wedge \neg average\_sepal \rightarrow \texttt{versicolor}.$$
$$big\_petal \rightarrow \texttt{setosa}.$$
$$average\_sepal \equiv (3 \leq SepalWidth < 5)$$
$$big\_petal \equiv (PetalLength > 3)$$

Next in Line...

Strategy 1: Guided Learning I



- learning is essentially an optimizionation process
- . . . often performed via gradient descent
  ie minimising a loss function

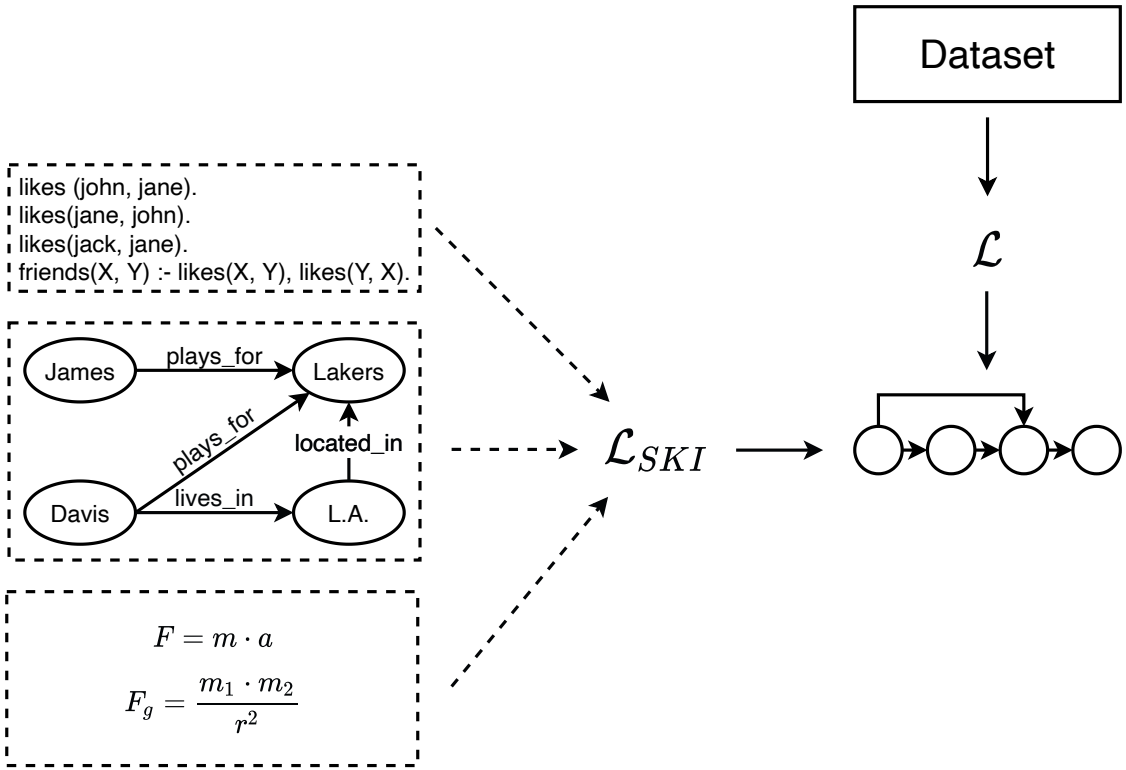
Strategy 1: Guided Learning II

SKI via Guided Learning

1. Input knowledge is converted into a cost factor
   ie the more the knowledge is violated, the higher the cost
2. The loss function is altered to include that cost factor
   eg as a simple additive regularisation factor
3. The predictor is then trained as usual

→ Training minimises both the predictors' error and inconsistency w.r.t. knowledge

Strategy 1: Guided Learning III



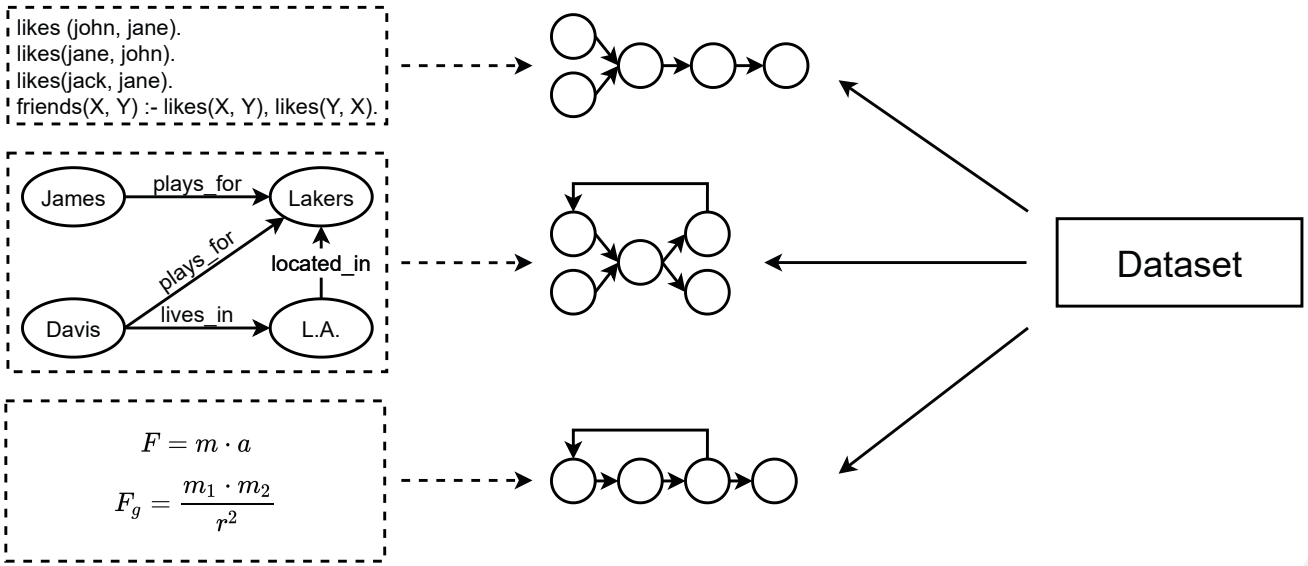
Strategy 2: Structuring I

SKI via Guided Learning

- The predictor's inner architecture is shaped to"mimic" the knowledge
- Shaping is predictor-dependent
  - eg for neural networks, this means creating ad-hoc layers
    - where small groups of neurons are used to compute pieces of a formula

→ The predictor directly exploits the knowledge during inference

Strategy 2: Structuring II

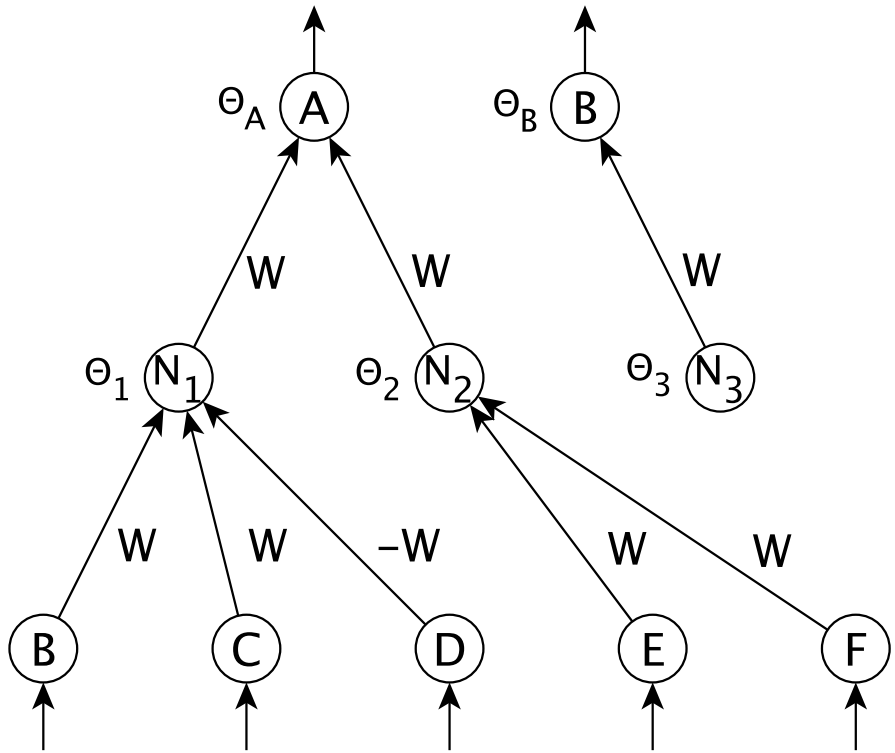


Strategy 2: Structuring III

Example:

$$A \leftarrow B \wedge C \wedge \neg D.$$
$$A \leftarrow E \wedge F.$$
$$B \leftarrow \texttt{true}.$$

$\leftrightarrow$

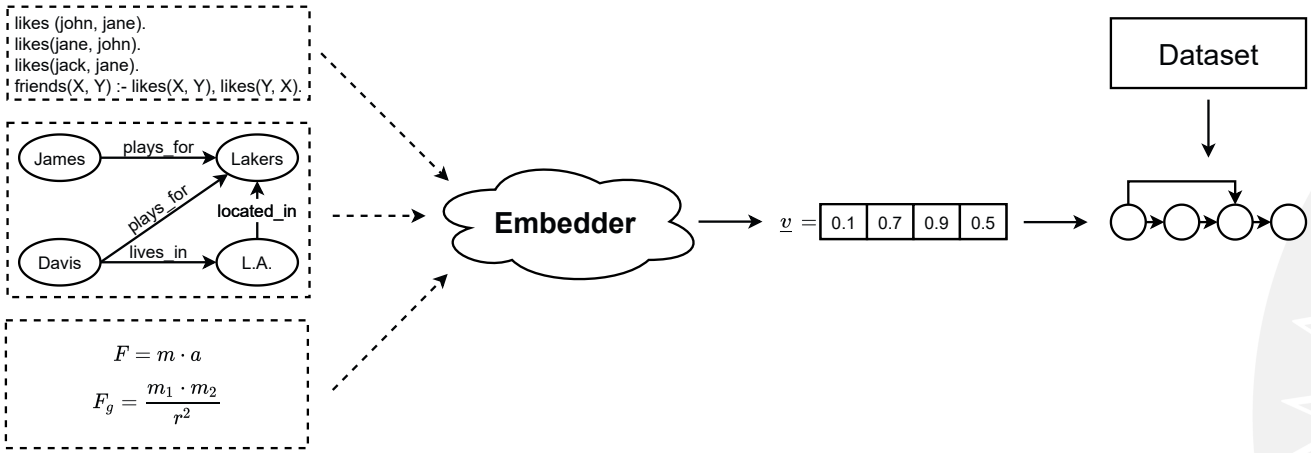


Strategy 3: Embedding I

SKI via Guided Learning

- Input knowledge is converted into numeric tensor(s)
- These are used as the training set for an ordinary learning process

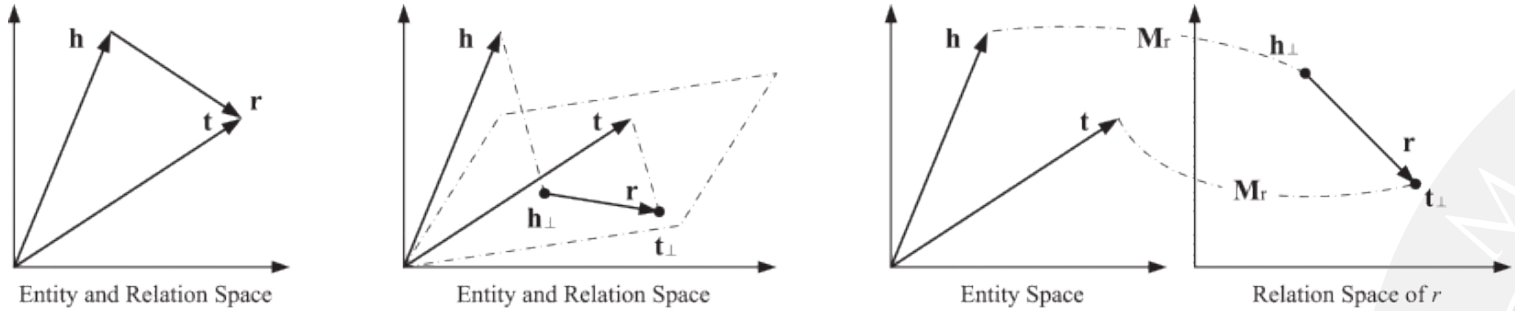
→ The predictor is trained and used 'as usual'



Strategy 3: Embedding II

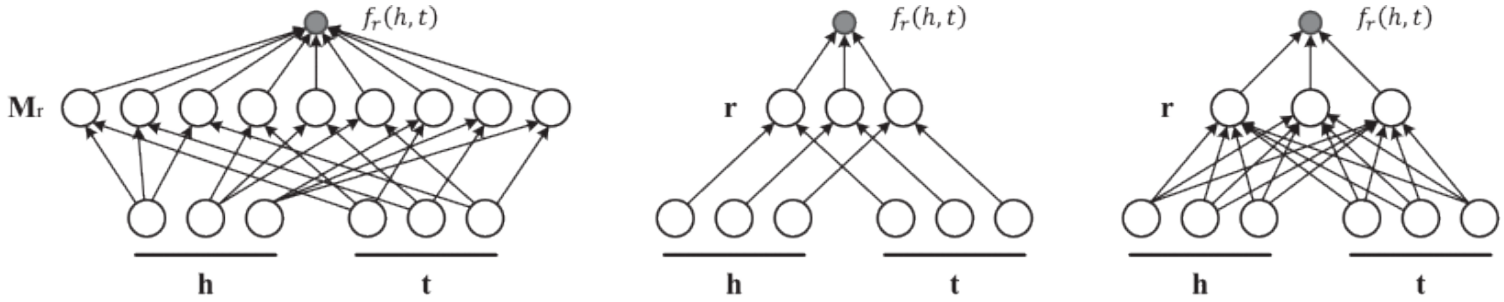
Example: knowledge graph embedding[Wang et al., 2017]

- entities and relations are embedded into continuos vector spaces;
- scoring function $f_r(h, t)$ defined on each fact (h, r, t) to measure its plausibility;

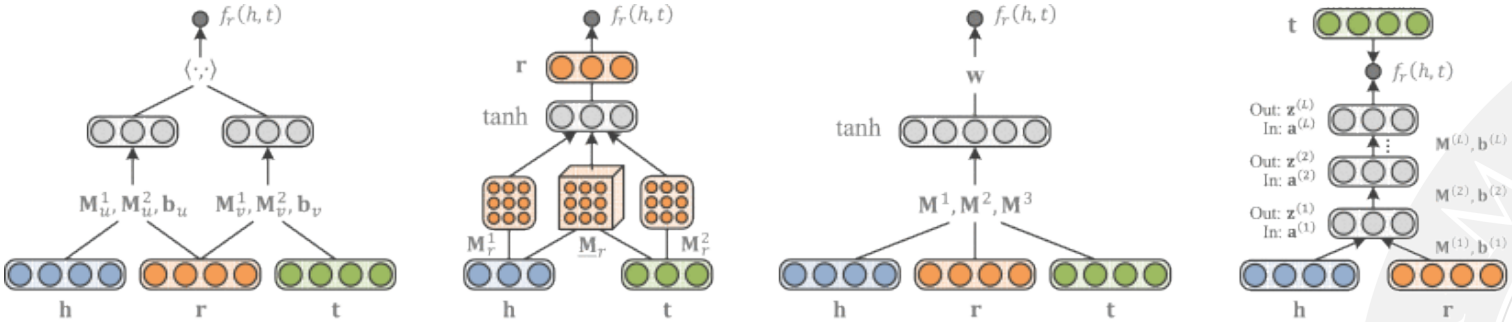


(a) TransE. (b) TransH. (c) TransR.

Strategy 3: Embedding III



(a) RESCAL. (b) DistMult. (c) HolE.



(a) SME. (b) NTN. (c) MLP. (d) NAM.

Next in Line...

Knowledge Injection via Network Structuring[Magnini et al., 2022a] I

KINS

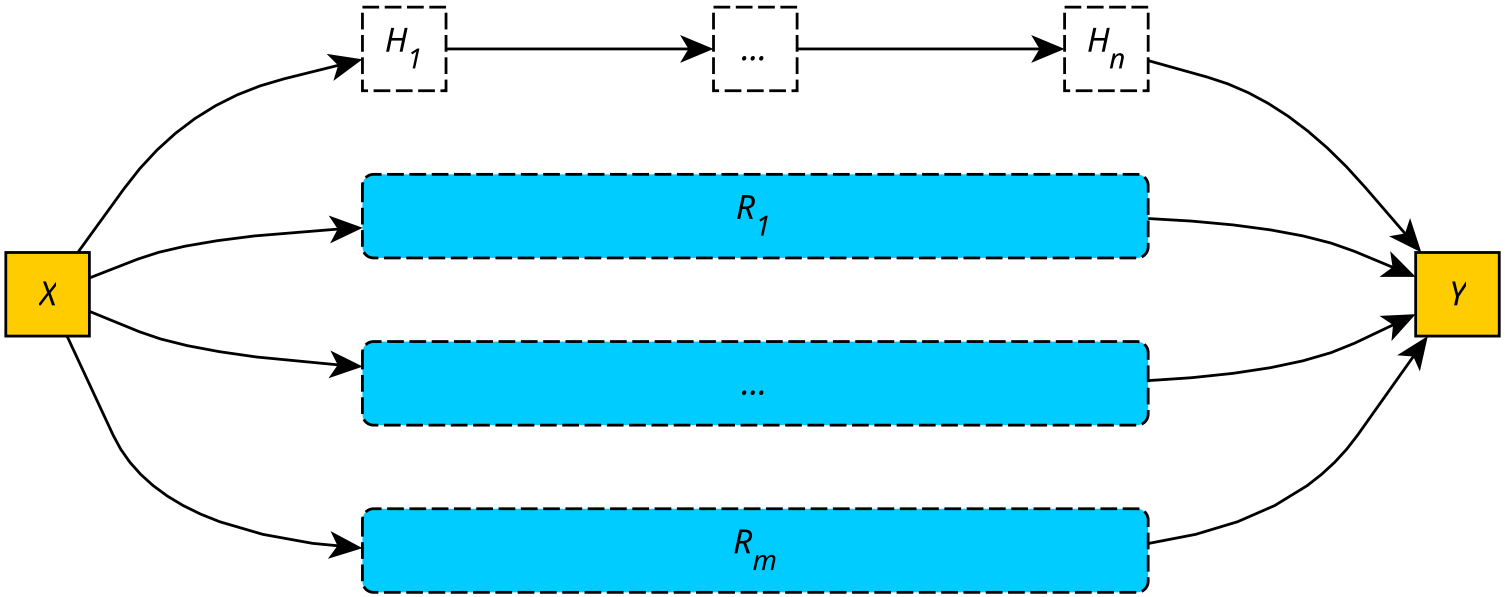
purpose → learning support;

target predictor → neural networks;

strategy → structuring;

input logic → stratified Datalog with negation.

Knowledge Injection via Network Structuring [Magnini et al., 2022a] II



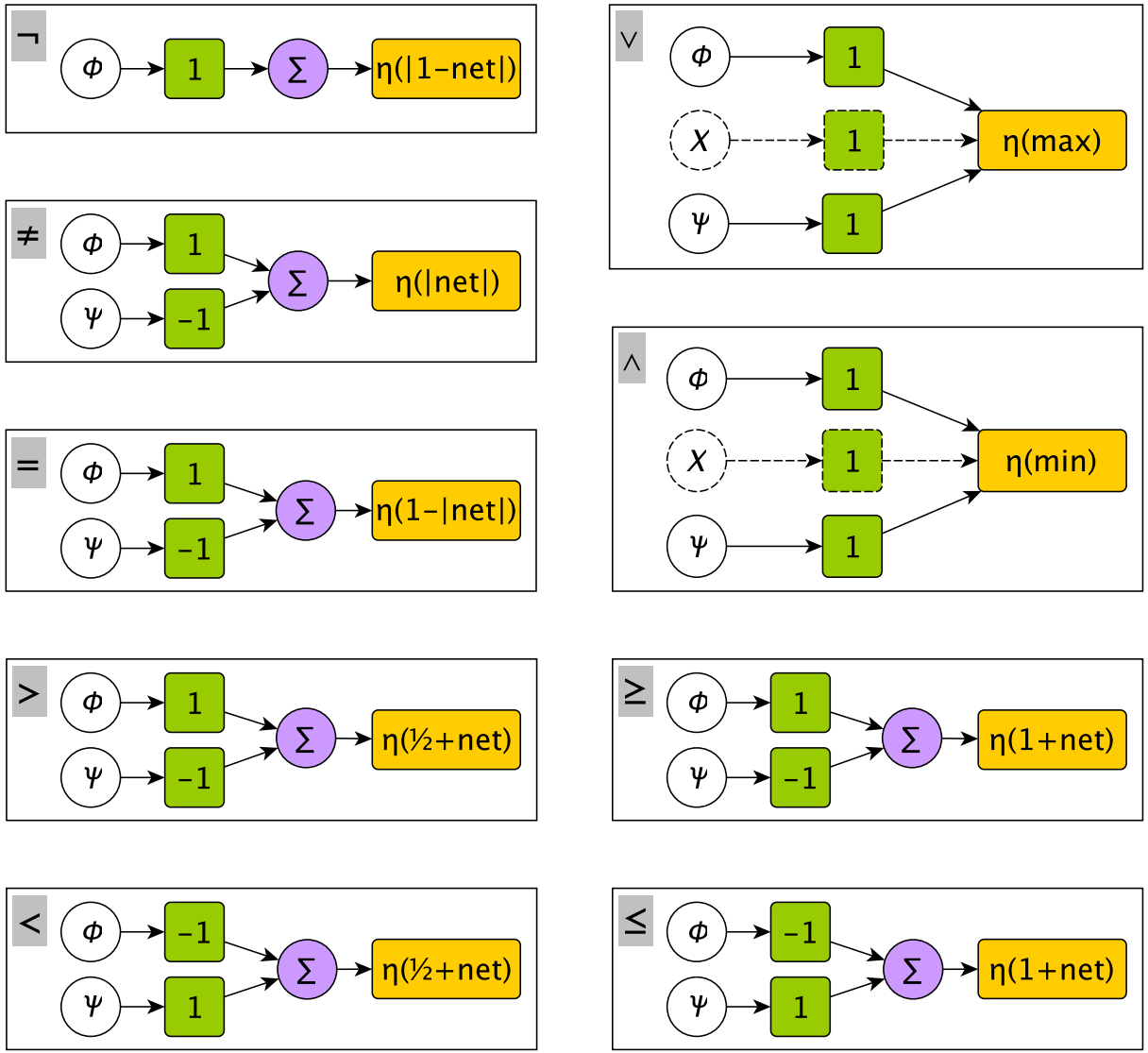
Knowledge Injection via Network Structuring[Magnini et al., 2022a] III

Formula	C. interpretation	Formula	C. interpretation
$[\![\neg\phi]\!]$	$\eta\{1 - [\![\phi]\!]\}$	$[\![\phi \leftarrow \psi]\!]$	$\eta\{min\{1, 1 - [\![\phi]\!] + [\![\psi]\!]\}\}$
$[\![\phi \wedge \psi]\!]$	$\eta\{min\{[\![\phi]\!], [\![\psi]\!]\}\}$	$[\![\phi \leftrightarrow \psi]\!]$	$\eta\{min\{1, 1 - \lvert[\![\phi]\!] - [\![\psi]\!]\rvert\}\}$
$[\![\phi \vee \psi]\!]$	$\eta\{max\{[\![\phi]\!], [\![\psi]\!]\}\}$	$[\![\text{expr}(\bar{X})]\!]$	$\text{expr}([\![\bar{X}]\!])$
$[\![\phi = \psi]\!]$	$\eta\{[\![\neg(\phi \neq \psi)]\!]\}$	$[\![\texttt{true}]\!]$	1
$[\![\phi \neq \psi]\!]$	$\eta\{\lvert[\![\phi]\!] - [\![\psi]\!]\rvert\}$	$[\![\texttt{false}]\!]$	0
$[\![\phi > \psi]\!]$	$\eta\{max\{0, [\![\phi]\!] - [\![\psi]\!]\}\}$	$[\![X]\!]$	x
$[\![\phi \geq \psi]\!]$	$\eta\{[\![(\phi > \psi) \vee (\phi = \psi)]\!]\}$	$[\![\texttt{k}]\!]$	k
$[\![\phi < \psi]\!]$	$\eta\{max\{0, [\![\psi]\!] - [\![\phi]\!]\}\}$	$[\![p(\bar{X})]\!]^{**}$	$[\![\psi_1 \vee \ldots \vee \psi_k]\!]$
$[\![\phi \leq \psi]\!]$	$\eta\{[\![(\phi < \psi) \vee (\phi = \psi)]\!]\}$	$[\![class(\bar{X}, \texttt{y}_i) \leftarrow \psi]\!]$	$[\![\psi]\!]^{*}$
$[\![\phi \rightarrow \psi]\!]$	$\eta\{min\{1, 1 - [\![\psi]\!] + [\![\phi]\!]\}\}$		

* encodes the value for the i^{th} output

** assuming p is defined by k clauses of the form:
$p(\bar{X}) \leftarrow \psi_1, \ \ldots, \ p(\bar{X}) \leftarrow \psi_k$

Knowledge Injection via Network Structuring [Magnini et al., 2022a] IV



Knowledge Injection via Lambda Layer[Magnini et al., 2022c] I

KILL

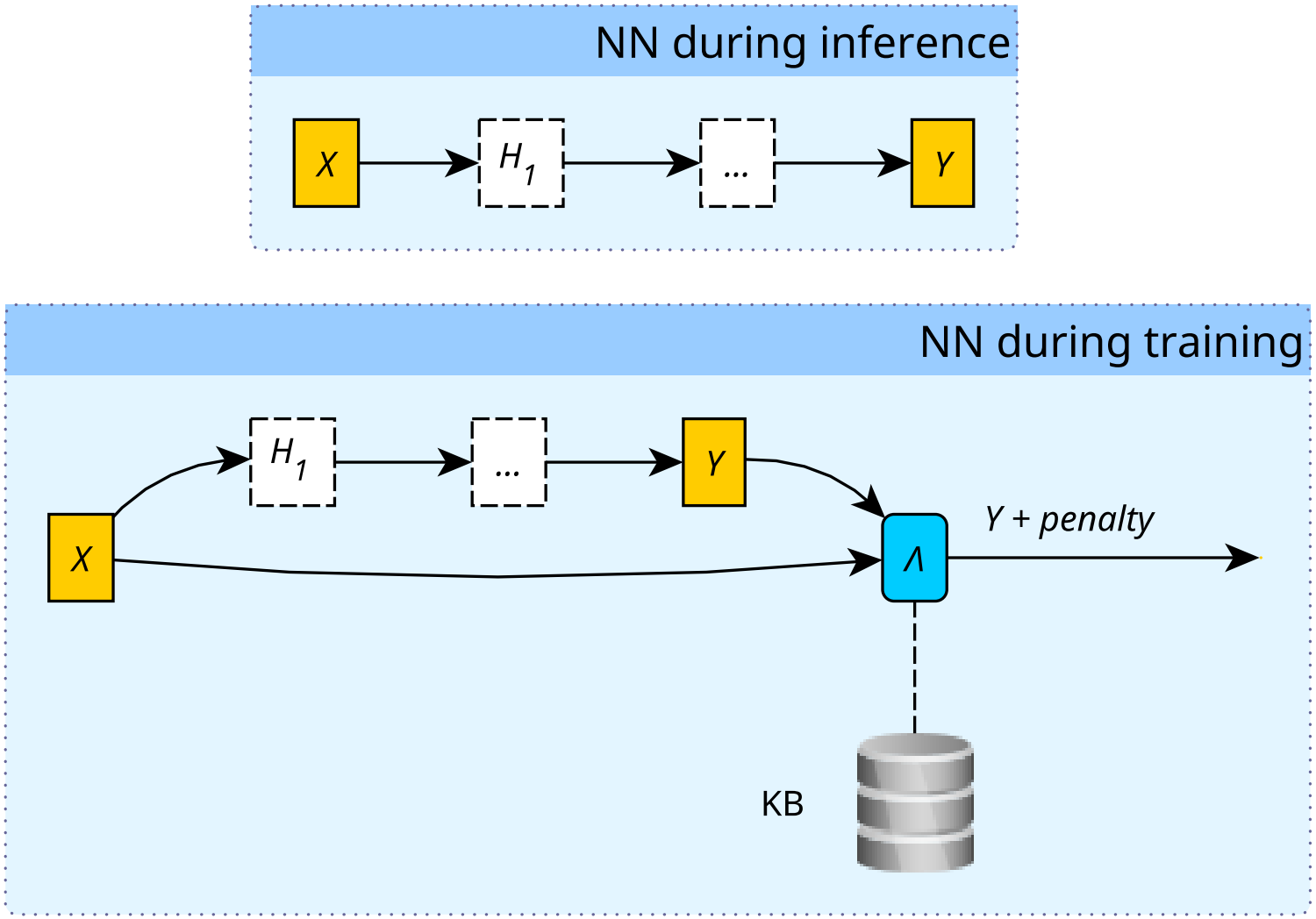
purpose → learning support;

target predictor → neural networks;

strategy → guided learning;

input logic → stratified Datalog with negation.

Knowledge Injection via Lambda Layer [Magnini et al., 2022c] II



Knowledge Injection via Lambda Layer[Magnini et al., 2022c] III

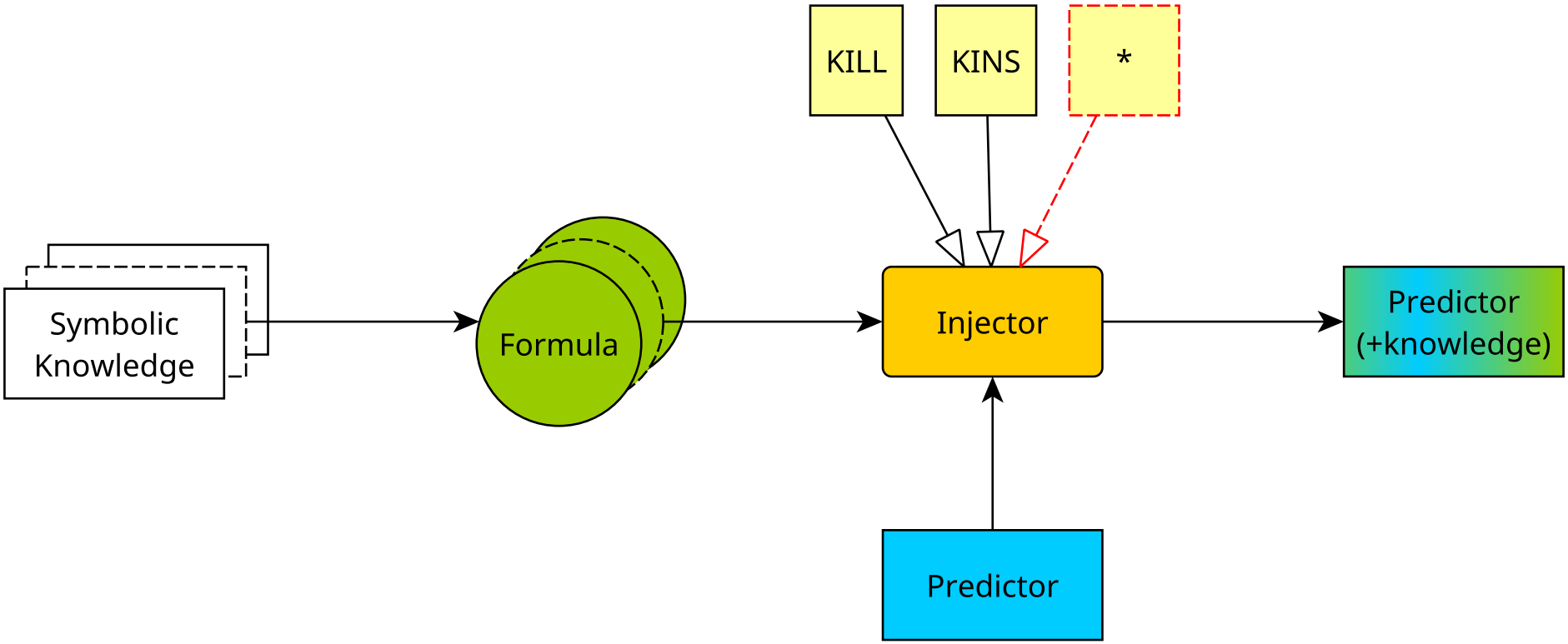
Formula	C. interpretation	Formula	C. interpretation
$[\![\neg\phi]\!]$	$\eta(1 - [\![\phi]\!])$	$[\![\phi \leq \psi]\!]$	$\eta([\![\phi]\!] - [\![\psi]\!])$
$[\![\phi \wedge \psi]\!]$	$\eta(max([\![\phi]\!], [\![\psi]\!]))$	$[\![class(\bar{X}, \mathrm{y}_i) \leftarrow \psi]\!]$	$[\![\psi]\!]^*$
$[\![\phi \vee \psi]\!]$	$\eta(min([\![\phi]\!], [\![\psi]\!]))$	$[\![\mathrm{expr}(\bar{X})]\!]$	$\mathrm{expr}([\![\bar{X}]\!])$
$[\![\phi = \psi]\!]$	$\eta(\lvert[\![\phi]\!] - [\![\psi]\!]\rvert)$	$[\![\mathtt{true}]\!]$	0
$[\![\phi \neq \psi]\!]$	$[\![\neg(\phi = \psi)]\!]$	$[\![\mathtt{false}]\!]$	1
$[\![\phi > \psi]\!]$	$\eta(0.5 - [\![\phi]\!] + [\![\psi]\!])$	$[\![X]\!]$	x
$[\![\phi \geq \psi]\!]$	$\eta([\![\psi]\!] - [\![\phi]\!])$	$[\![\mathtt{k}]\!]$	k
$[\![\phi < \psi]\!]$	$\eta(0.5 + [\![\phi]\!] - [\![\psi]\!])$	$[\![p(\bar{X})]\!]^{**}$	$[\![\psi_1 \vee \ldots \vee \psi_k]\!]$

* encodes the penalty for the i^{th} neuron

** assuming predicate p is defined by k clauses of the form:
$p(\bar{X}) \leftarrow \psi_1, \ \ldots, \ p(\bar{X}) \leftarrow \psi_k$

Next in Line...

Overall Design I



Overall Design II

Key components:

injector: any entity capable of injecting knowledge into a sub-symbolic predictor

- it simply alters/reconfigures the predictor...
- ...which should be trained after the injector operates

predictor: the partially-trained classifier/regressor where knowledge should be injected into

- untrained is ok too

formula: formal representation of the symbolic knowledge to be injected

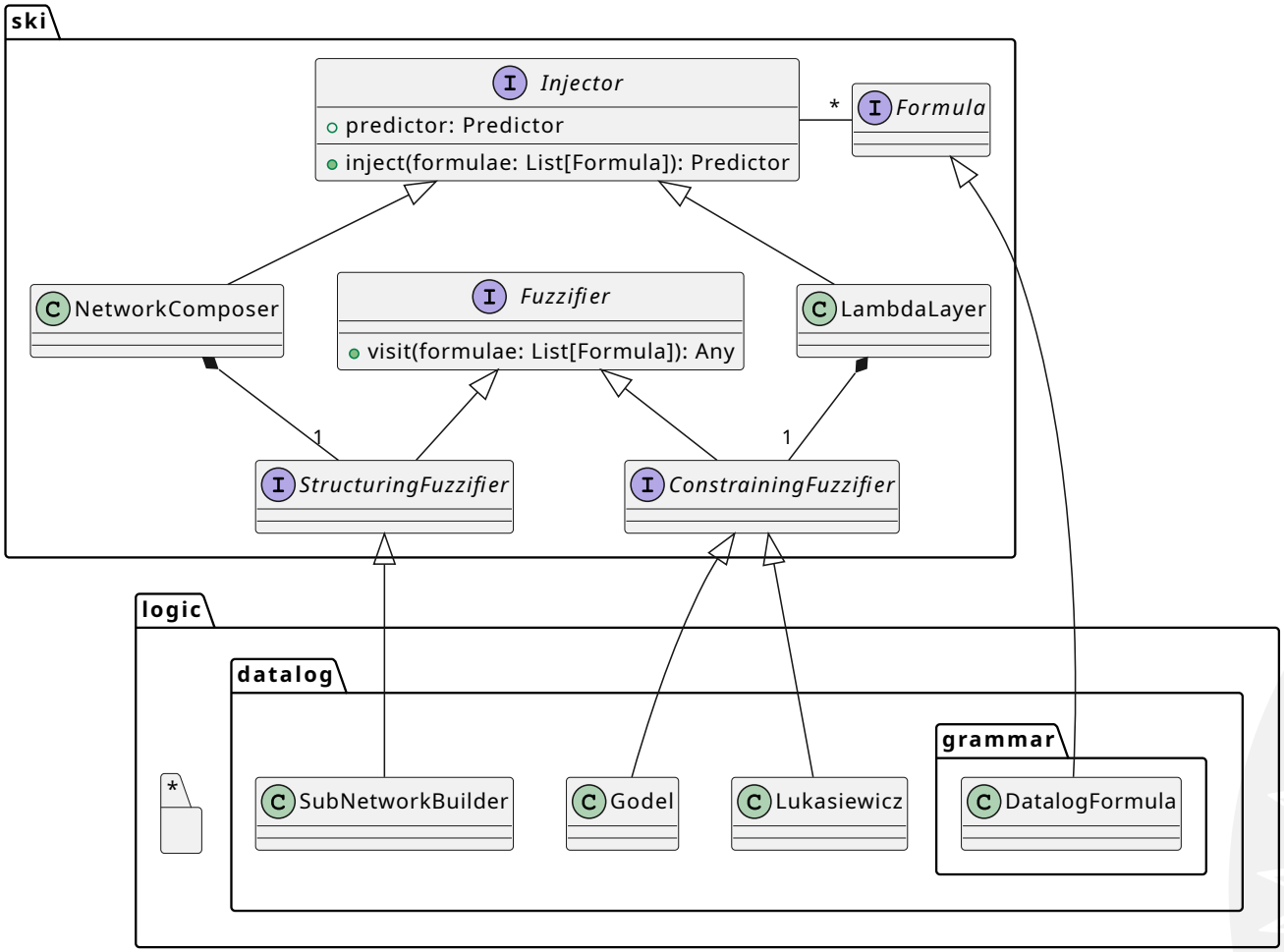
- e.g. in Prolog or FOL syntax

Overall Design III

Unified API for SKI

- 1 interface for `Injector`, several implementations
  eg KILL, KINS, etc.
- 1 interface for `Formula`, several implementations
  eg FOL, Datalog, etc.
- 1 interface for `Predictor`, several implementations
  eg NN, kNN, DT

API Design I


ski
Injector
predictor: Predictor
inject(formulae: List[Formula]): Predictor
*
Formula
NetworkComposer
Fuzzifier
visit(formulae: List[Formula]): Any
LambdaLayer
1
1
StructuringFuzzifier
ConstrainingFuzzifier
logic
datalog
grammar
*
SubNetworkBuilder
Godel
Lukasiewicz
DatalogFormula

API Design II

Remarks

- The user only needs to know:
  - the particular injector to exploit (and its paramenters)
  - the particular parser to decode logic rules

Next in Line...

Tutorial

Two ways to reproduce the tutorial:

GitHub Repository (long way)

https://github.com/pikalab-unibo/prima-tutorial-2022

DockerHub Images (quick way)

https://hub.docker.com/r/pikalab/prima-tutorial-2022/tags

Next in Line...

How to set the tutorial up from GitHub I

Enviromental pre-requisites

- Python `3.9.x`
- JDK $\geq$ `11`
- Git

1. `git clone https://github.com/pikalab-unibo/prima-tutorial-2022`
2. `cd prima-tutorial-2022`
3. `pip install -r requirements.txt`
4. `jupyter notebook`

How to set the tutorial up from GitHub II

5. Your browser should automatically open showing the following page:

6. open the `psyki-tutorial.ipynb` notebook
7. listen to the speaker presenting the tutorial =)

Next in Line...

How to set the tutorial up via Docker I

Enviromental pre-requisites

- Docker

1. $\texttt{DOCKER\_IMAGE}=\begin{cases}\texttt{pikalab/prima-tutorial-2022:latest} \\ \texttt{pikalab/prima-tutorial-2022:latest-apple-m1}\end{cases}$
2. `docker pull $DOCKER_IMAGE`
   - in case of lacking Internet access:

     ```
     docker image load -i /path/to/local/image/file.tar
     ```
3. `docker run -it -rm -name prima-tutorial-ske-ski -p 8888:8888 $DOCKER_IMAGE`
4. Some textual output such as the following one should appear:

How to set the tutorial up via Docker II

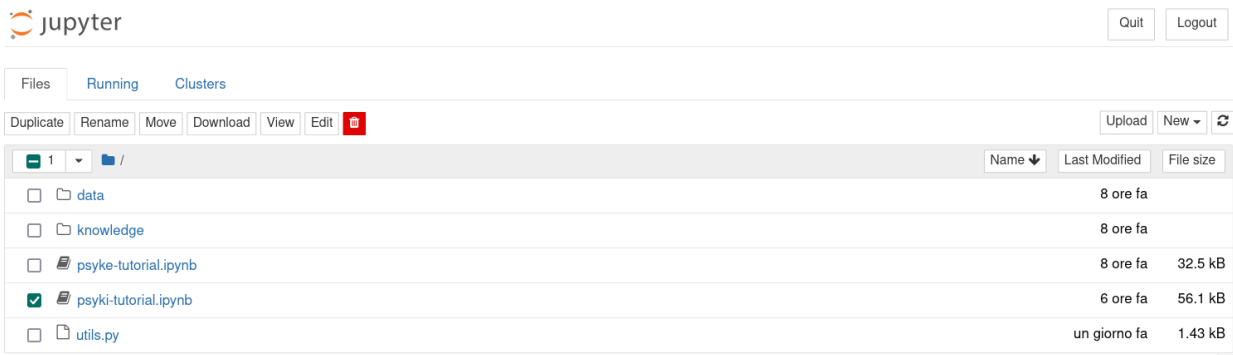
```
1  [I 09:51:46.940 NotebookApp] Writing notebook server cookie secret to /root/.local/
       share/jupyter/runtime/notebook_cookie_secret
2  [I 09:51:47.159 NotebookApp] Serving notebooks from local directory: /notebook
3  [I 09:51:47.159 NotebookApp] Jupyter Notebook 6.5.2 is running at:
4  [I 09:51:47.159 NotebookApp] http://cb0a3641caf0:8888/?token=2
       b02d31671c6ad9e9cf8e036eb6962d3592af9cfdd5e60bd
5  [I 09:51:47.159 NotebookApp]  or http://127.0.0.1:8888/?token=2
       b02d31671c6ad9e9cf8e036eb6962d3592af9cfdd5e60bd
6  [I 09:51:47.160 NotebookApp] Use Control-C to stop this server and shut down all
       kernels (twice to skip confirmation).
7  [C 09:51:47.162 NotebookApp]
8
9  To access the notebook, open this file in a browser:
10     file:///root/.local/share/jupyter/runtime/nbserver-7-open.html
11 Or copy and paste one of these URLs:
12     http://cb0a3641caf0:8888/?token=2
           b02d31671c6ad9e9cf8e036eb6962d3592af9cfdd5e60bd
13 or http://127.0.0.1:8888/?token=2b02d31671c6ad9e9cf8e036eb6962d3592af9cfdd5e60bd
```

How to set the tutorial up via Docker III

5. Copy-paste into your browser any link of the form:

   `http://cb0a3641caf0:8888/?token=`*`TOKEN`*

6. Your browser should now be showing the following page:



7. open the `psyki-tutorial.ipynb` notebook
8. listen to the speaker presenting the tutorial =)

Next in Line...

Notable Remarks

- knowledge bases should express relations about input–output pairs
- embedding implies extensional representation of knowledge
  - guided learning, and structuring support intensional knowledge
- propositional knowledge implies binarising the I/O spaces

Current Limitations

- support for regression is preliminary
- recursive data structures are not supported
- recursive clauses are not supported
- extensional representation cost storage
  - not always possible
- guided learning works poorly with lacking data

Future research activities

- foundational: address recursion
- practical: address regression
- is SKI possible outside the NN domain?

Symbolic Knowledge Injection via PSyKI

A tutorial

Giovanni Ciatto Matteo Magnini
giovanni.ciatto@unibo.it matteo.magnini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna, Cesena, Italy

24^{th} International Conference on
Principles and Practice of Multi-Agent Systems
November 16, 2022

References I

[Baldi and Sadowski, 2016] Baldi, P. and Sadowski, P. J. (2016).
A theory of local learning, the learning channel, and the optimality of backpropagation.
Neural Networks, 83:51–74
DOI:10.1016/j.neunet.2016.07.006.

[Besold et al., 2017] Besold, T. R., d'Avila Garcez, A. S., Bader, S., Bowman, H., Domingos, P. M., Hitzler, P., Kühnberger, K., Lamb, L. C., Lowd, D., Lima, P. M. V., de Penning, L., Pinkas, G., Poon, H., and Zaverucha, G. (2017).
Neural-symbolic learning and reasoning: A survey and interpretation.
CoRR, abs/1711.03902
http://arxiv.org/abs/1711.03902.

[Brachman and Levesque, 2004] Brachman, R. J. and Levesque, H. J. (2004).
The tradeoff between expressiveness and tractability.
In Brachman, R. J. and Levesque, H. J., editors, *Knowledge Representation and Reasoning*, The Morgan Kaufmann Series in Artificial Intelligence, pages 327–348. Morgan Kaufmann, San Francisco
DOI:https://doi.org/10.1016/B978-155860932-7/50101-1.

References II

[Calegari et al., 2020] Calegari, R., Ciatto, G., and Omicini, A. (2020).
On the integration of symbolic and sub-symbolic techniques for XAI: A survey.
Intelligenza Artificiale, 14(1):7–32
DOI:10.3233/IA-190036.

[Levesque and Brachman, 1987] Levesque, H. J. and Brachman, R. J. (1987).
Expressiveness and tractability in knowledge representation and reasoning.
Comput. Intell., 3:78–93
DOI:10.1111/j.1467-8640.1987.tb00176.x.

[Magnini et al., 2022a] Magnini, M., Ciatto, G., and Omicini, A. (2022a).
KINS: Knowledge injection via network structuring.
In Calegari, R., Ciatto, G., and Omicini, A., editors, *CILC 2022 – Italian Conference on Computational Logic*, volume 3204 of *CEUR Workshop Proceedings*, pages 254–267. CEUR-WS
http://ceur-ws.org/Vol-3204/paper_25.pdf.

References III

[Magnini et al., 2022b] Magnini, M., Ciatto, G., and Omicini, A. (2022b).
On the design of PSyKI: a platform for symbolic knowledge injection into sub-symbolic predictors.
In Calvaresi, D., Najjar, A., Winikoff, M., and Främling, K., editors, *Explainable and Transparent AI and Multi-Agent Systems*, volume 13283 of *Lecture Notes in Computer Science*, chapter 6, pages 90–108. Springer.
4th International Workshop, EXTRAAMAS 2022, Virtual Event, May 9–10, 2022, Revised Selected Papers
DOI:10.1007/978-3-031-15565-9_6.

[Magnini et al., 2022c] Magnini, M., Ciatto, G., and Omicini, A. (2022c).
A view to a KILL: Knowledge injection via lambda layer.
In Ferrando, A. and Mascardi, V., editors, *WOA 2022 – 23rd Workshop "From Objects to Agents"*, volume 3261 of *CEUR Workshop Proceedings*, pages 61–76. Sun SITE Central Europe, RWTH Aachen University
http://ceur-ws.org/Vol-3261/paper5.pdf.

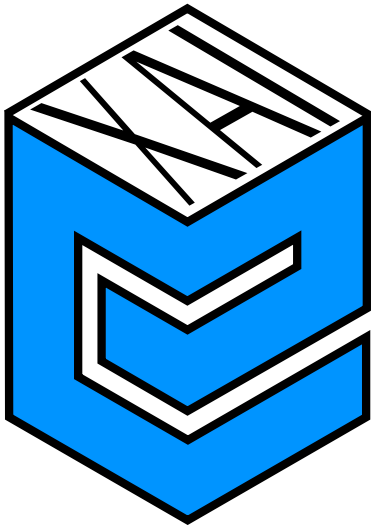
References IV

[van Gelder, 1990] van Gelder, T. (1990).
Why distributed representation is inherently non-symbolic.
In Dorffner, G., editor, *Konnektionismus in Artificial Intelligence und Kognitionsforschung. Proceedings 6. Österreichische Artificial Intelligence-Tagung (KONNAI), Salzburg, Österreich, 18. bis 21. September 1990*, volume 252 of *Informatik-Fachberichte*, pages 58–66. Springer
DOI:10.1007/978-3-642-76070-9_6.

[Wang et al., 2017] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017).
Knowledge graph embedding: A survey of approaches and applications.
IEEE Trans. Knowl. Data Eng., 29(12):2724–2743
DOI:10.1109/TKDE.2017.2754499.

[Xie et al., 2019] Xie, Y., Xu, Z., Meel, K. S., Kankanhalli, M. S., and Soh, H. (2019).
Embedding symbolic knowledge into deep networks.
In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4235–4245
https://proceedings.neurips.cc/paper/2019/hash/7b66b4fd401a271a1c7224027ce111bc-Abstract.html.

EXPECTATION



EXPECTATION

PERSONALIZED EXPLAINABLE ARTIFICIAL INTELLIGENCE FOR DECENTRALIZED AGENTS WITH HETEROGENEOUS KNOWLEDGE

This presentation was partially supported by the CHIST-ERA IV project "EXPECTATION" – CHIST-ERA-19-XAI-005 –, co-funded by EU and the Italian MUR (Ministry for University and Research).