From Statistical Relational AI to Neural Symbolic Computation

Luc De Raedt, Sebastijan Dumancic, Robin Manhaeve, Giuseppe Marra
firstname.lastname@kuleuven.be

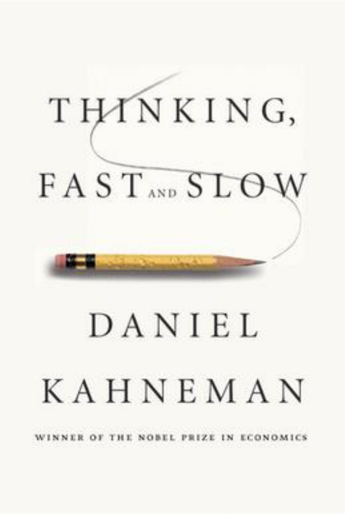
reusing some slides from previous tutorials with
Angelika Kimmig, Kristian Kersting, David Poole, and Sriraam Natarajan

You can find an up-to-date version of this tutorial and additional content at

https://dtai.cs.kuleuven.be/tutorials/nesytutorial

Introduction

Learning and Reasoning both needed



- System 1 - thinking fast - can do things like 2+2 = ? and recognise objects in image
- System 2 - thinking slow - can reason about solving complex problems - planning a complex task
- alternative terms — data-driven vs knowledge-driven, symbolic vs subsymbolic, solvers and learners, neuro-symbolic…
- **A lot of work on integrating learning and reasoning, neural symbolic computation to integrate logic / symbols reasoning with neural networks**

  - see also arguments
    by Marcus, Darwiche, Levesque, Tenenbaum, Geffner, Bengio, Le Cun, Kautz, …
    see also AI Debates

Real-life problems involve two important aspects.

https://www.theorie-blokken.be/nl/gratis-proefexamen

Who can go first ?

A. The red car

B. The blue van

C. The white car

Real-life problems involve two important aspects.

https://www.theorie-blokken.be/nl/gratis-proefexamen

Who can go first ?

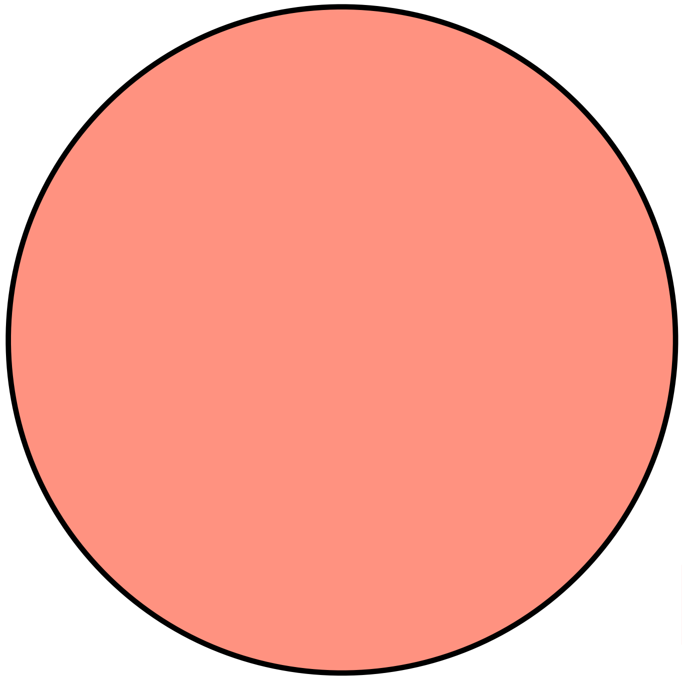
A. The red car
B. The blue van
C. The white car

Reasoning

Sub-symbolic perception

Thinking fast

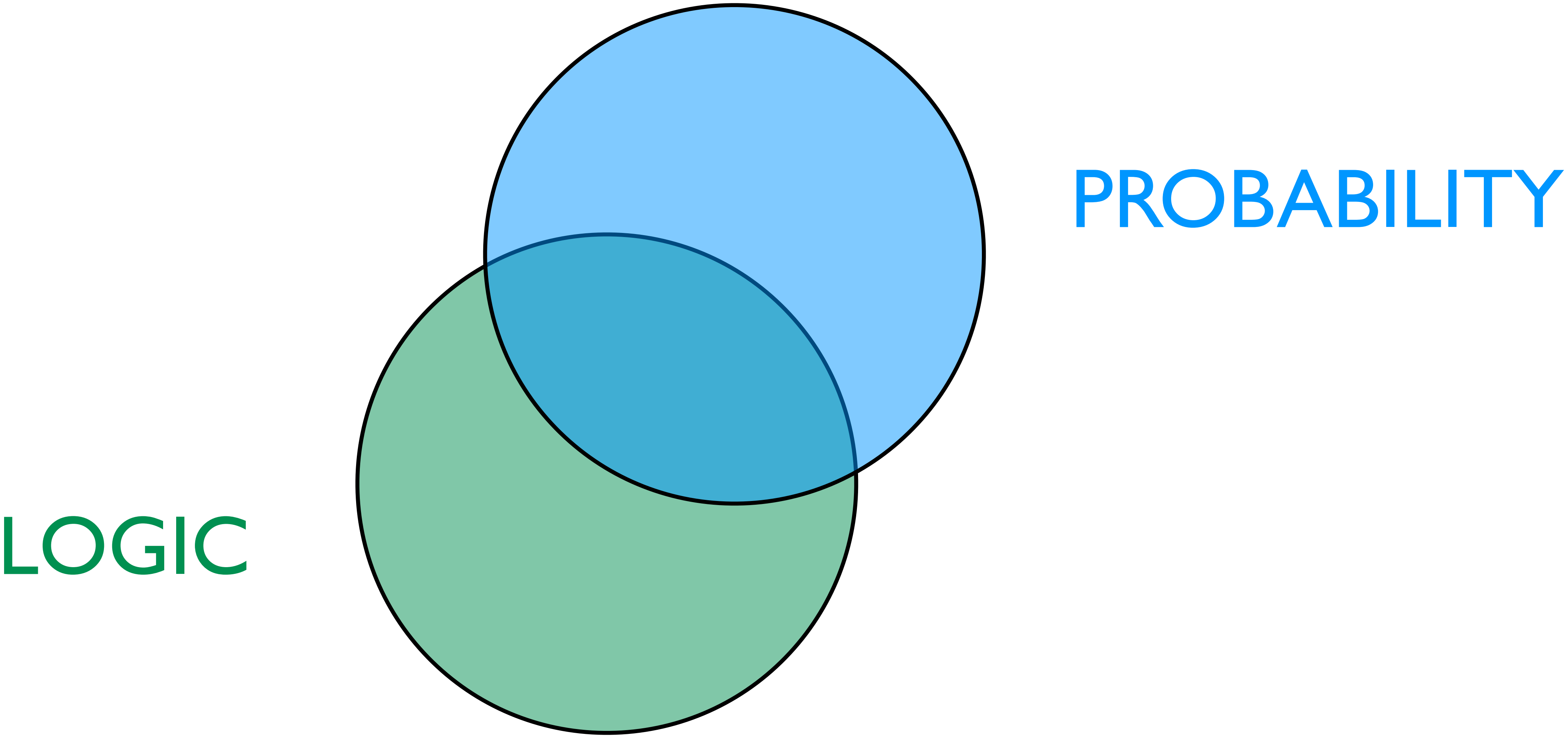
MAIN PARADIGM in AI
Focus on Learning



NEURAL

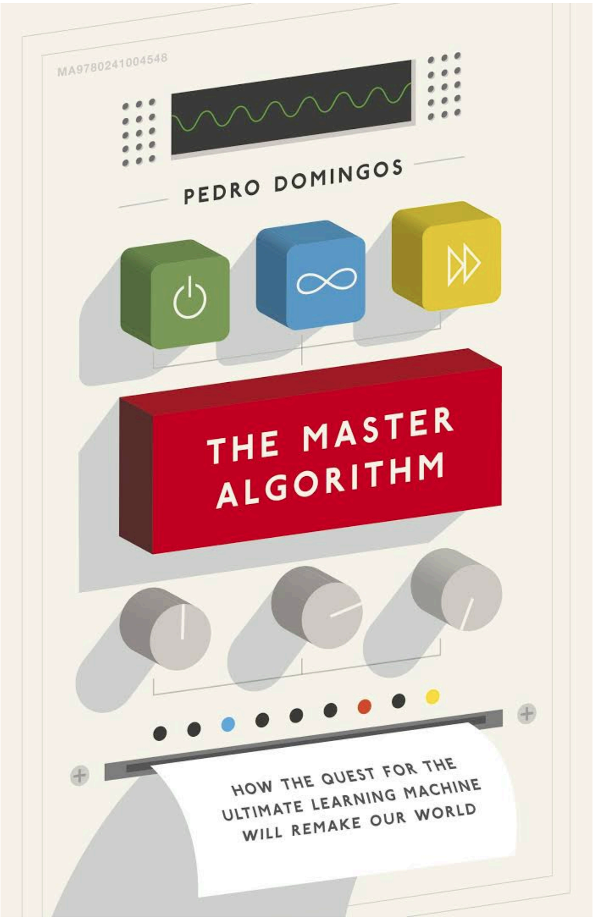
Thinking slow = reasoning

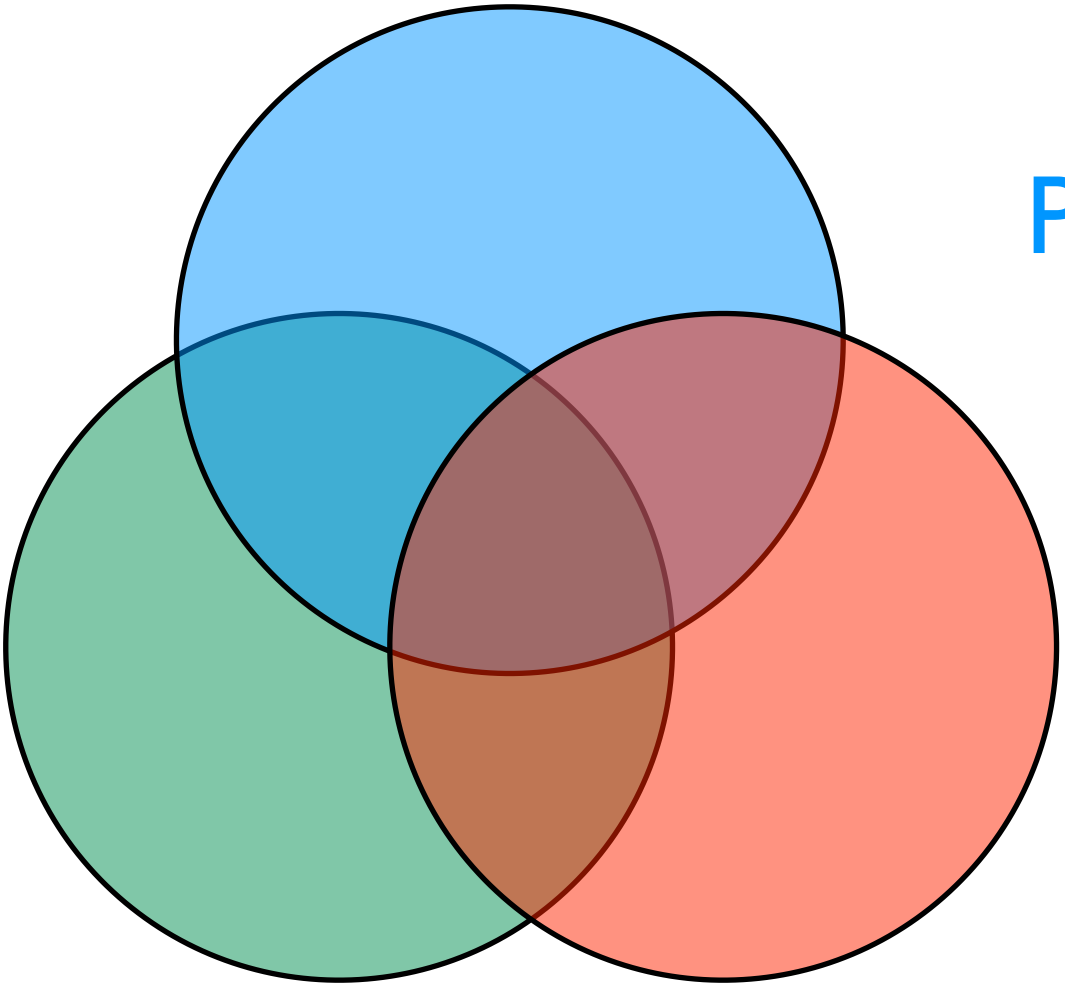
TWO MAIN PARADIGMS in AI



Their integration has been well studied in Probabilistic (Logic) Programming and Statistical Relational AI (StarAI)

Learning





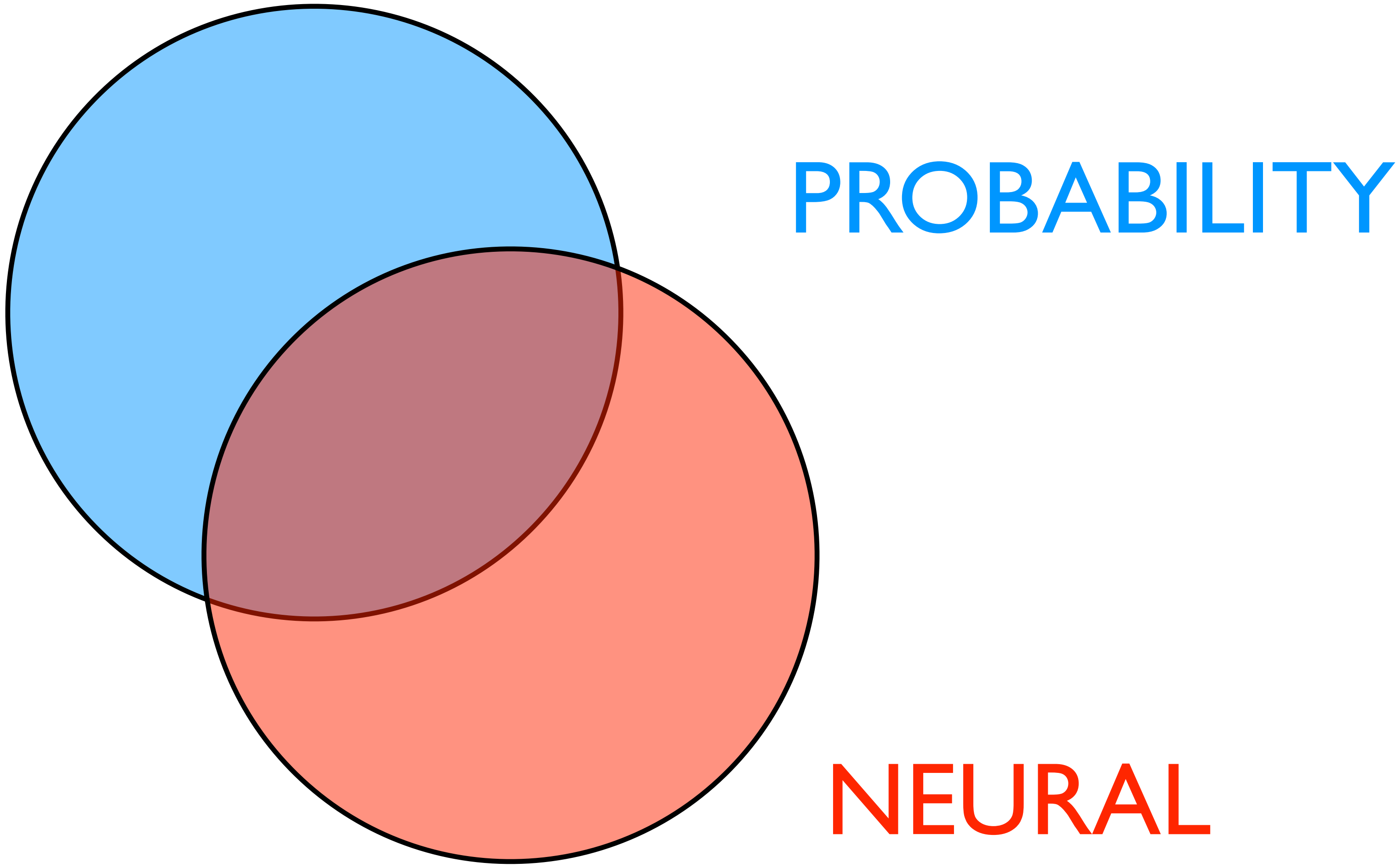
PROBABILITY

LOGIC

NEURAL

How to integrate these three paradigms in AI ?

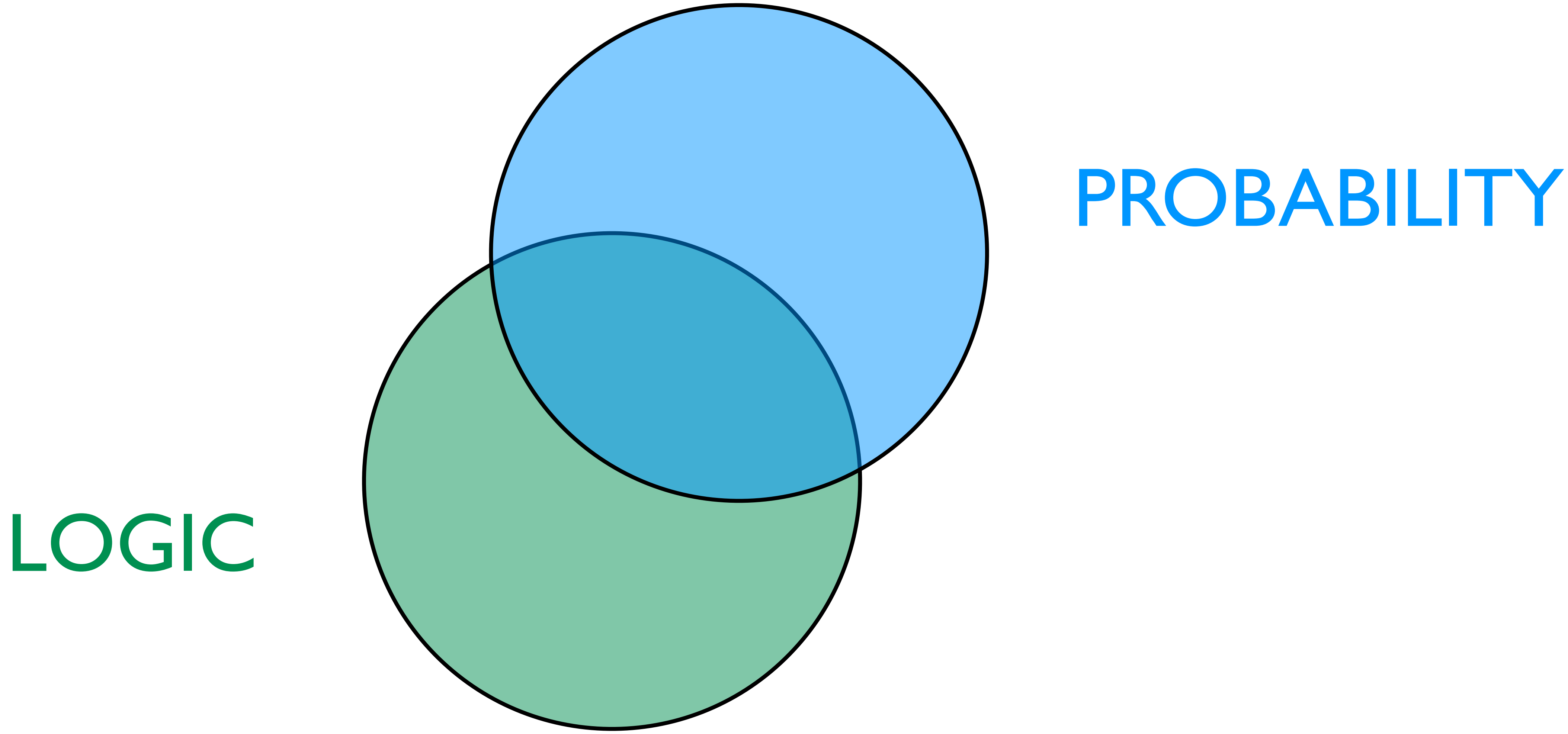
A lot of ML



Well studied from a LEARNING perspective in Deep Learning

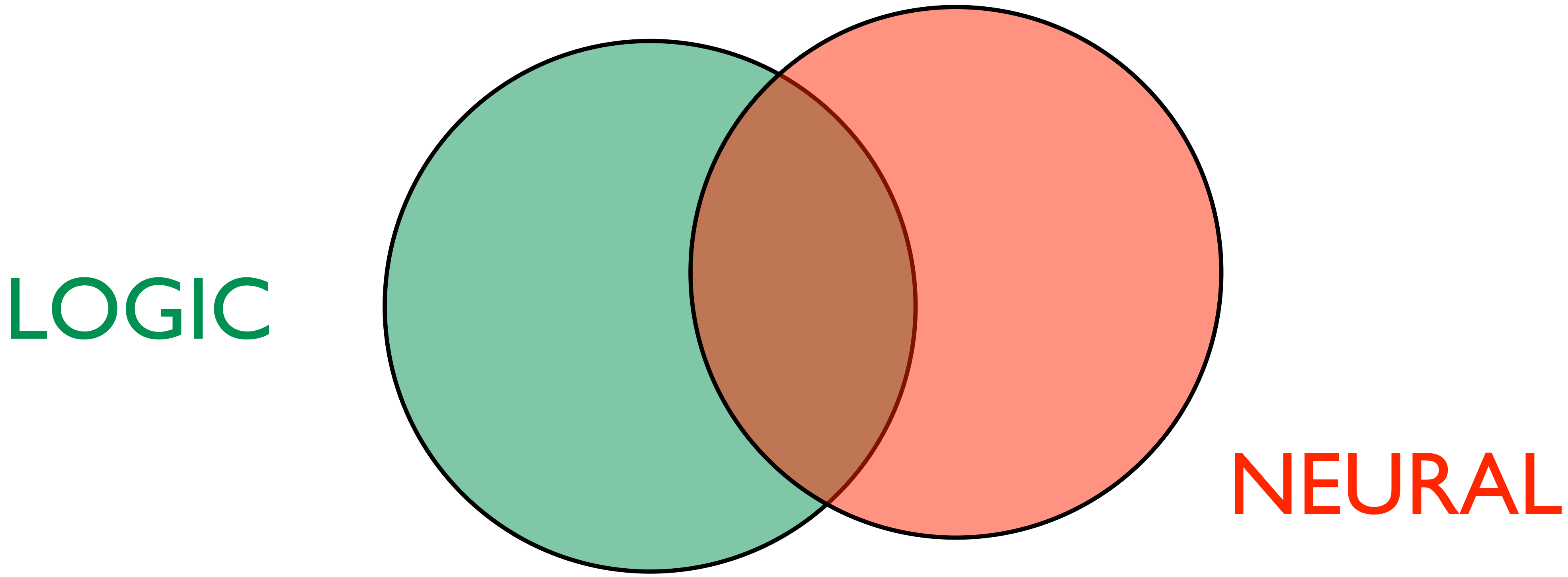
Thinking slow = reasoning

TWO MAIN PARADIGMS in AI

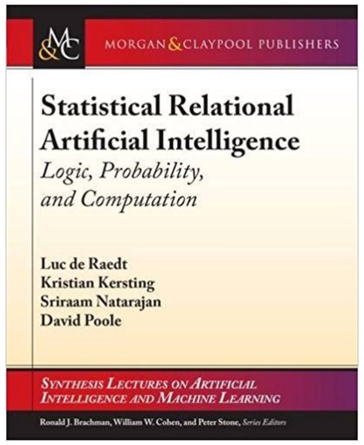


Their integration has been well studied in
Probabilistic (Logic) Programming and Statistical Relational AI (StarAI)

State of the Art



Being studied from a LEARNING perspective in Neuro Symbolic Computation



MORGAN & CLAYPOOL PUBLISHERS

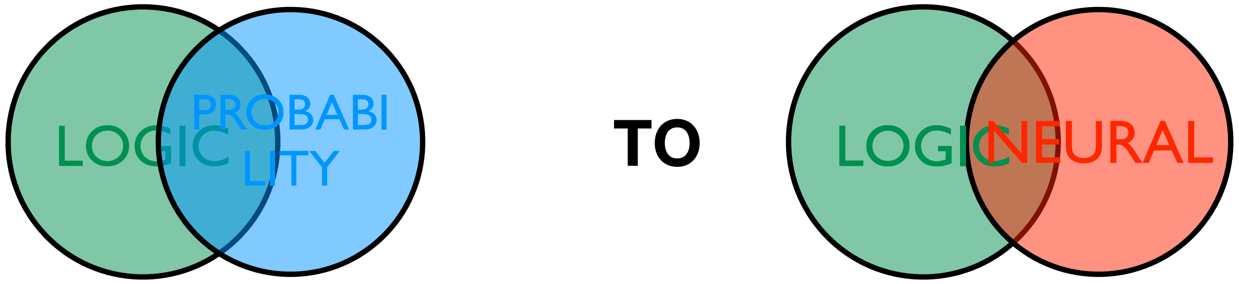
Statistical Relational Artificial Intelligence
Logic, Probability, and Computation

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning
Ronald J. Brachman, William W. Cohen, and Peter Stone, *Series Editors*

Key Message

FROM **TO**



StarAI and NeSy share similar problems and thus similar solutions apply

WARNING

TALK MAY NOT COVER ALL of NESY

See also
De Raedt, Dumancic, Marra, Manhaeve
From Statistical Relational to Neuro-Symbolic Artificial Intelligence
IJCAI 20

Applications

Feedback in two directions

- Logic can help neural networks to use external knowledge:
  - Better performance
  - Less data

- Neural networks can help logic-based systems to explore combinatorial spaces more efficiently.

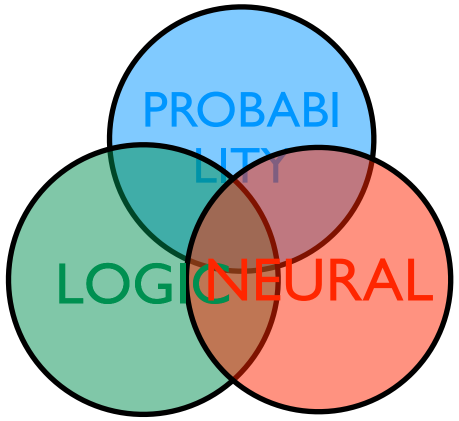
Addition

Learn to add the sum of lists of MNIST images

35 962

example multi-addition predicate

Assume you do not know how to map MNIST images to numbers, but do know the rules of addition. Can you lean from these examples how to map MNIST to numbers ?



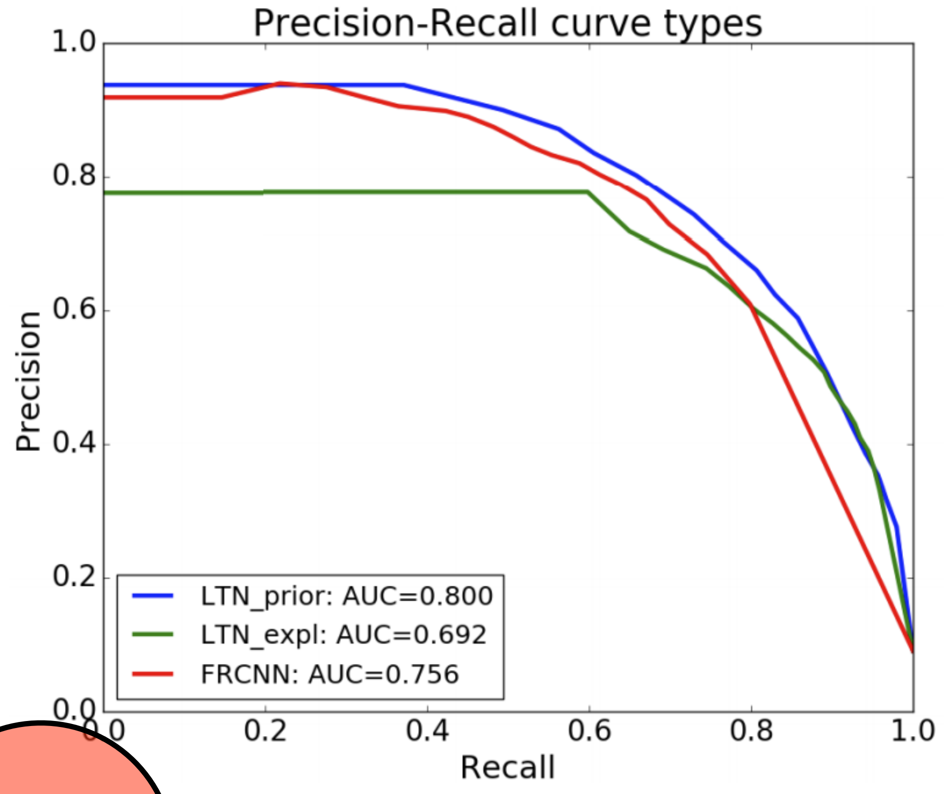
DeepProbLog, Manhaeve et al, NeurIPS 2018

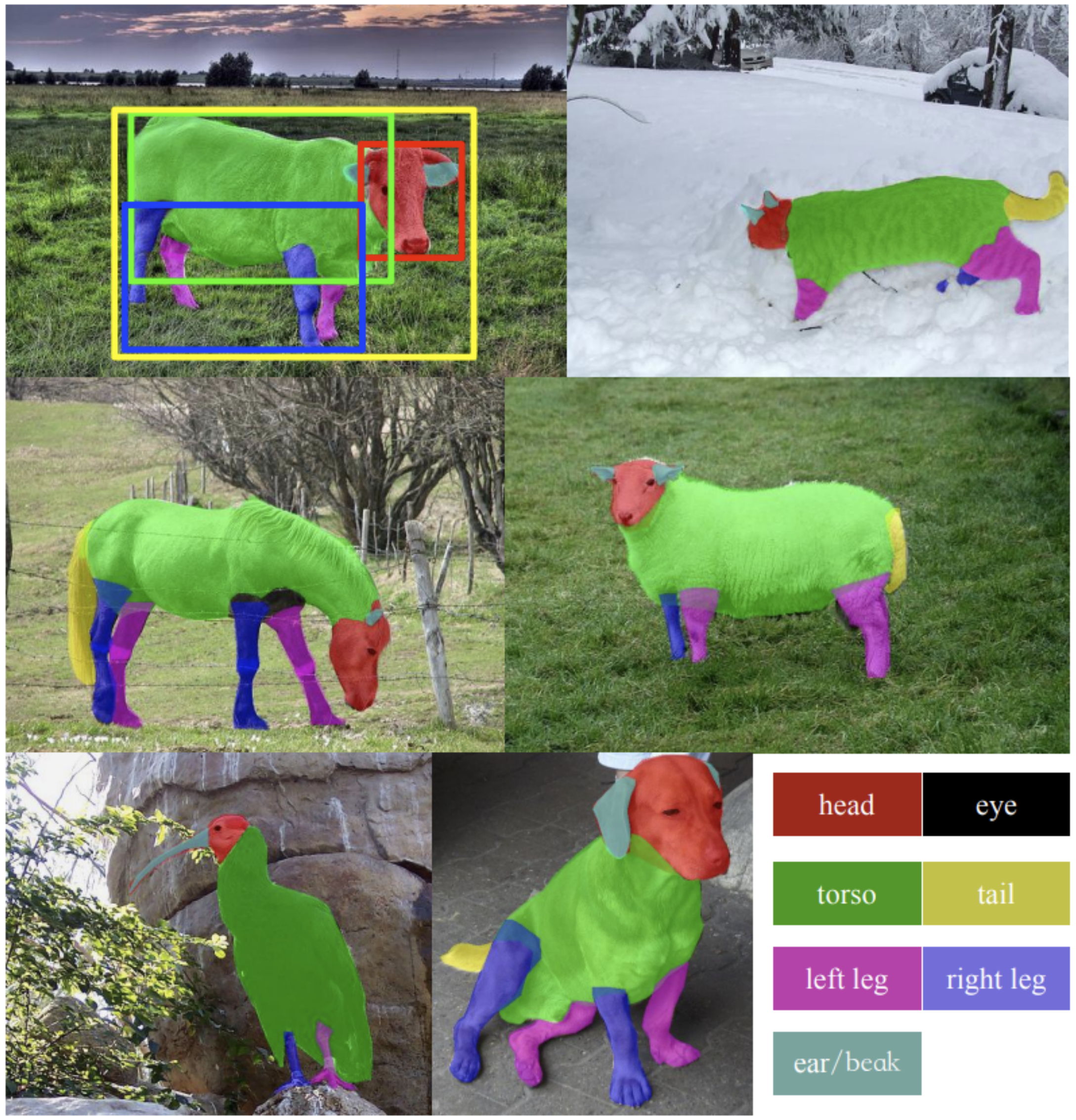
Semantic Image Interpretation

$$\forall xy(\mathsf{partOf}(x,y) \rightarrow \neg\mathsf{partOf}(y,x))$$

$$\forall xy(\mathsf{Cat}(x) \wedge \mathsf{partOf}(x,y) \rightarrow \mathsf{Tail}(y) \vee \mathsf{Muzzle}(y))$$

$$\forall xy(\mathsf{Cat}(x) \rightarrow \neg\mathsf{partOf}(x,y))$$





LTN, Serafini et al , NeSY@HLAI 2016

Visual Reasoning

How many blocks are on the right of the three-level tower?

Will the block tower fall if the top block is removed?

What is the shape of the object closest to the large cylinder?

Are there more trees than animals?

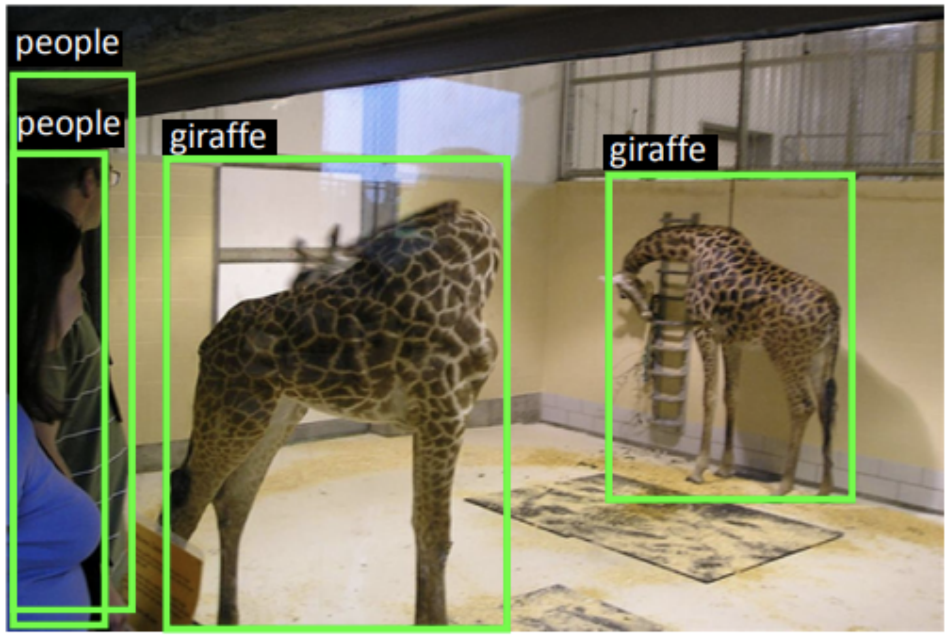
Figure 1: Human reasoning is interpretable and disentangled: we first draw abstract knowledge of the scene via visual perception and then perform logic reasoning on it. This enables compositional, accurate, and generalizable reasoning in rich visual contexts.

Adding a reasoning component on top of the perception can improve performance.

NS-VQA, Yi et al , NeurIPS 2019

Visual Reasoning

One can also add ontological knowledge.



Attributes:
glass
house
room
standing
walking
wall
zoo

Scenes:
museum
indoor

Visual Question: How many giraffes in the image?
Answer: Two. Reason: Two giraffes are detected.

Common-Sense Question: Is this image related to zoology?
Answer: Yes. Reason: Object/Giraffe --> Herbivorous animals --> Animal --> Zoology; Attribute/Zoo --> Zoology.

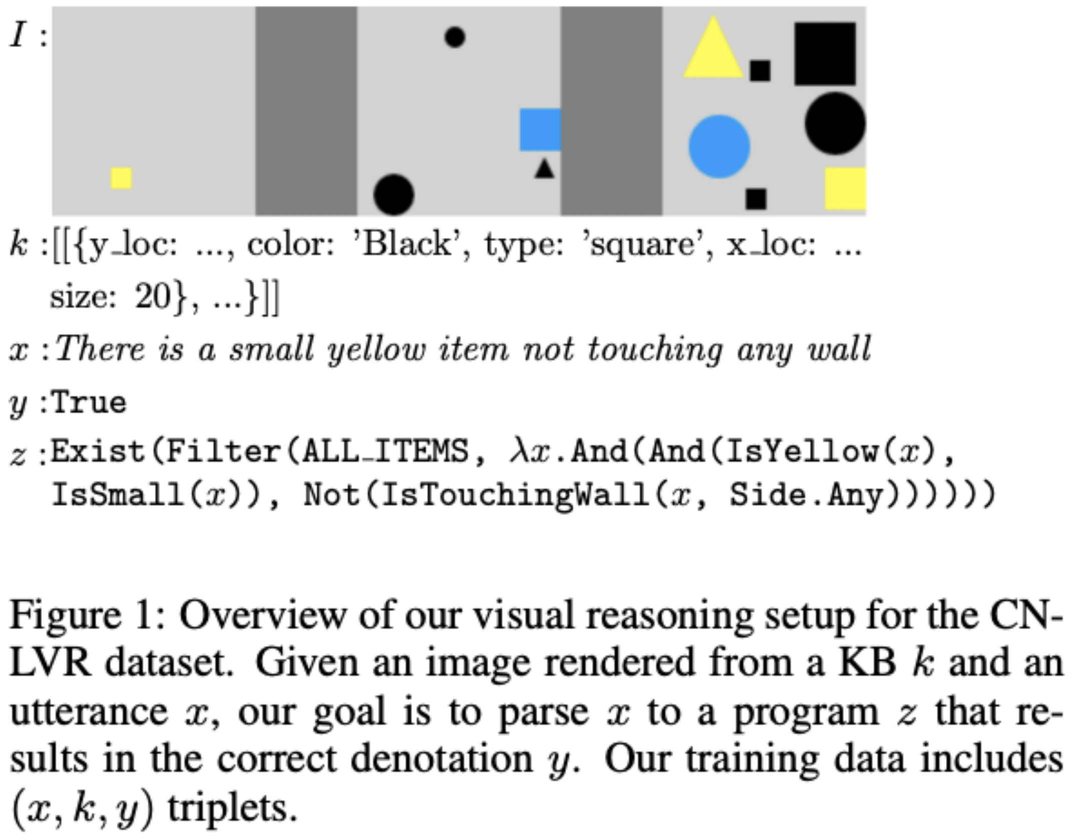
KB-Knowledge Question: What are the common properties between the animal in this image and the zebra?
Answer: Herbivorous animals; Animals; Megafauna of Africa.

Wang et al, IJCAI 2017

Program Induction from image and language

Goldman et al, ACL 2018

Adding an intermediate symbolic representation helps generalization



I :

k :[[{y_loc: ..., color: 'Black', type: 'square', x_loc: ... size: 20}, ...}]]

x :*There is a small yellow item not touching any wall*

y :True

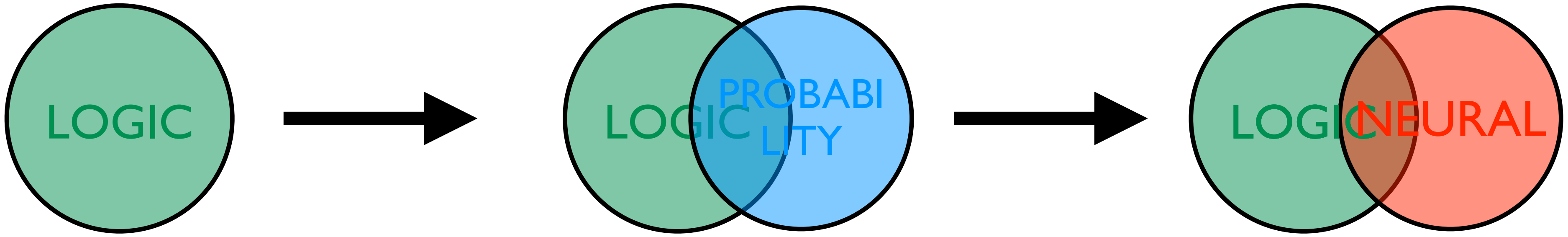
z :Exist(Filter(ALL_ITEMS, λx.And(And(IsYellow(x), IsSmall(x)), Not(IsTouchingWall(x, Side.Any)))))

Figure 1: Overview of our visual reasoning setup for the CNLVR dataset. Given an image rendered from a KB k and an utterance x, our goal is to parse x to a program z that results in the correct denotation y. Our training data includes (x, k, y) triplets.

The Seven Dimensions

1. Proof vs Model based
2. Directed vs Undirected
3. Type of Logic
4. Symbols vs Subsymbols
5. Parameter vs Structure Learning
6. Semantics
7. Logic vs Probability vs Neural

1. Proof vs Model based



1. Proof vs Model based

LOGIC

1. Proof vs Model based the logic dimension

- Model- vs proof-based
- First order / relational vs propositional
- Grounding
- Differences important for both StarAI and NeSY

Logic Programs

as in the programming language Prolog

Propositional logic program

```
burglary.
hears_alarm_mary.


earthquake.
hears_alarm_john.

alarm :– earthquake.
alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.
calls_john :– alarm, hears_alarm_john.
```

facts :
burglary = true

LOGIC

Logic Programs

as in the programming language Prolog

Propositional logic program

```
burglary.
hears_alarm_mary.


earthquake.
hears_alarm_john.


alarm :– earthquake.

alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.
calls_john :– alarm, hears_alarm_john.
```

rule:

calls_mary =true IF alarm = true AND hears_alarm_mary = true

LOGIC

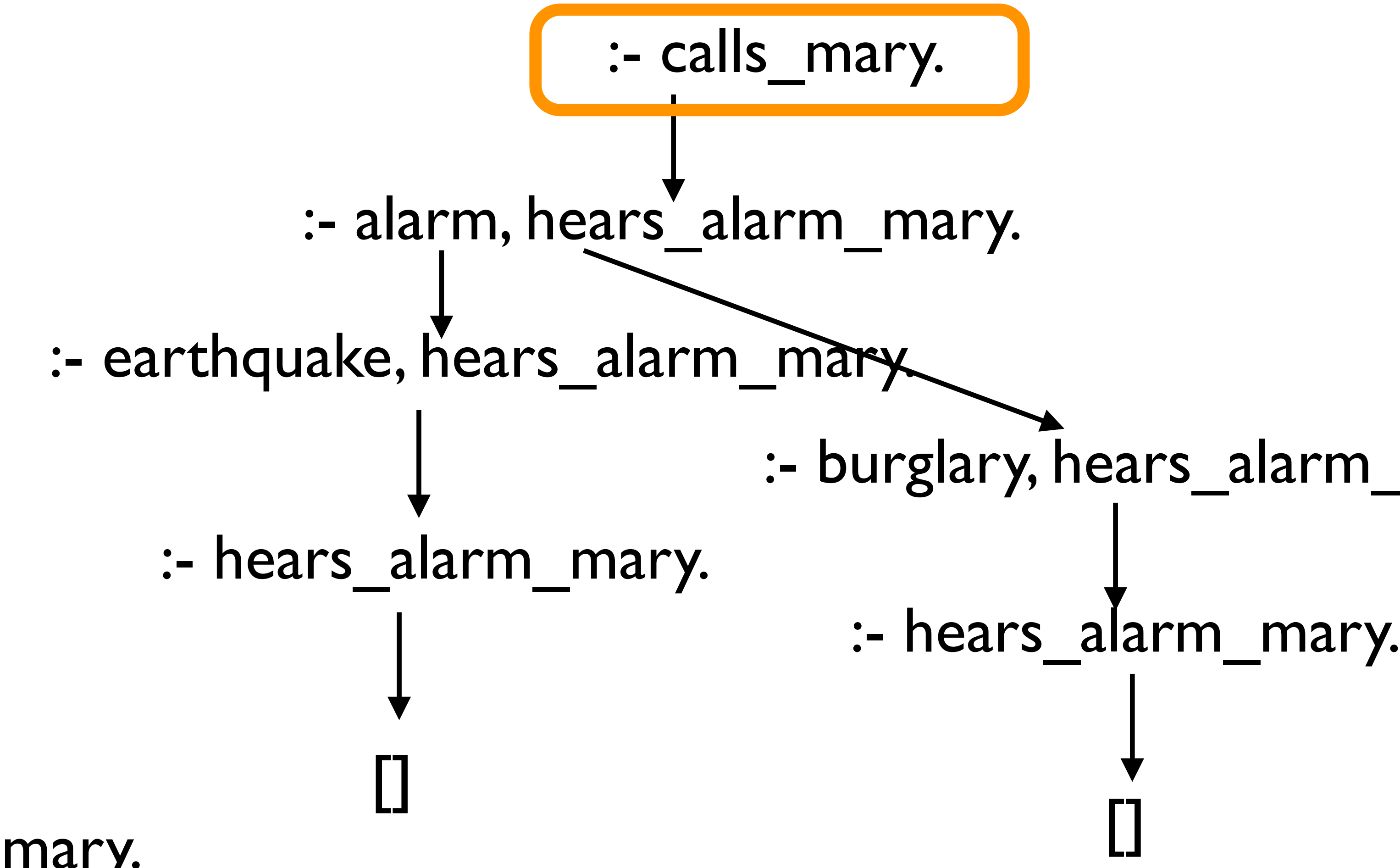
Logic Programs

as in the programming language Prolog

Propositional logic program

```
burglary.
hears_alarm_mary.

earthquake.
hears_alarm_john.

alarm :– earthquake.
alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.
calls_john :– alarm, hears_alarm_john.
```

Two proofs (by refutation)



:- calls_mary.

:- alarm, hears_alarm_mary.

:- earthquake, hears_alarm_mary.

:- burglary, hears_alarm_

:- hears_alarm_mary.

[]

:- hears_alarm_mary.

[]

A proof-theoretic view

backward chaining

LOGIC

Logic as constraints

as in SAT solvers

Propositional logic

Model / Possible World

IF AND

calls(mary)←hears_alarm(mary) ∧ alarm

calls(john) ← hears_alarm(john) ∧ alarm

OR

alarm ← earthquake v burglary

{ burglary,

hears_alarm(john),

alarm,

calls(john)}

the facts that are true
in this model / possible world

SAT: Find a model / possible world that satisfies all the constraints
SAT SOLVERS

A model-theoretic view

LOGIC

Relational/First Order Logic

Introduce Variables and Domains
The meaning of this is always the GROUNDED theory

allows to exploit symmetries / templates …

burglary.
hears_alarm(mary).

earthquake.
hears_alarm(john).

alarm :– earthquake.

alarm :– burglary.
calls(X) :– alarm, hears_alarm(X).

Variable X
Domain = {mary, john}

burglary.
hears_alarm(mary).

earthquake.
hears_alarm(john).

alarm :– earthquake.

alarm :– burglary.
calls(mary) :– alarm, hears_alarm(mary).

calls(john) :– alarm, hears_alarm(john).

Grounded Theory

LOGIC

BOTH for model and proof-based appraoch

Logical Theory

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

GROUNDING OUT

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(ann) :- stress(ann).
smokes(bob) :- stress(bob).
smokes(carl) :- stress(carl).

smokes(ann) :- influences(ann,ann), smokes(ann).
smokes(ann) :- influences(bob,ann), smokes(bob).
smokes(ann) :- influences(carl,ann), smokes(carl).

smokes(bob) :- influences(ann,bob), smokes(ann).
smokes(bob) :- influences(bob,bob), smokes(bob).
smokes(bob) :- influences(carl,bob), smokes(carl).

smokes(carl) :- influences(ann,carl), smokes(ann).
smokes(carl) :- influences(bob,carl), smokes(bob).
smokes(carl) :- influences(carl,carl), smokes(carl).
```

IF INTERESTED ONLY IN CERTAIN QUERIES, CLEVER TECHNIQUES EXIST TO AVOID GROUNDING OUT COMPLETELY

Logical Reasoning: Model Theoretic

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

FINDING A MODEL

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(ann) :- stress(ann).
-> infer smokes(ann)

smokes(bob) :- influences(ann,bob), smokes(ann)
-> infer smokes(bob)

smokes(carl) :- influences(bob,carl), smokes(bob).
-> infer smokes(carl).
```

FINDING A MODEL

here — the least Herbrand model as in Prolog using the Tp Operator (forward reasoning

Logical Reasoning: Model Theoretic

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
     influences(Y,X),
     smokes(Y).
```

Clark's completion AND call a SAT Solver

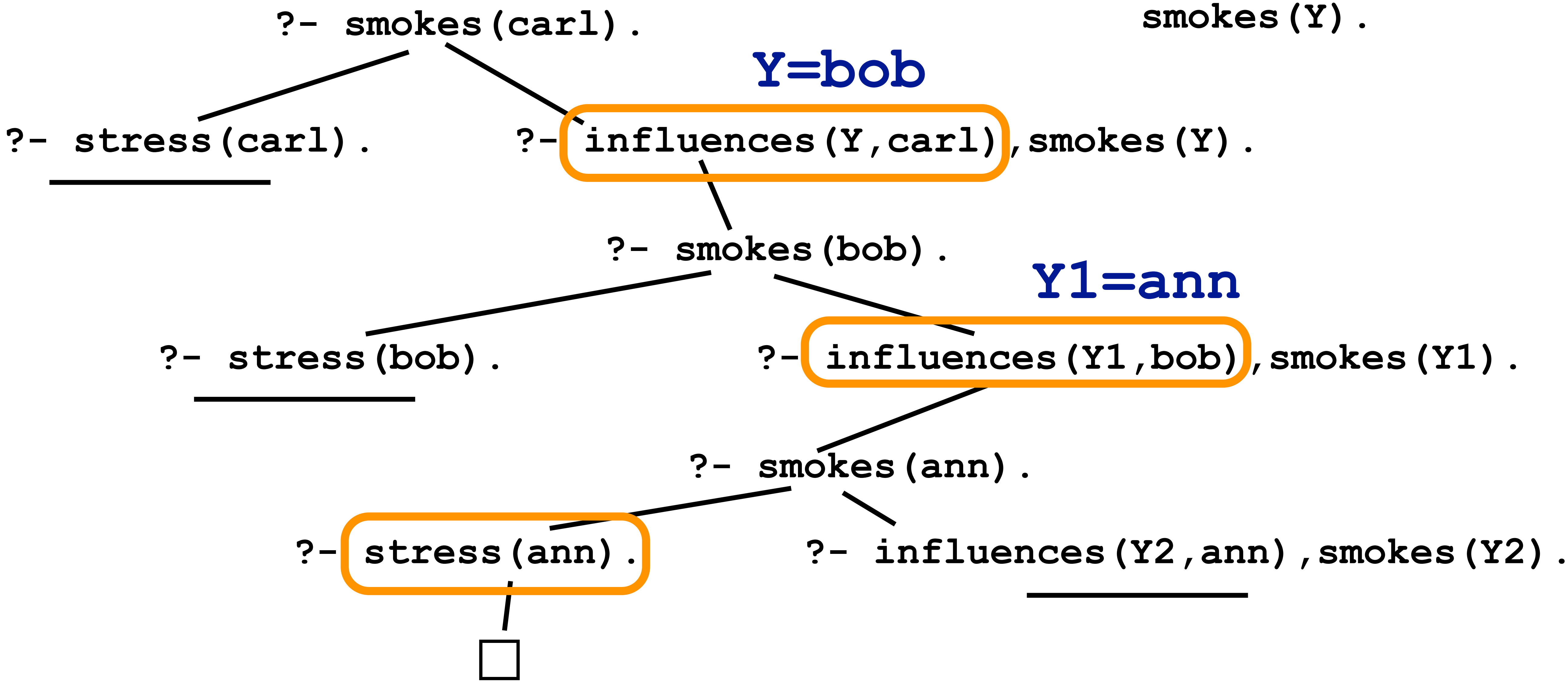
```
stress(ann).
influences(ann,bob).
influences(bob,carl).


smokes(ann) <-> stress(ann)
             v  influences(ann,ann), smokes(ann)
             v influences(bob,ann), smokes(bob)
             v influences(carl,ann), smokes(carl)

smokes(bob) <-> stress(bob)
             v influences(ann,bob), smokes(ann)
            v influences(bob,bob), smokes(bob)
            v influences(carl,bob), smokes(carl)

smokes(carl) <-> stress(carl)
             v influences(ann,carl), smokes(ann)
             v influences(bob,carl), smokes(bob)
             v influences(carl,carl), smokes(carl)
```

Clark's completion's as a grounding is incorrect for Prolog when there are cycles

but it is too hard to explain why here

Logical Reasoning Proofs

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```



```
?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).
?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
?- smokes(ann).
?- stress(ann).
?- influences(Y2,ann),smokes(Y2).
□
```

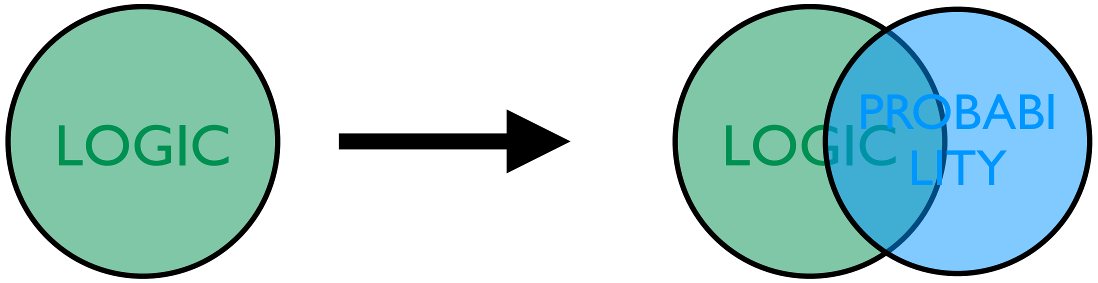
facts used in successful derivation:

`influences(bob,carl)&influences(ann,bob)&stress(ann)`

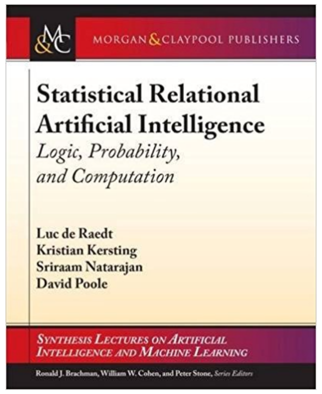
1. Proof vs Model based the logic dimension

- Model- vs proof-based
- First order / relational vs propositional
- Grounding
- Differences important for both StarAI and NeSY

1. Proof vs Model based
2. Directed vs Undirected



2. Directed vs Undirected the PGM / StarAI dimension



MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational Artificial Intelligence
Logic, Probability, and Computation

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning

0.1 :: burglary.

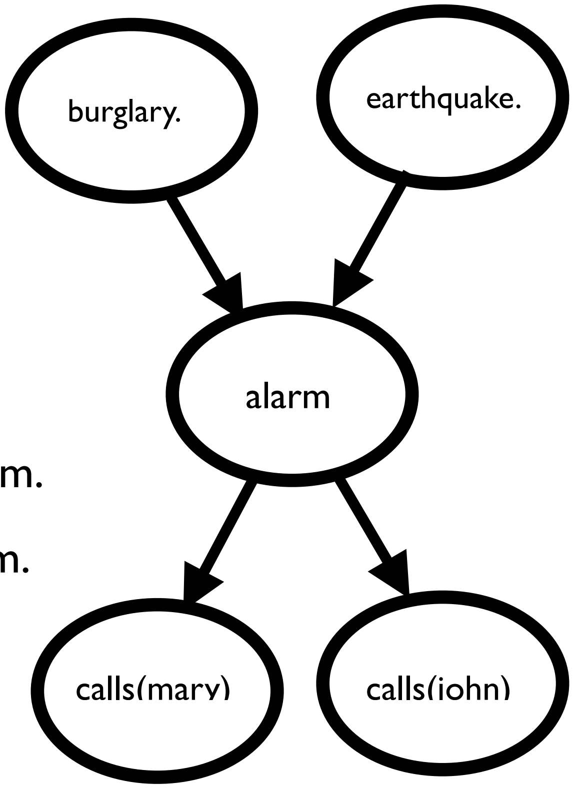
0.05 :: earthquake.

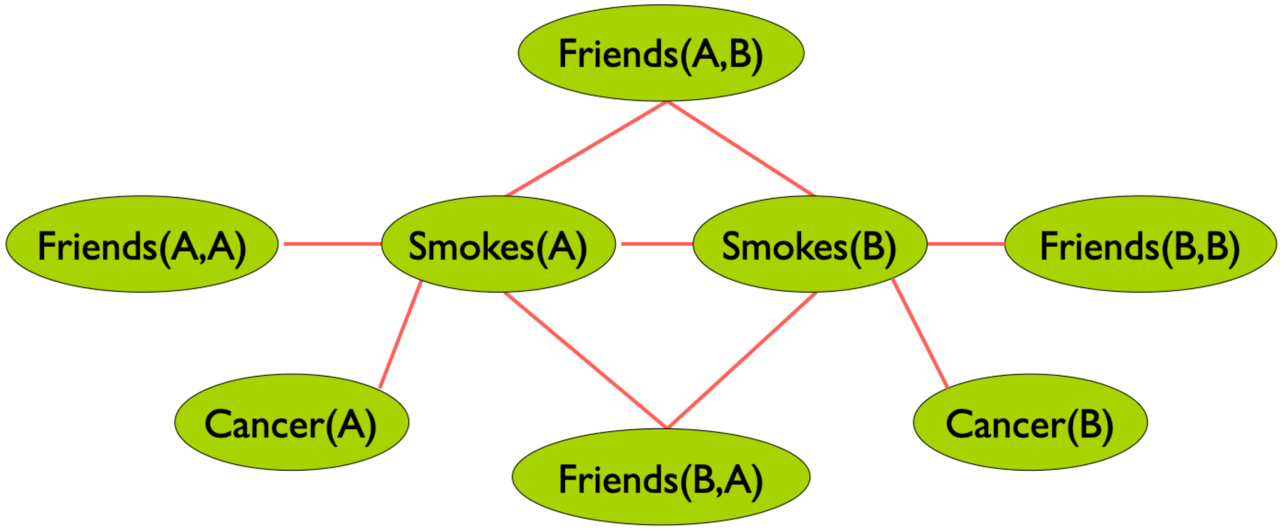
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.





$$1.5 \quad \forall x\ Smokes(x) \Rightarrow Cancer(x)$$

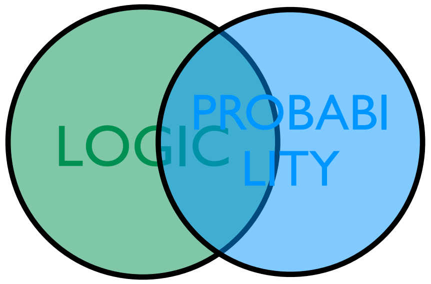
$$1.1 \quad \forall x, y\ Friends(x,y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

Markov Logic

undirected
Markov Net
model theoretic



key representatives

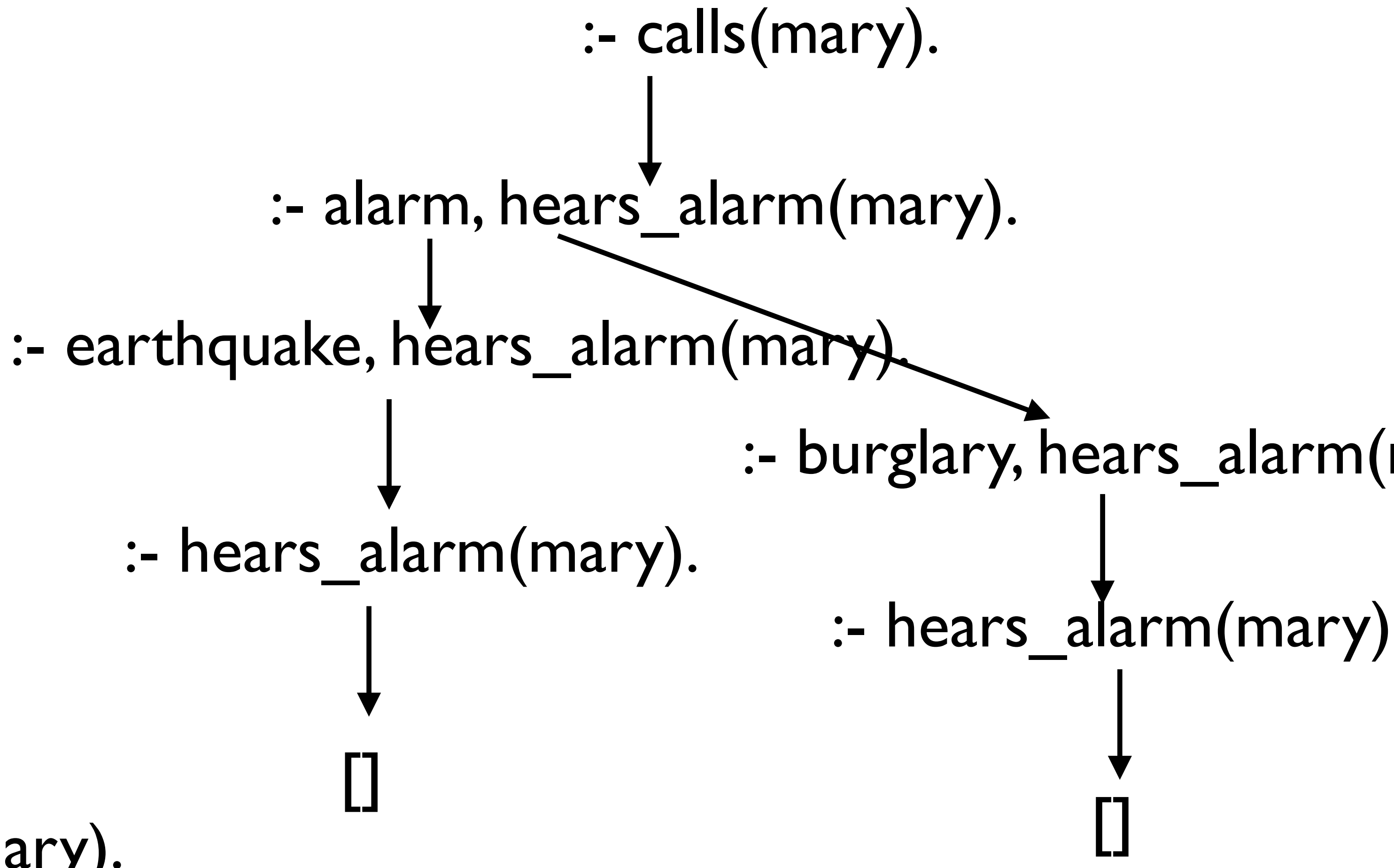
Logic Programs

as in the programming language Prolog

Propositional logic program

```
burglary.
hears_alarm(mary).

earthquake.
hears_alarm(john).

alarm :– earthquake.
alarm :– burglary.

calls(mary) :– alarm, hears_alarm(mary).
calls(john) :– alarm, hears_alarm(john).
```

Two proofs (by refutation)



:- calls(mary).

:- alarm, hears_alarm(mary).

:- earthquake, hears_alarm(mary).

:- burglary, hears_alarm(

:- hears_alarm(mary).

:- hears_alarm(mary)

[]

[]

A proof-theoretic view

LOGIC

Probabilistic Logic Programs

as in the probabilistic programming language ProbLog

Propositional logic program

```
0.1 :: burglary.
0.3 ::hears_alarm(mary).

0.05 ::earthquake.
0.6 ::hears_alarm(john).
```

Probabilistic facts

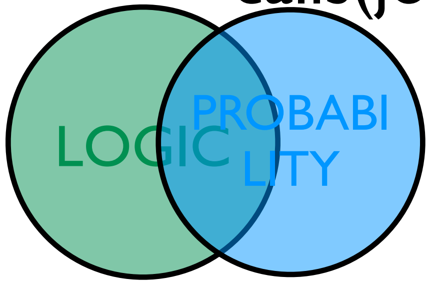
```
alarm :– earthquake.
alarm :– burglary.

calls(mary) :– alarm, hears_alarm(mary).
calls(john) :– alarm, hears_alarm(john).
```

Key Idea (Sato & Poole)
the distribution semantics:

unify the basic concepts in logic and probability:

random variable ~ propositional variable

an interface between logic and probability



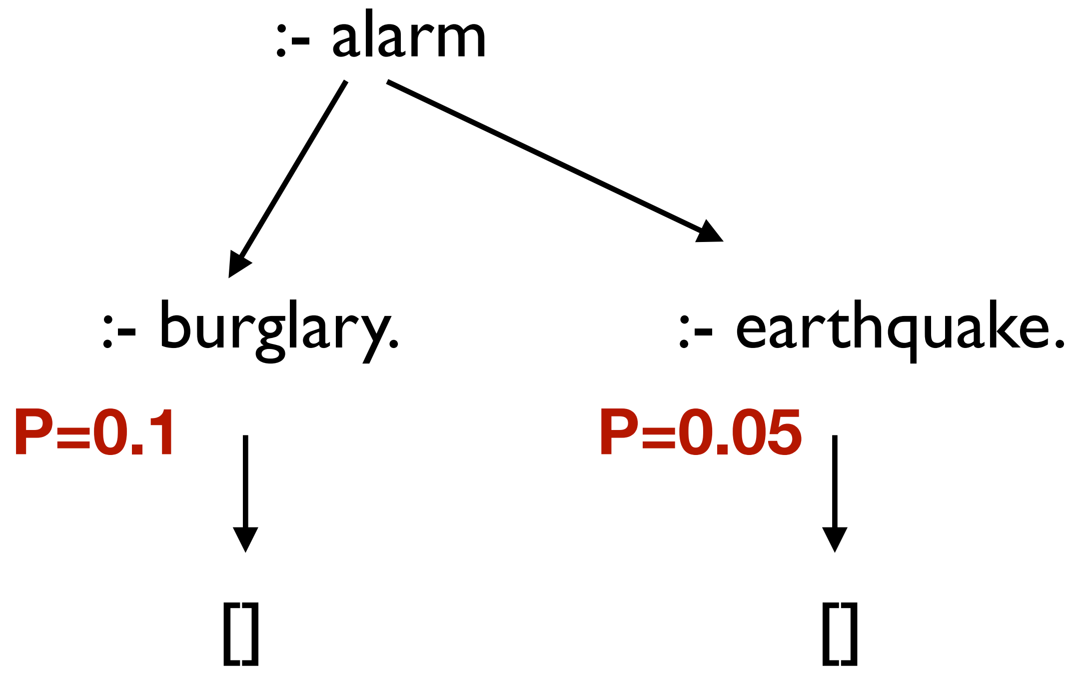
Probabilistic Logic Programs

as in the probabilistic programming language ProbLog

Propositional logic program

```
0.1 :: burglary.
0.3 ::hears_alarm(mary).


0.05 ::earthquake.
0.6 ::hears_alarm(john).


alarm :– earthquake.

alarm :– burglary.


calls(mary) :– alarm, hears_alarm(mary).

calls(john) :– alarm, hears_alarm(john).
```

Two proofs (by refutation)



Probability of one proof : $\prod_{f:fact\in Proof} P_f$

Probabilistic Logic Programs

as in the probabilistic programming language ProbLog

Propositional logic program

0.1 :: burglary.
0.3 ::hears_alarm(mary).

0.05 ::earthquake.
0.6 ::hears_alarm(john).

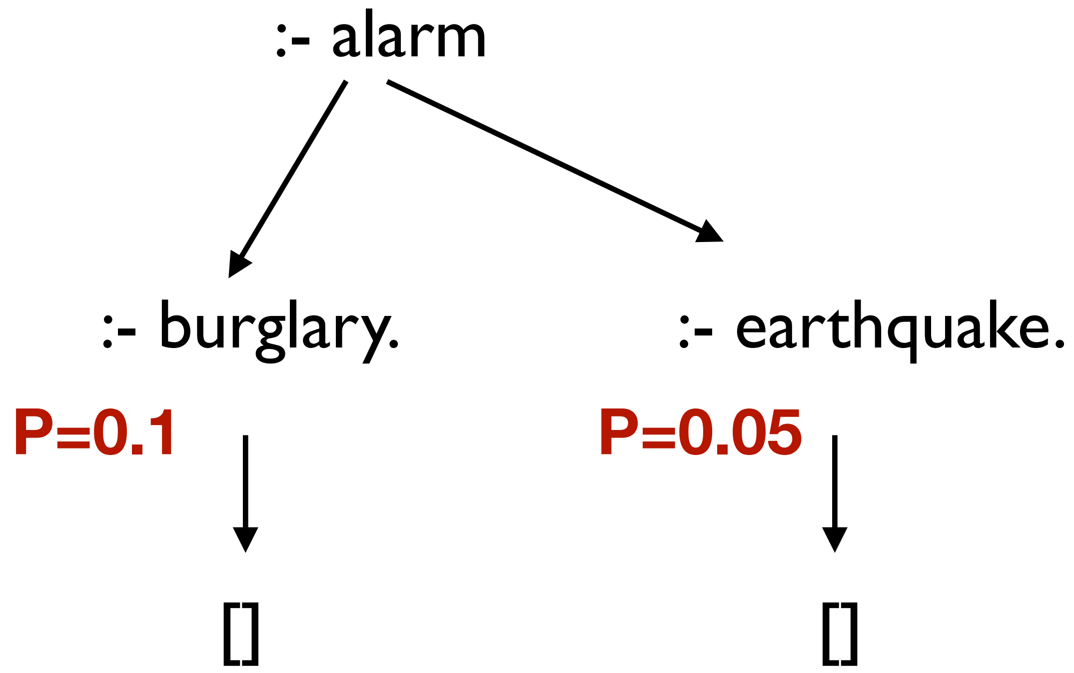
alarm :– earthquake.

alarm :– burglary.

calls(mary) :– alarm, hears_alarm(mary).

calls(john) :– alarm, hears_alarm(john).

Disjoint sum problem



:- alarm

:- burglary.

:- earthquake.

P=0.1

P=0.05

[]

[]

Probability of one proof : $\prod_{f:fact \in Proof} P_f$

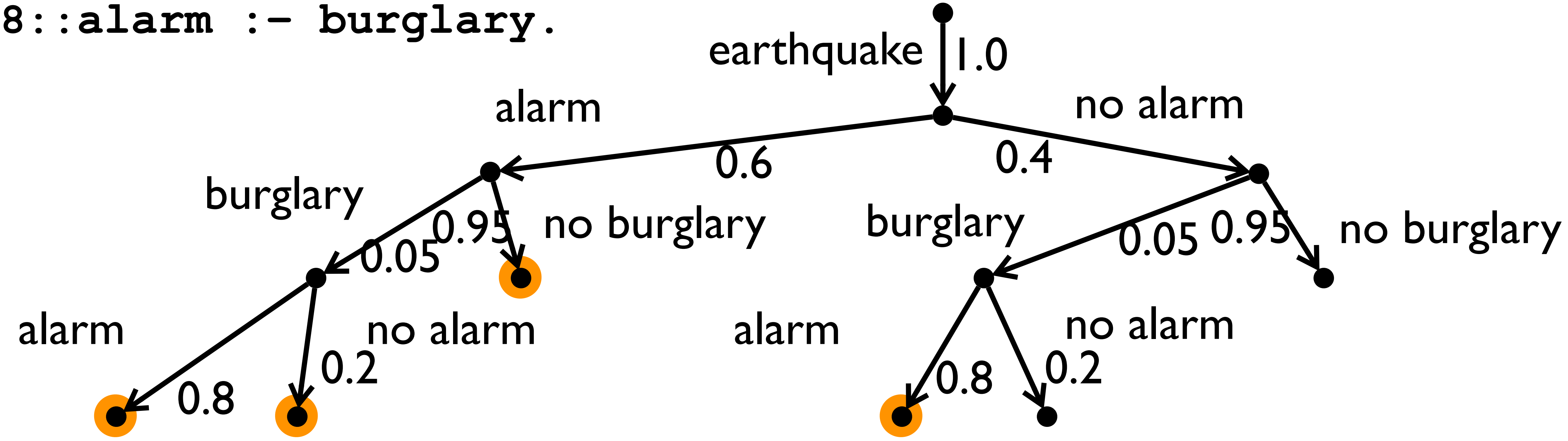
P(alarm) = P(burg OR earth)
= P(burg) + P(earth) - P(burg AND earth)
=/= P(burg) + P(earth)

Probabilistic Logic Program Semantics

[Vennekens et al, ICLP 04]

```
earthquake.
0.05::burglary.

0.6::alarm :- earthquake.
0.8::alarm :- burglary.
```

probabilistic causal laws



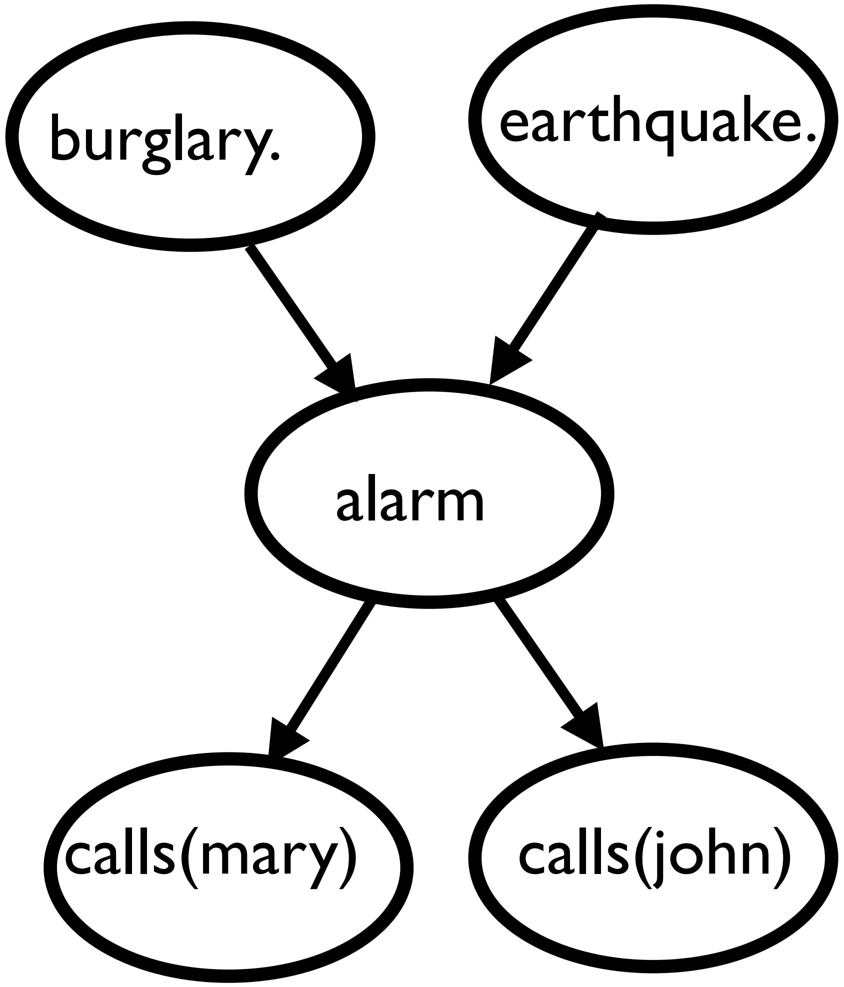
P(alarm)=0.6×0.05×0.8+0.6×0.05×0.2+0.6×0.95+0.4×0.05×0.8

Probabilistic Logic Program Semantics

Propositional logic program

```
0.1 :: burglary.
0.05 :: earthquake.

alarm :– earthquake.
alarm :– burglary.

0.7::calls(mary) :– alarm.
0.6::calls(john) :– alarm.
```

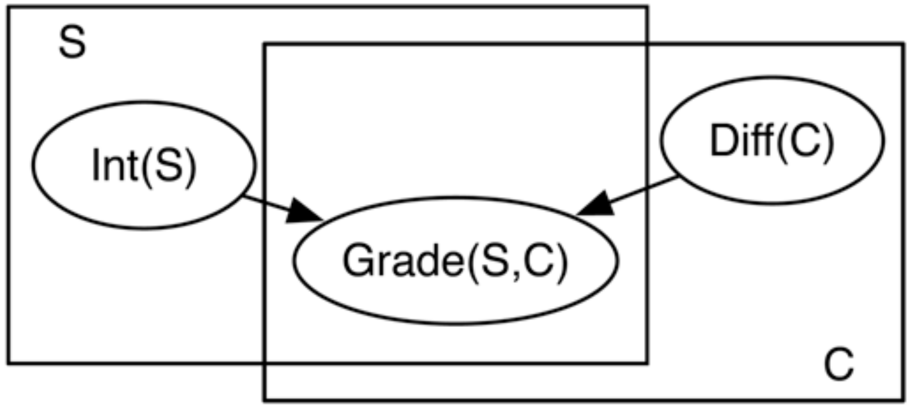
Bayesian Network



Bayesian net encoded as Probabilistic Logic Program
PLPs correspond to directed graphical models

ProbLog has both (directed) probabilistic graphic models, the programming language Prolog (and probabilistic databases) as special case

Flexible and Compact Relational Model for Predicting Grades



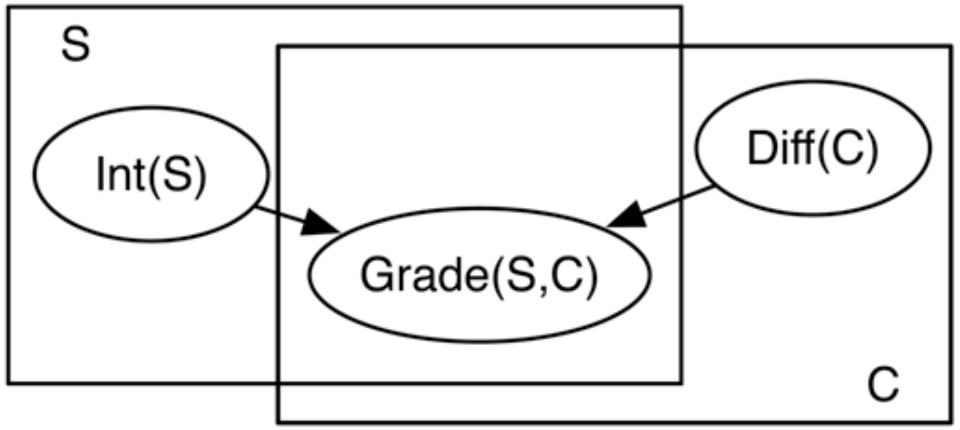
"Program" Abstraction:

- S, C logical variable representing students, courses
- the set of individuals of a type is called a population
- Int(S), Grade(S, C), D(C) are parametrized random variables

Grounding:

- for every student s, there is a random variable Int(s)
- for every course c, there is a random variable Di(c)
- for every s, c pair there is a random variable Grade(s,c)
- all instances share the same structure and parameters

ProbLog by example: Grading



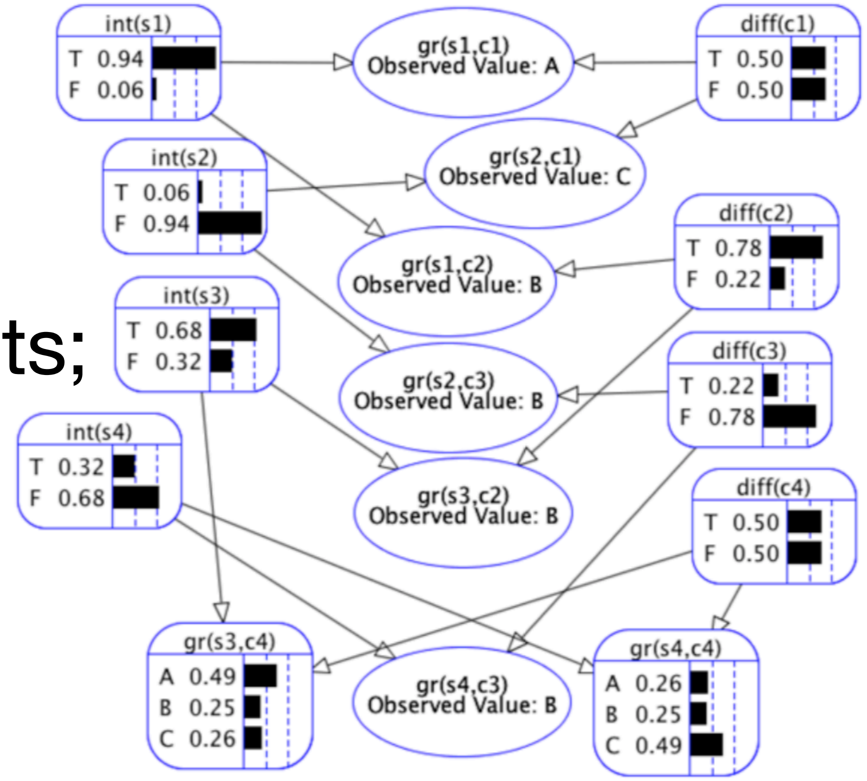
Shows relational structure

- grounded model: replace variables by constants

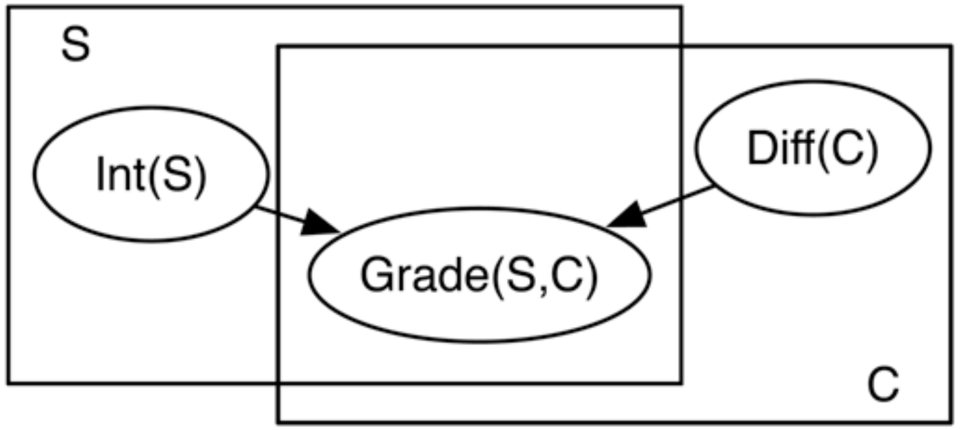
Works for any number of students / classes (for 1000 students and 100 classes, you get 101100 random variables); still only few parameters

With SRL / PP

- build and learn compact models,
- from one set of individuals - > other sets;
- reason also about exchangeability,
- build even more complex models,
- incorporate background knowledge



ProbLog by example: Grading



Shows relational structure

- grounded model: replace variables by constants

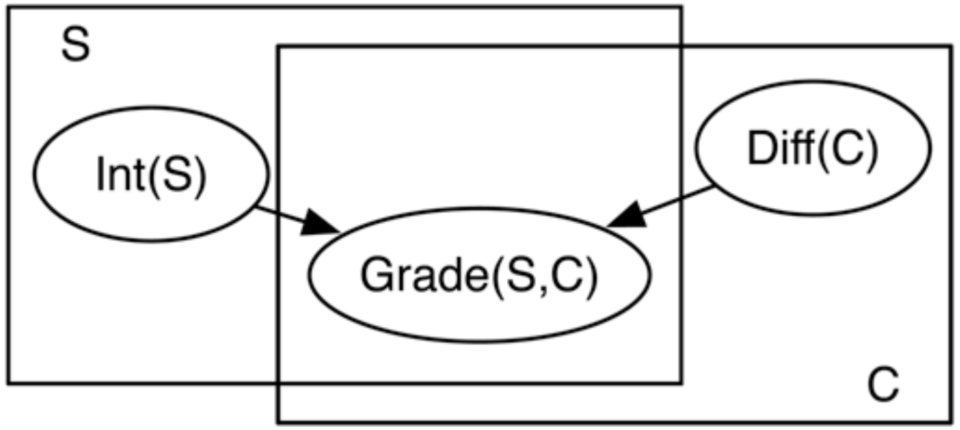
Works for any number of students / classes (for 1000 students and 100 classes, you get 101100 random variables); still only few parameters

With SRL / PP

- build and learn compact models,
- from one set of individuals - > other sets;
- reason also about exchangeability,
- build even more complex models,
- incorporate background knowledge

Student	*Course*	*Grade*
s_1	c_1	A
s_2	c_1	C
s_1	c_2	B
s_2	c_3	B
s_3	c_2	B
s_4	c_3	B
s_3	c_4	?
s_4	c_4	?

ProbLog by example: Grading

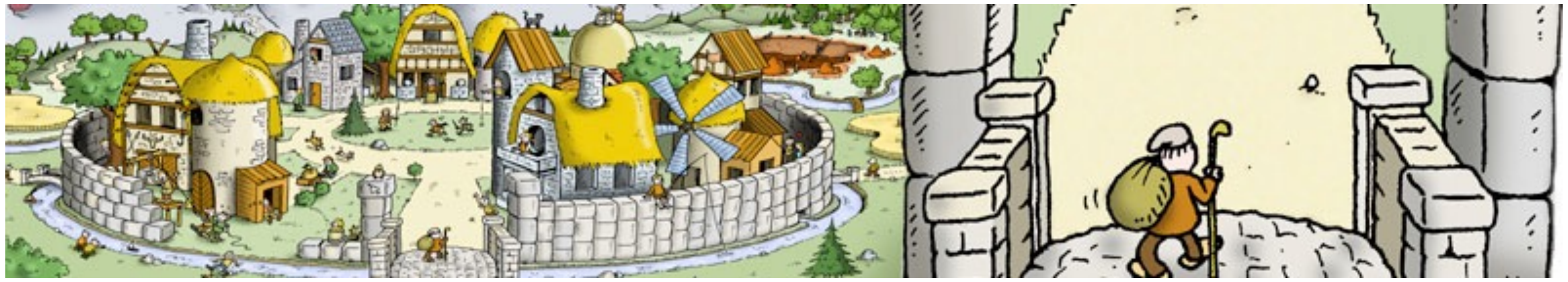


```
0.4 :: int(S) :- student(S).
0.5 :: diff(C):- course(C).

student(john). student(anna). student(bob).
course(ai).    course(ml).    course(cs).

gr(S,C,a) :- int(S), not diff(C).
0.3::gr(S,C,a); 0.5::gr(S,C,b);0.2::gr(S,C,c) :-
           int(S), diff(C).
0.1::gr(S,C,b); 0.2::gr(S,C,c); 0.2::gr(S,C,f) :-
           student(S), course(C),
           not int(S), not diff(C).
0.3::gr(S,C,c); 0.2::gr(S,C,f) :-
           not int(S), diff(C).
```

ProbLog by example: Grading

```
unsatisfactory(S) :- student(S), grade(S,C,f).

excellent(S):- student(S), not(grade(S,C1,G),below(G,a)),
               grade(S,C2,a).
0.4 :: int(S) :- student(S).
0.5 :: diff(C):- course(C).

student(john). student(anna). student(bob).
course(ai).    course(ml).    course(cs).

gr(S,C,a) :- int(S), not diff(C).
0.3::gr(S,C,a); 0.5::gr(S,C,b);0.2::gr(S,C,c) :-
           int(S), diff(C).
0.1::gr(S,C,b); 0.2::gr(S,C,c); 0.2::gr(S,C,f) :-
           student(S), course(C),
           not int(S), not diff(C).
0.3::gr(S,C,c); 0.2::gr(S,C,f) :-
           not int(S), diff(C).
```

Dynamic networks



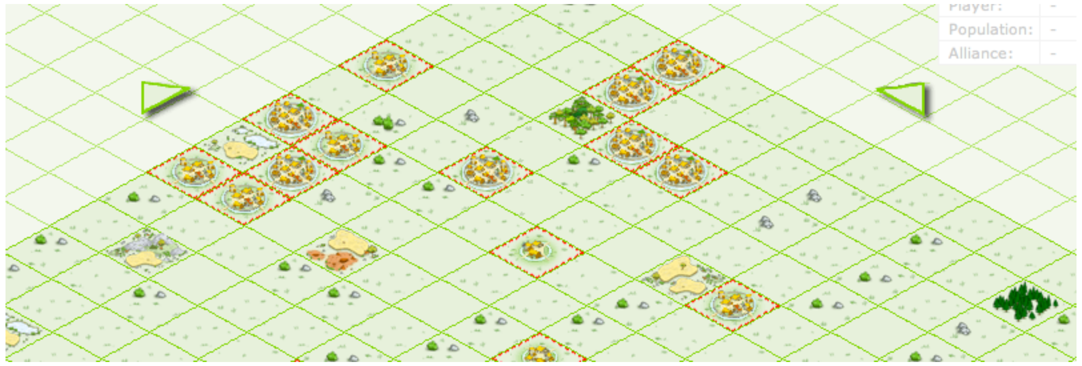
Travian: A massively multiplayer real-time strategy game

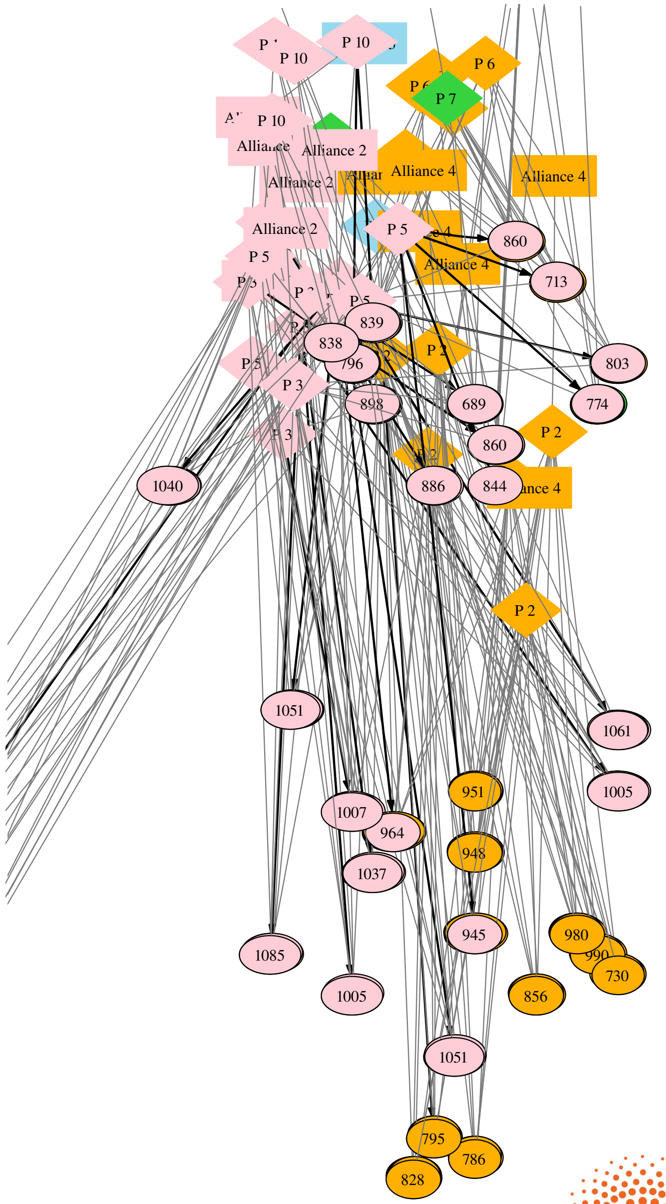
Can we build a model

of this world ?

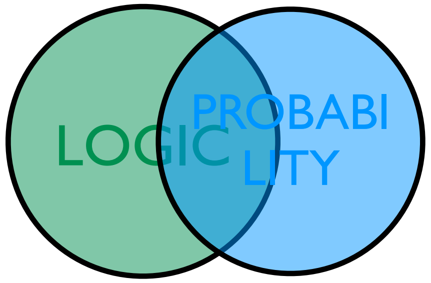
Can we use it for playing

better ?

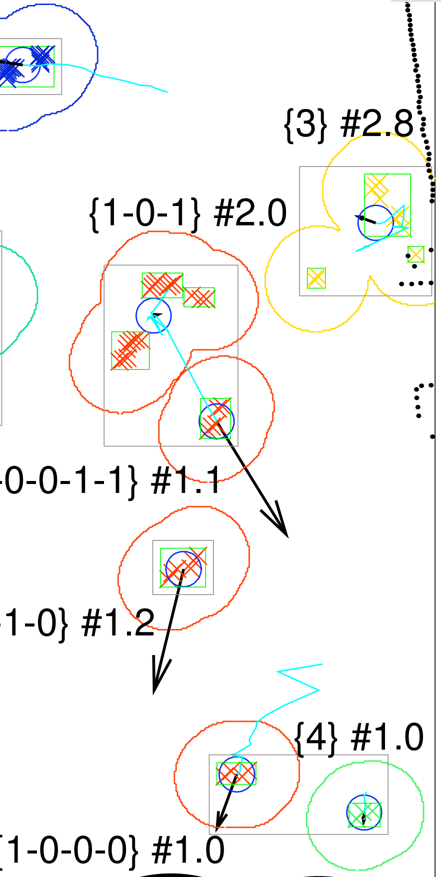




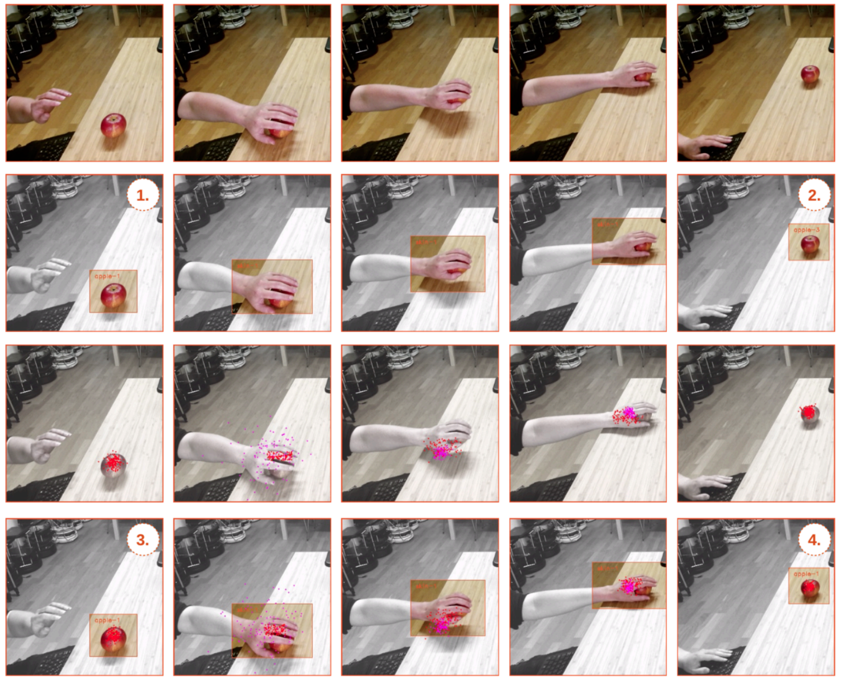
[Thon et al, MLJ 11]



Activity analysis and tracking video analysis

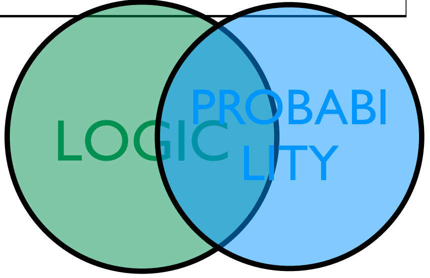


- Track people or objects over time? Even if temporarily hidden?
- Recognize activities?
- Infer object properties?

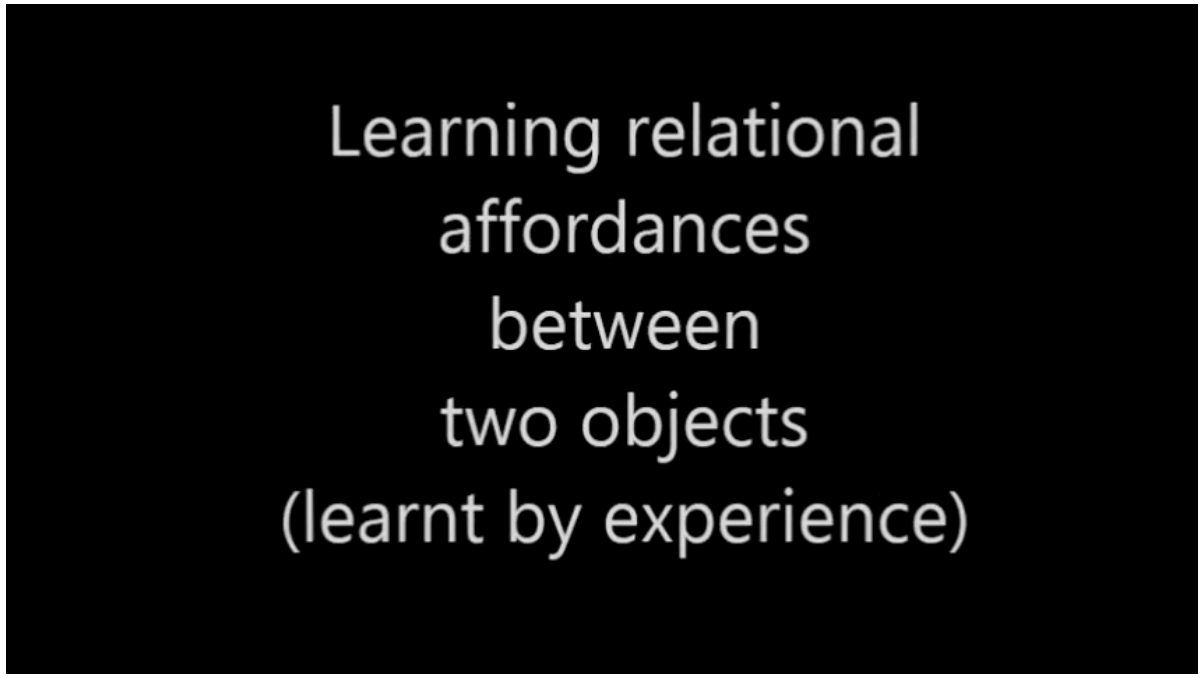


[Skarlatidis et al, TPLP 14; Nitti et al, IROS 13, ICRA 14, MLJ 16]

[Persson et al, IEEE Trans on Cogn. & Dev. Sys. 19; IJCAI 20]

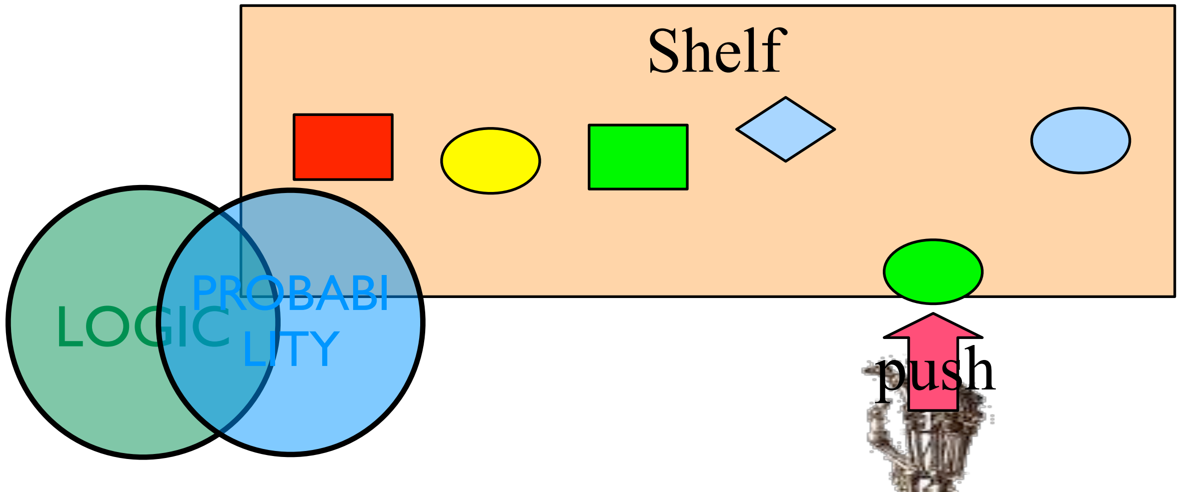


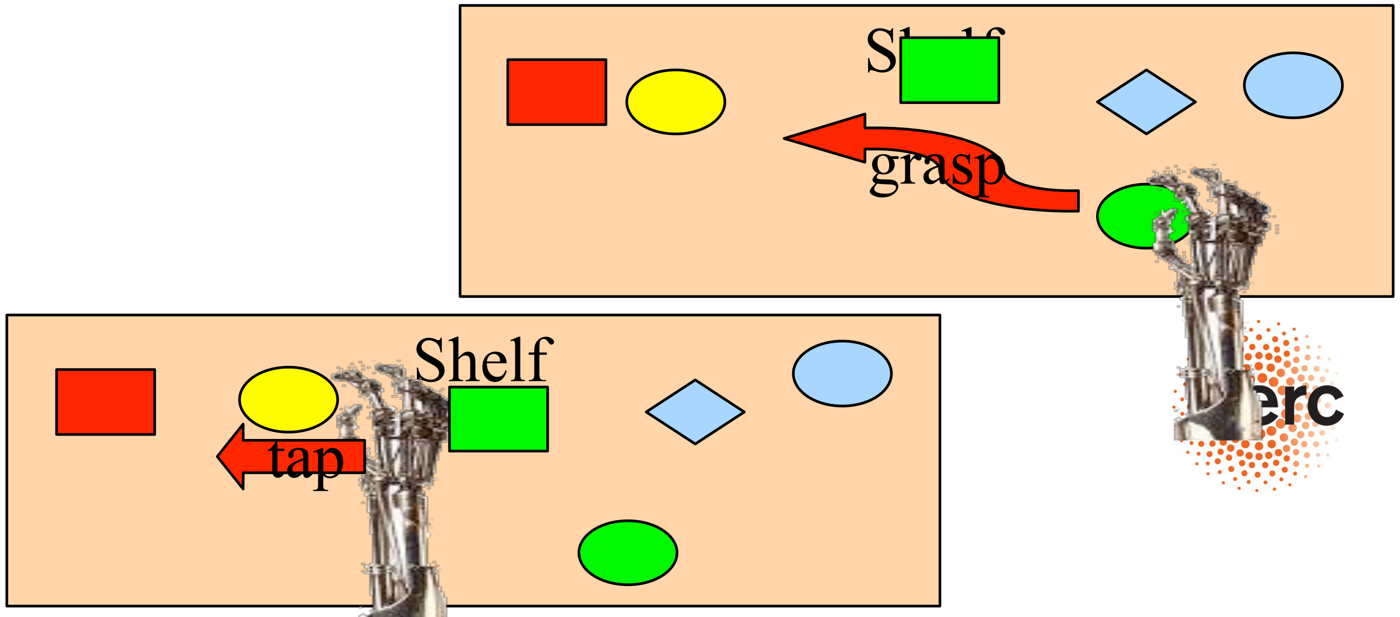
Learning relational affordances


Learning relational
affordances
between
two objects
(learnt by experience)

1), an **similar to probabilistic Strips (with continuous distributions)**

Moldovan et al. ICRA 12, 13, 14;Auton. Robots 18





Biology

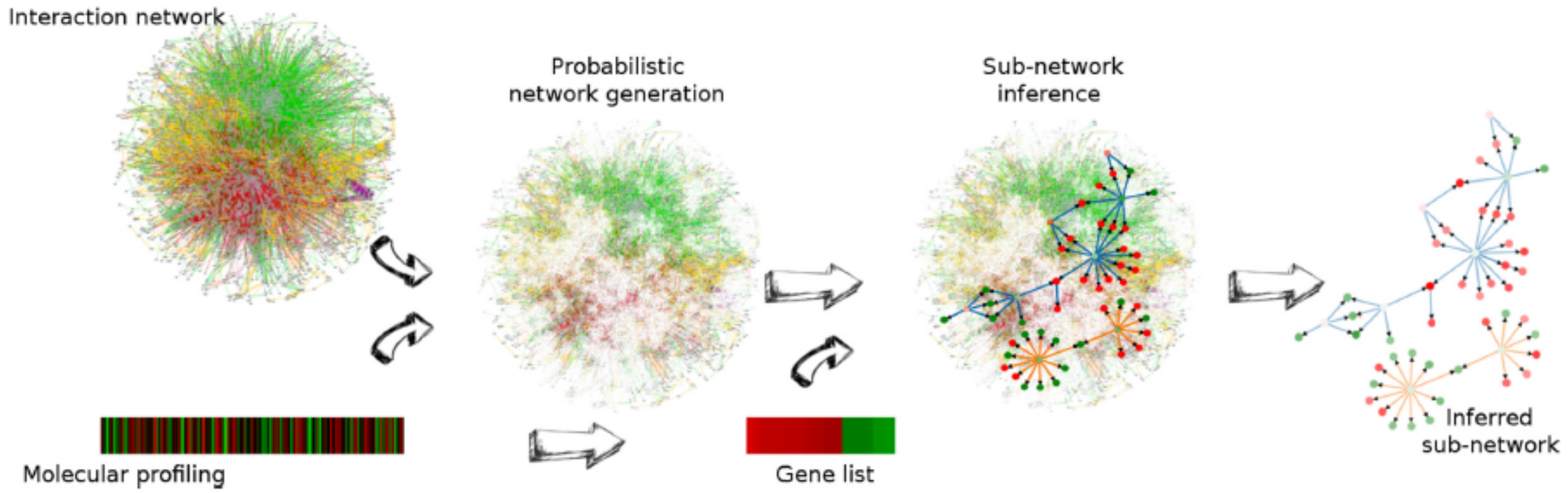


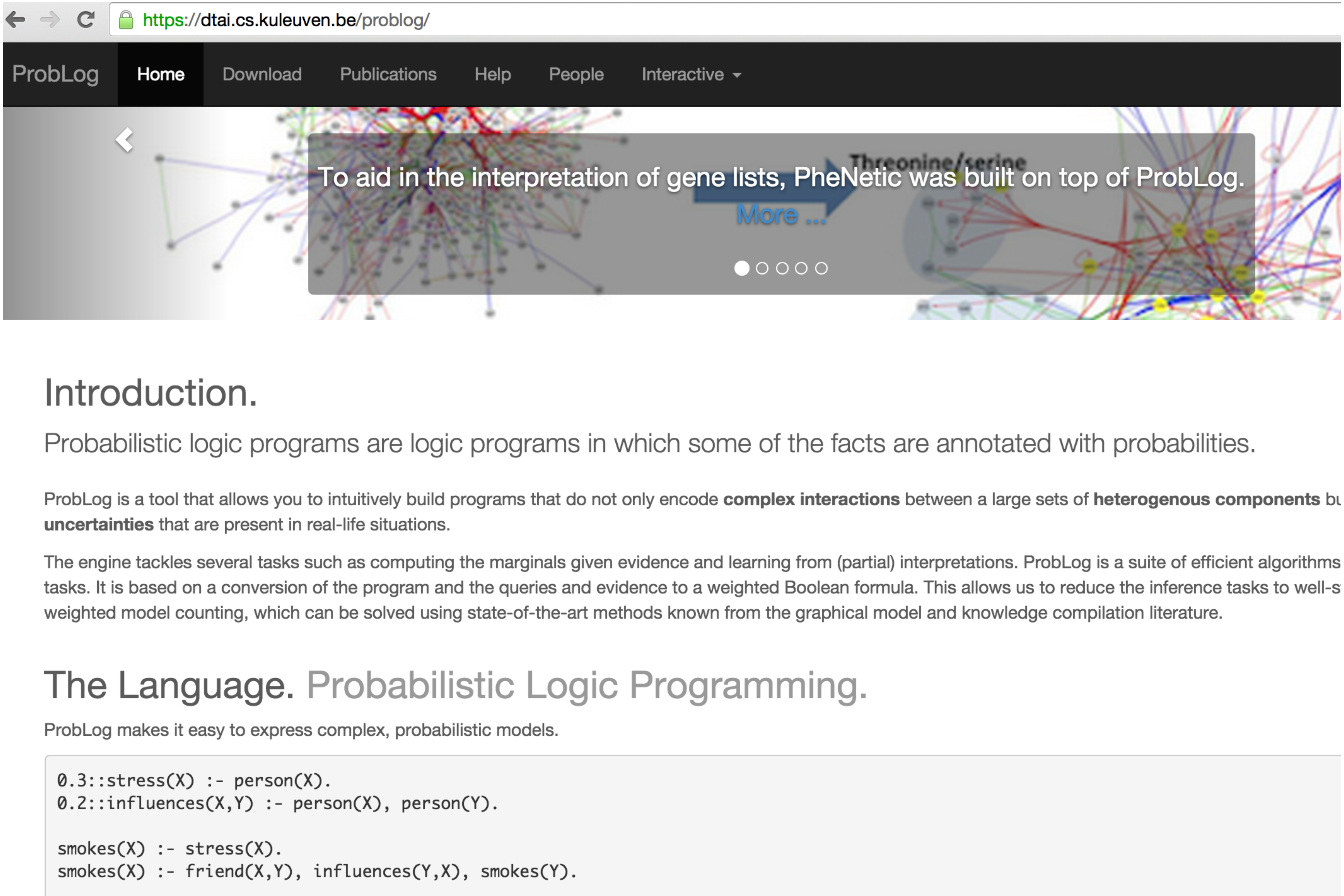
Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for a wide variety of organisms are readily available from the web server. Using the uploaded user-generated molecular data the interaction network is converted into a probabilistic network: edges receive a probability proportional to the levels measured for the terminal nodes in the molecular profiling data set. This probabilistic interaction network is used to infer the sub-network that best links the genes from the gene list. The inferred sub-network provides a trade-off between linking as many genes as possible from the gene list and selecting the least number of edges.

- Causes: Mutations
  - All related to similar phenotype
- Effects: Differentially expressed genes
- 27 000 cause effect pairs

- Interaction network:
  - 3063 nodes
    - Genes
    - Proteins
  - 16794 edges
    - Molecular interactions
    - Uncertain

- Goal: connect causes to effects through common subnetwork
  - = Find mechanism
- Techniques:
  - DTProbLog
  - Approximate inference

[De Maeyer et al., Molecular Biosystems 13, NAR 15] [Gross et al. Communications Biology, 19]



To aid in the interpretation of gene lists, PheNetic was built on top of ProbLog.

Introduction.

Probabilistic logic programs are logic programs in which some of the facts are annotated with probabilities.

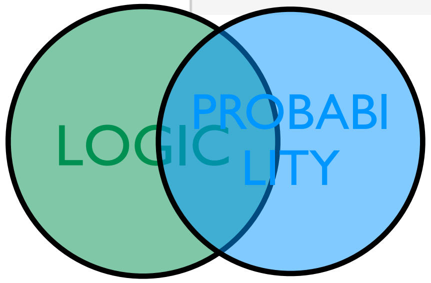
ProbLog is a tool that allows you to intuitively build programs that do not only encode **complex interactions** between a large sets of **heterogenous components** bu **uncertainties** that are present in real-life situations.

The engine tackles several tasks such as computing the marginals given evidence and learning from (partial) interpretations. ProbLog is a suite of efficient algorithms tasks. It is based on a conversion of the program and the queries and evidence to a weighted Boolean formula. This allows us to reduce the inference tasks to well-s weighted model counting, which can be solved using state-of-the-art methods known from the graphical model and knowledge compilation literature.

The Language. Probabilistic Logic Programming.

ProbLog makes it easy to express complex, probabilistic models.

```
0.3::stress(X) :- person(X).
0.2::influences(X,Y) :- person(X), person(Y).

smokes(X) :- stress(X).
smokes(X) :- friend(X,Y), influences(Y,X), smokes(Y).
```



Markov Logic: Intuition

- *Undirected graphical model*
- A logical KB is a set of **hard constraints** on the set of possible worlds
- Let's make them **soft constraints**:
  When a world violates a formula,
  it becomes less probable, not impossible
- Give each formula a **weight**
  (Higher weight $\Rightarrow$ Stronger constraint)

$$P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$

A possible worlds view

Say we have two domain elements **Anna** and **Bob** as well as two predicates **Friends** and **Happy**

	$\neg Happy(Bob)$	$Happy(Bob)$
$\neg Friends(Anna, Bob)$		
$Friends(Anna, Bob)$		

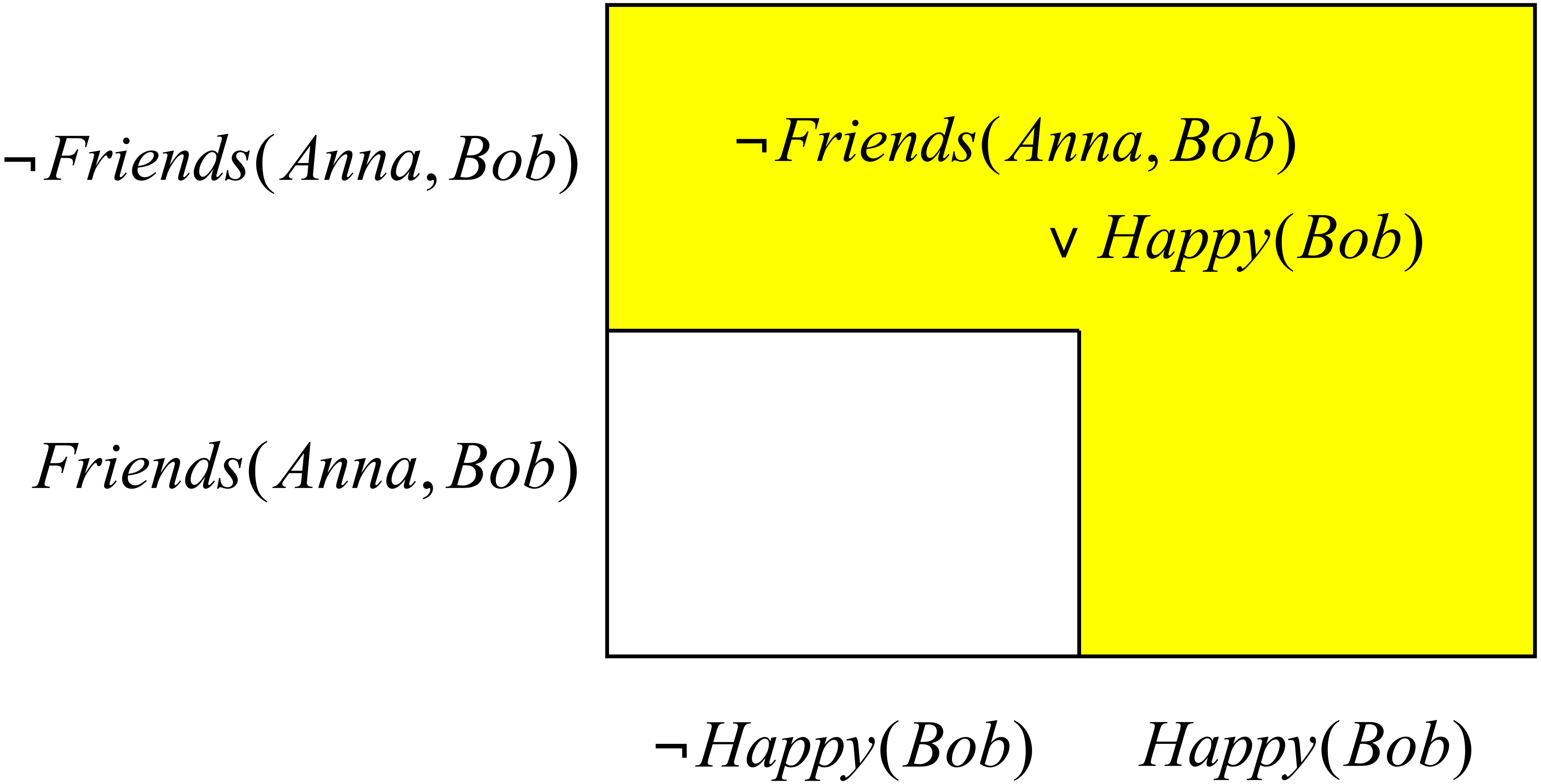
slides by Pedro Domingos

A possible worlds view

Logical formulas such as

not Friends(Anna,Bob) or Happy(Bob)

exclude possible worlds



$\neg Friends(Anna, Bob)$

$Friends(Anna, Bob)$

$\neg Friends(Anna, Bob) \lor Happy(Bob)$

$\neg Happy(Bob)$ $Happy(Bob)$

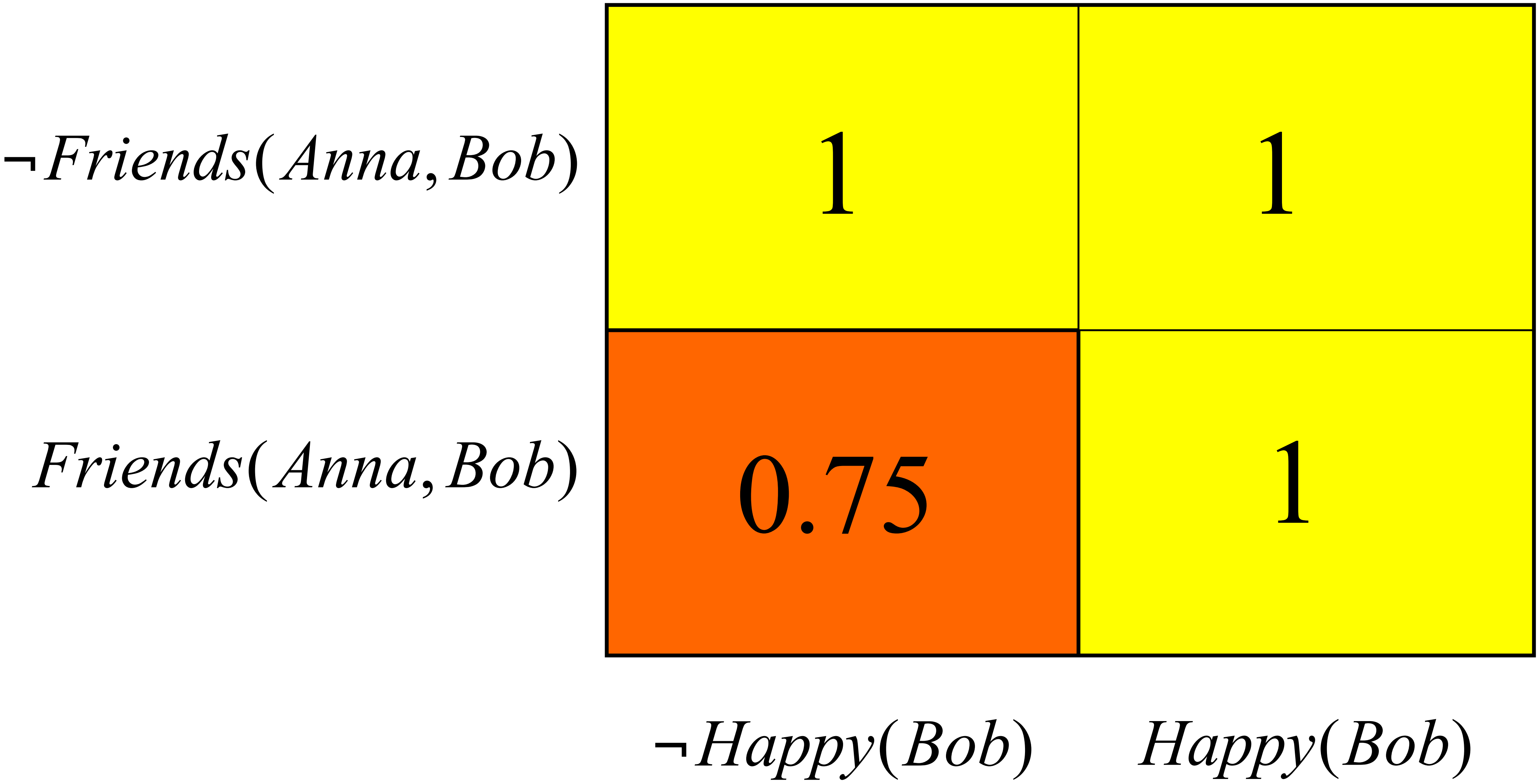
slides by Pedro Domingos

A possible worlds view

four times as likely that rule holds

$$\Phi(\neg Friends(Anna, Bob) \vee Happy(Bob)) = 1$$

$$\Phi(Friends(Anna, Bob) \wedge \neg Happy(Bob)) = 0.75$$



	$\neg Happy(Bob)$	$Happy(Bob)$
$\neg Friends(Anna, Bob)$	1	1
$Friends(Anna, Bob)$	0.75	1

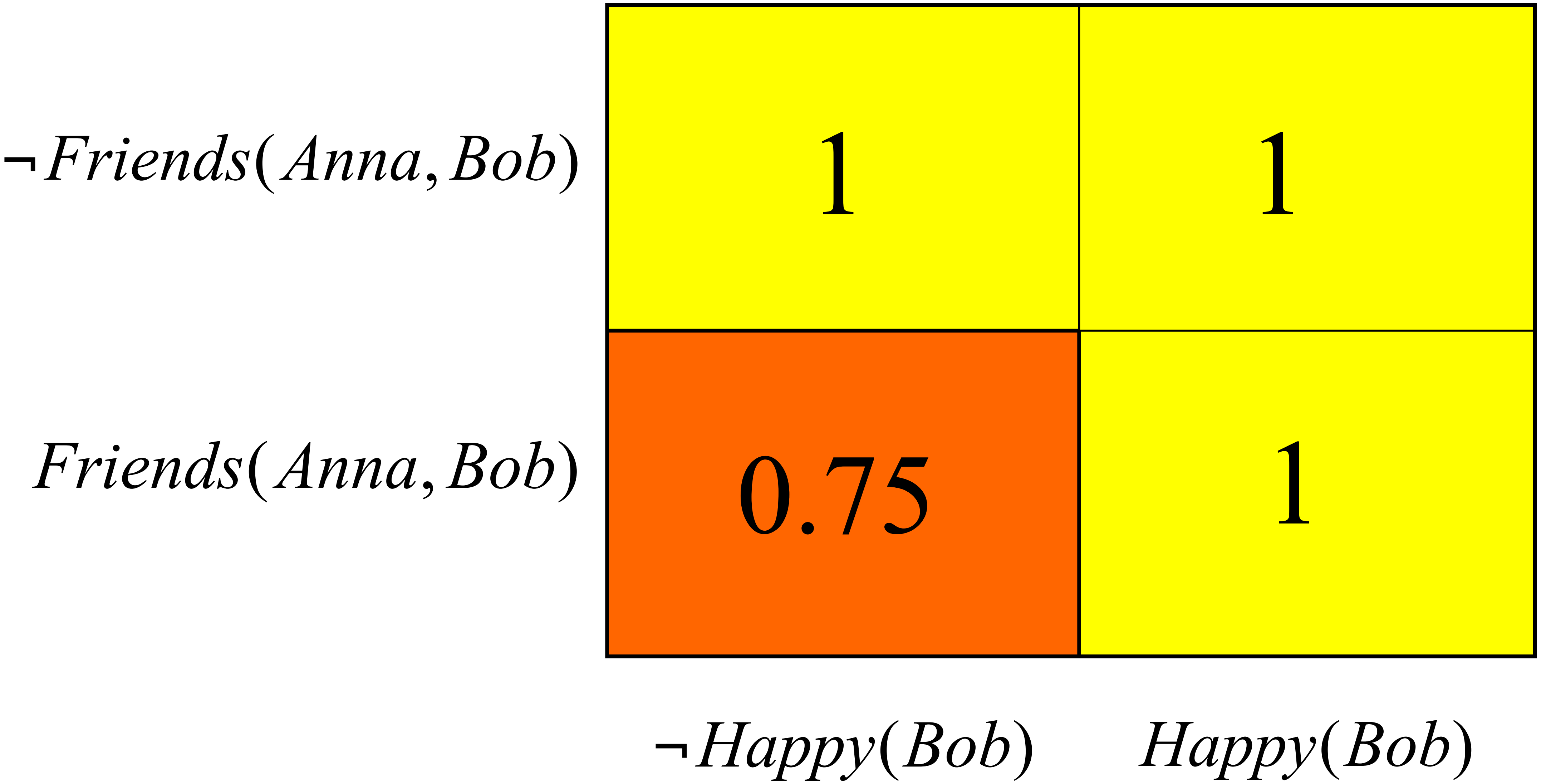
slides by Pedro Domingos

A possible worlds view

Or as log-linear model this is:

$$w(\Phi(\neg Friends(Anna, Bob) \vee Happy(Bob)))$$

$$= \log(1/0.75) = 0.29$$



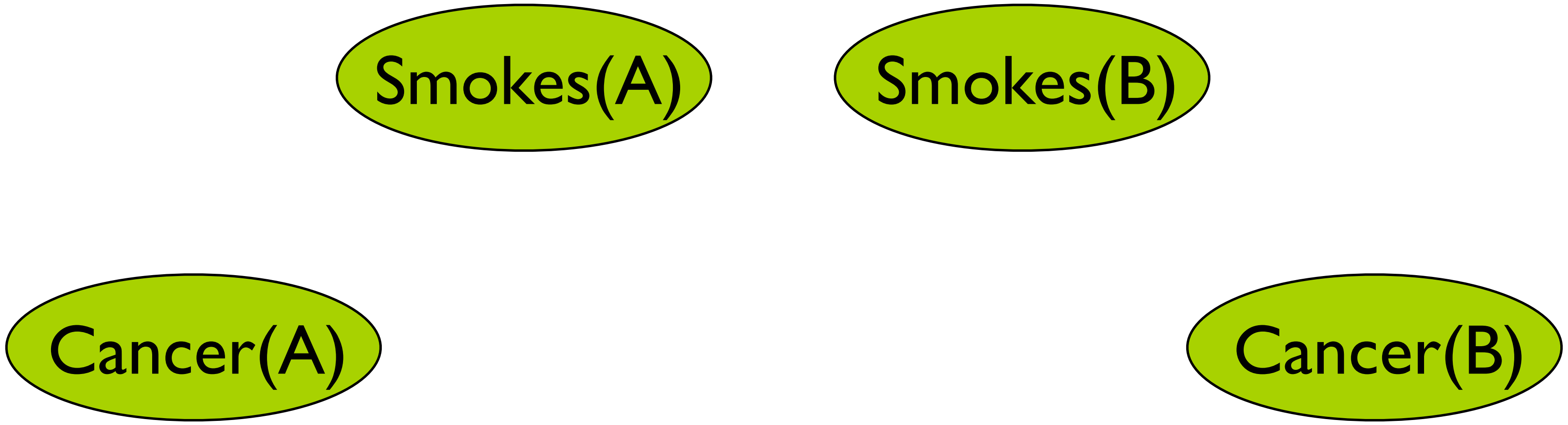
	$\neg Happy(Bob)$	$Happy(Bob)$
$\neg Friends(Anna, Bob)$	1	1
$Friends(Anna, Bob)$	0.75	1

This can also be viewed as building a graphical model

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)



slides by Pedro Domingos

Markov Logic

1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

Friends(A,B)

Friends(A,A)
Smokes(A)
Smokes(B)
Friends(B,B)

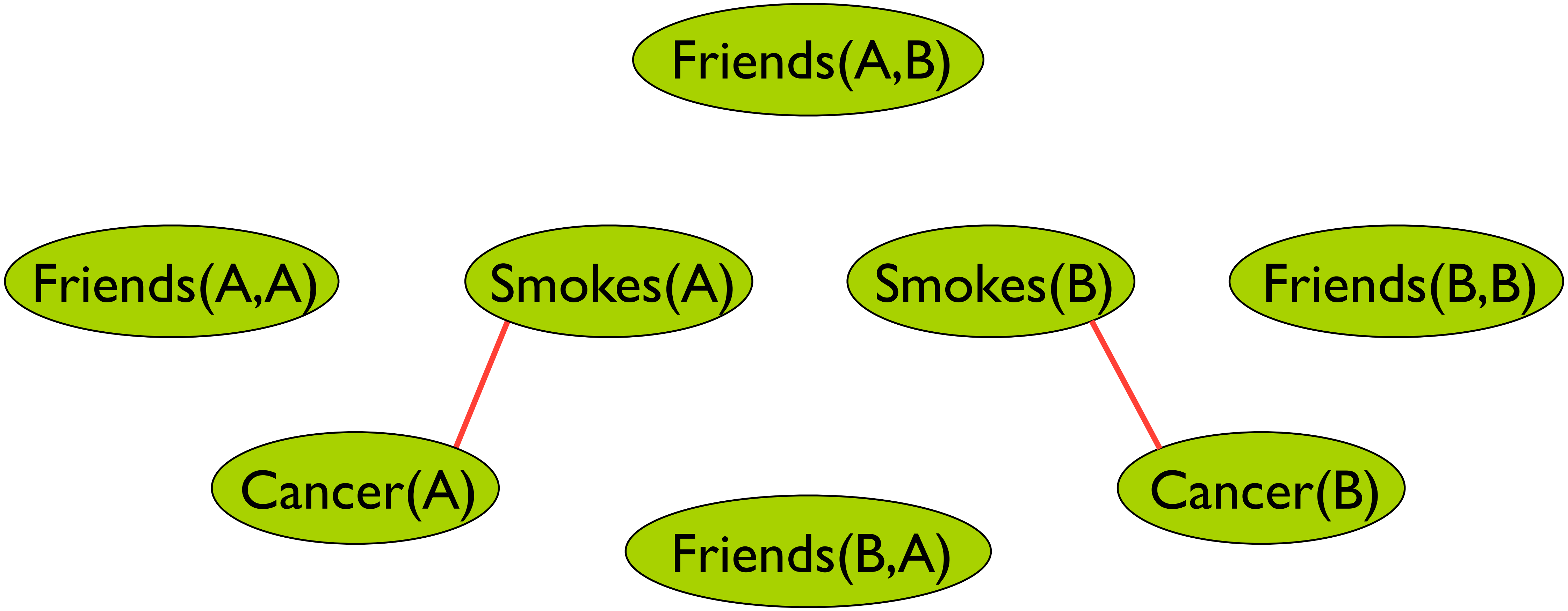
Cancer(A)
Friends(B,A)
Cancer(B)

slides by Pedro Domingos

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x,y\ Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

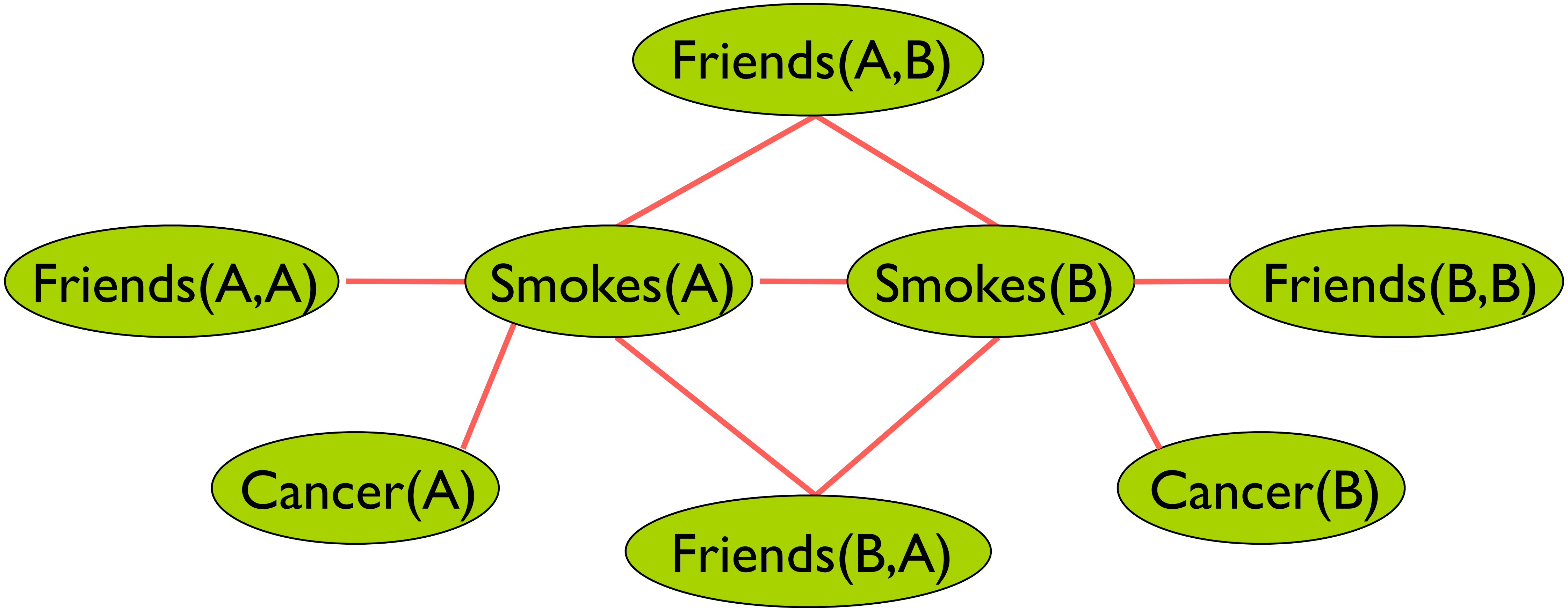


slides by Pedro Domingos

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x,y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

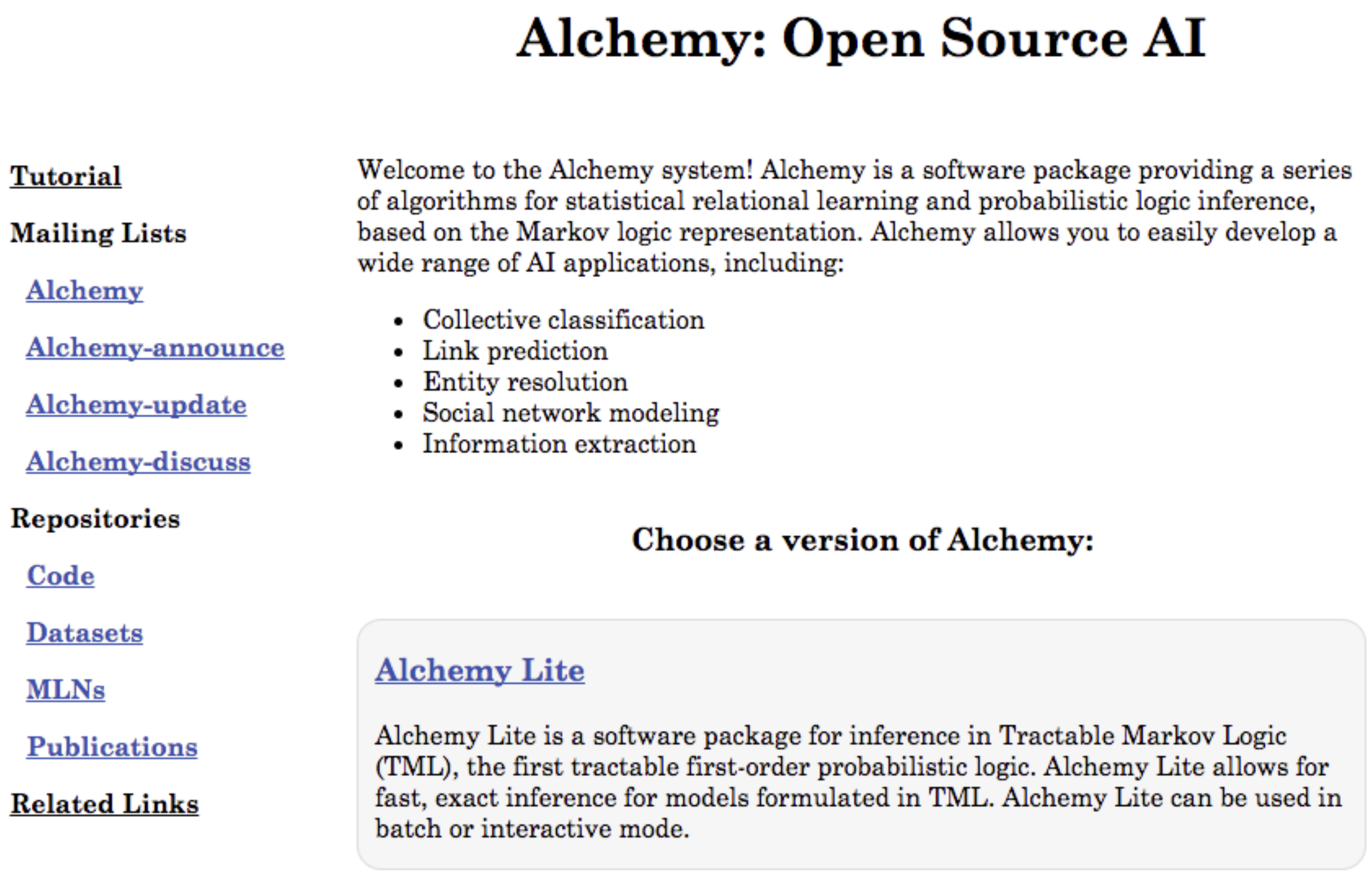
Suppose we have two constants: **Anna** (A) and **Bob** (B)



slides by Pedro Domingos

Applications

- Natural language processing, Collective Classification, Social Networks, Activity Recognition, …



Alchemy: Open Source AI

Tutorial

Mailing Lists

- Alchemy
- Alchemy-announce
- Alchemy-update
- Alchemy-discuss

Repositories

- Code
- Datasets
- MLNs
- Publications

Related Links

Welcome to the Alchemy system! Alchemy is a software package providing a series of algorithms for statistical relational learning and probabilistic logic inference, based on the Markov logic representation. Alchemy allows you to easily develop a wide range of AI applications, including:

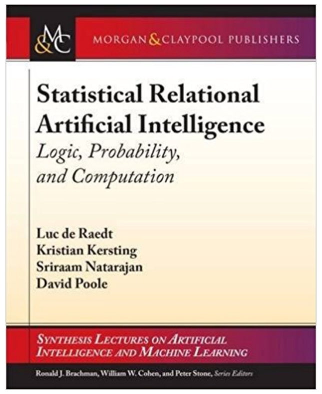
- Collective classification
- Link prediction
- Entity resolution
- Social network modeling
- Information extraction

Choose a version of Alchemy:

Alchemy Lite

Alchemy Lite is a software package for inference in Tractable Markov Logic (TML), the first tractable first-order probabilistic logic. Alchemy Lite allows for fast, exact inference for models formulated in TML. Alchemy Lite can be used in batch or interactive mode.

2. Directed vs Undirected the PGM / StarAI dimension



MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational Artificial Intelligence
Logic, Probability, and Computation

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning

0.1 :: burglary.

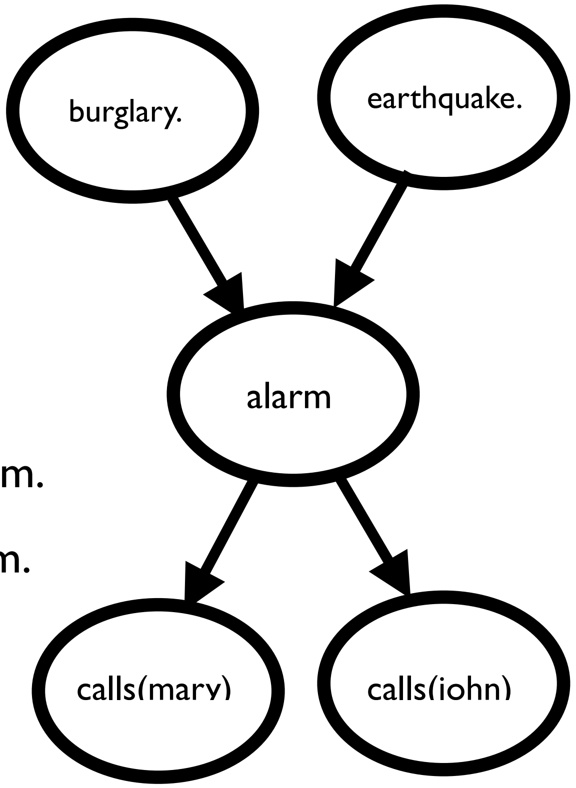
0.05 :: earthquake.

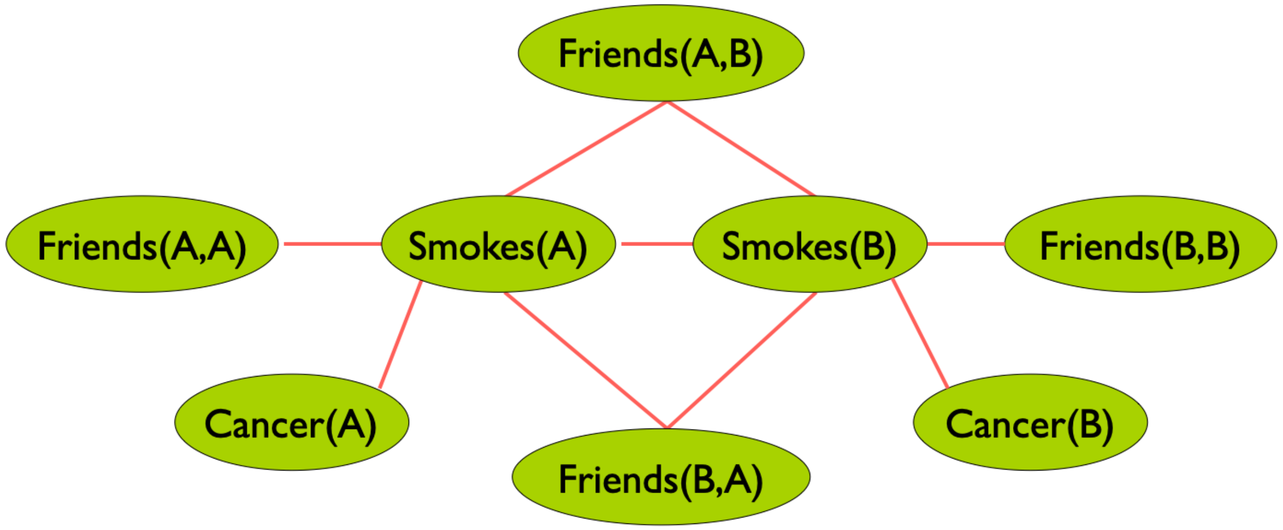
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.





$$1.5 \quad \forall x \; Smokes(x) \Rightarrow Cancer(x)$$

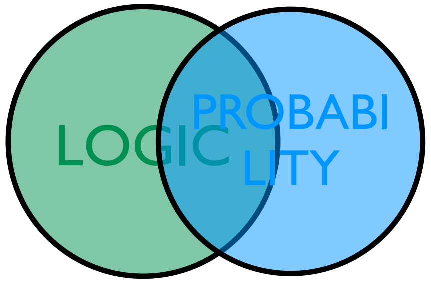
$$1.1 \quad \forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

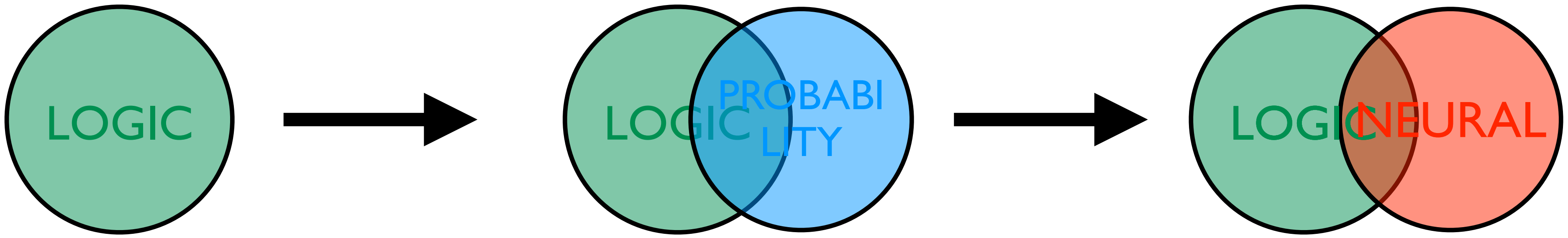
Markov Logic

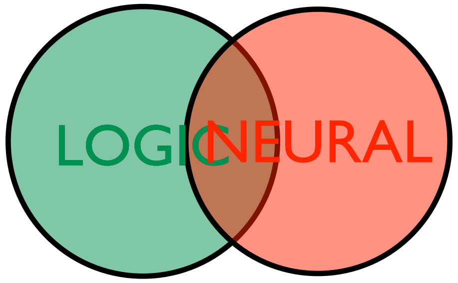
undirected
Markov Net
model theoretic



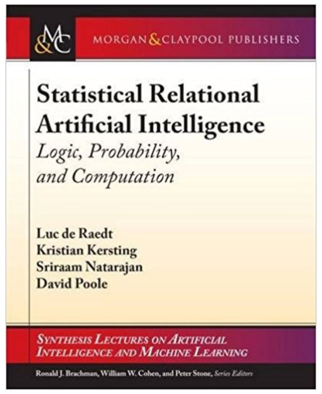
key representatives

1. Proof vs Model based
2. Directed vs Undirected





2. Directed vs Undirected the NeSy dimension


MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational Artificial Intelligence
Logic, Probability, and Computation

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Ronald J. Brachman, William W. Cohen, and Peter Stone, Series Editors

Two types of Neural Symbolic Systems

Just like in StarAI

Logic as a kind of ***neural program***

directed StarAI approach and logic programs

Logic as the ***regularizer***
(reminiscent of Markov Logic Networks)

undirected StarAI approach and (soft) constraints

Also, many NeSy systems are doing
knowledge based model construction KBMC
where logic is used as a template

Just like in StarAI

Logic as a neural program

directed StarAI approach and logic programs

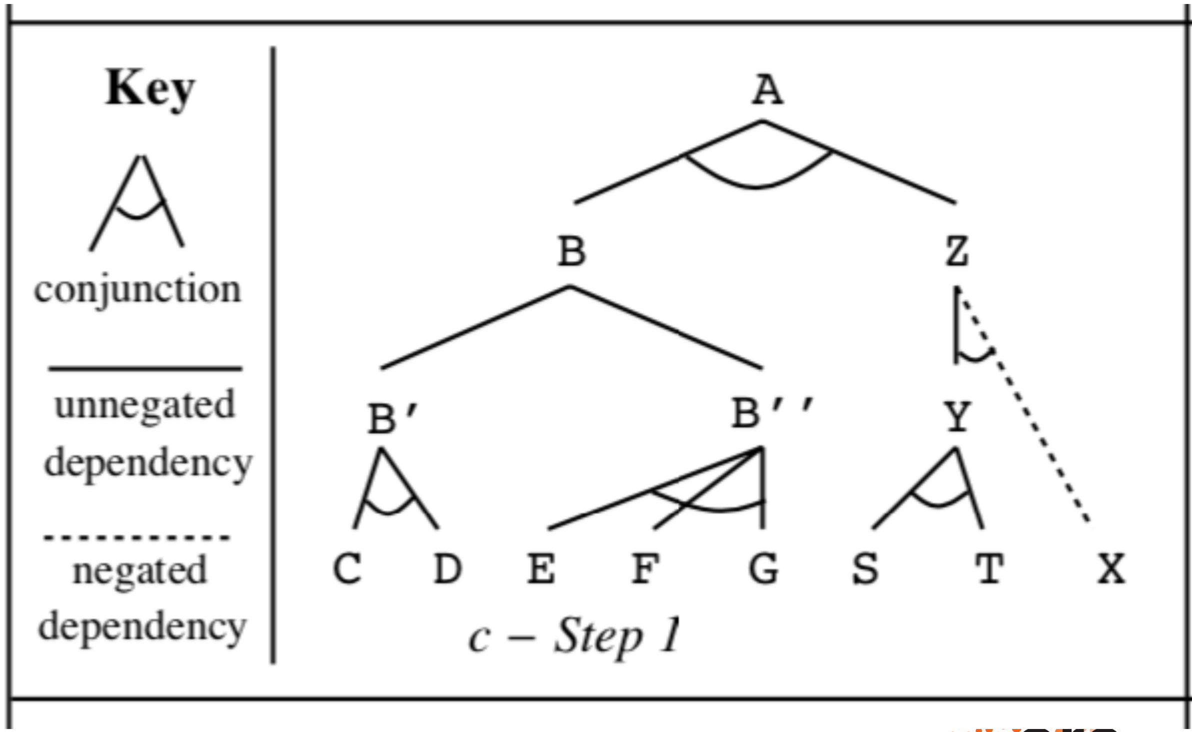
- KBANN (Towell and Shavlik AIJ 94)
- Turn a (propositional) Prolog program into a neural network and learn

```
A :- B, Z.
B :- C, D.
B :- E, F, G.
Z :- Y, not X.
Y :- S, T.
```

REWRITE →

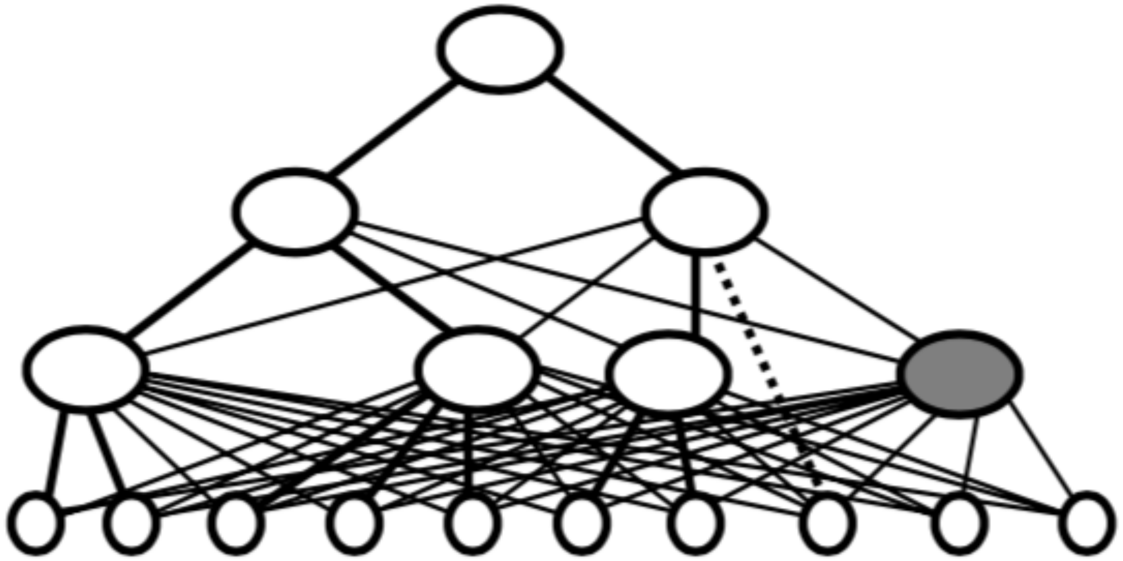
```
A    :- B, Z.
B    :- B′.
B    :- B′′.
B′   :- C, D.
B′′  :- E, F, G.
Z    :- Y, not X.
Y    :- S, T.
```

→

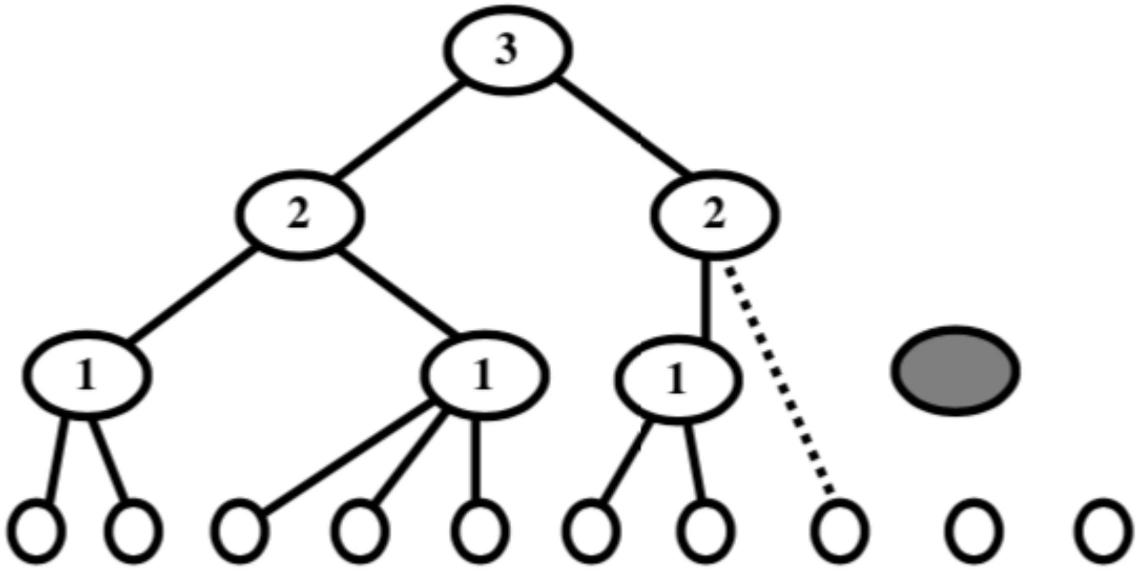


Logic as a neural program

directed StarAI approach and logic programs



f – Steps 4–6



e – Step 3

ADD LINKS — ALSO SPURIOUS ONES

HIDDEN UNIT

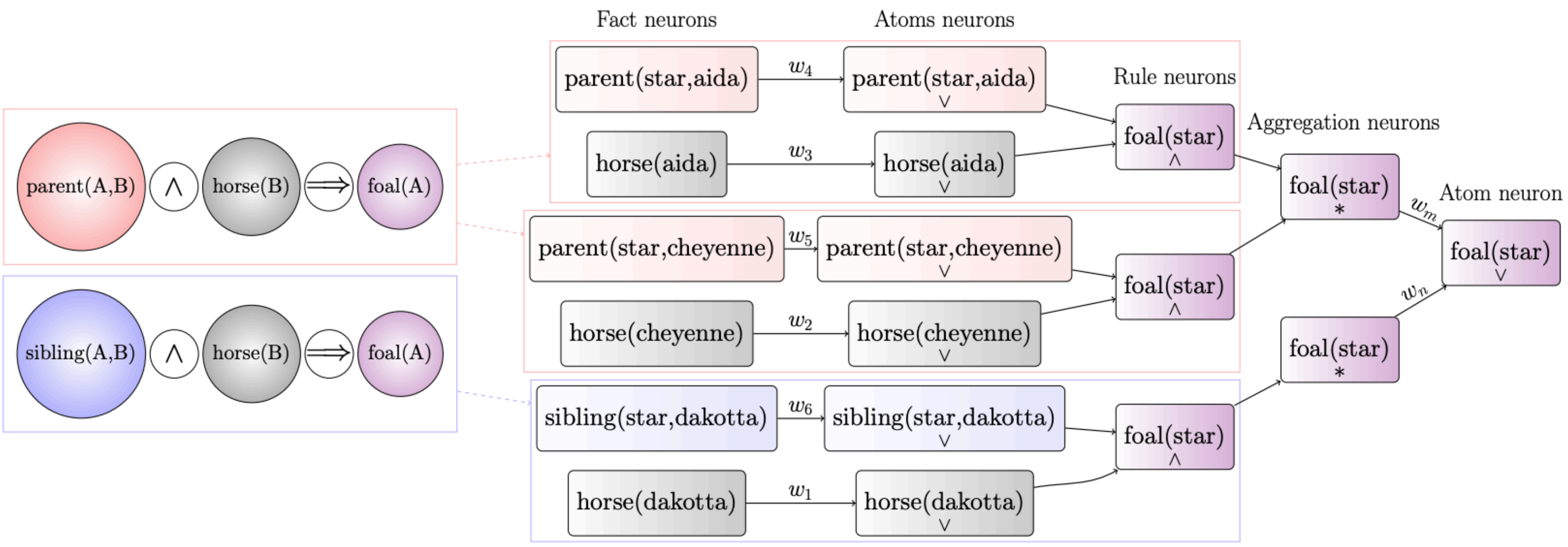
and then learn

(Details of activation & loss functions not mentioned)

Lifted Relational Neural Networks

directed StarAI approach and logic programs

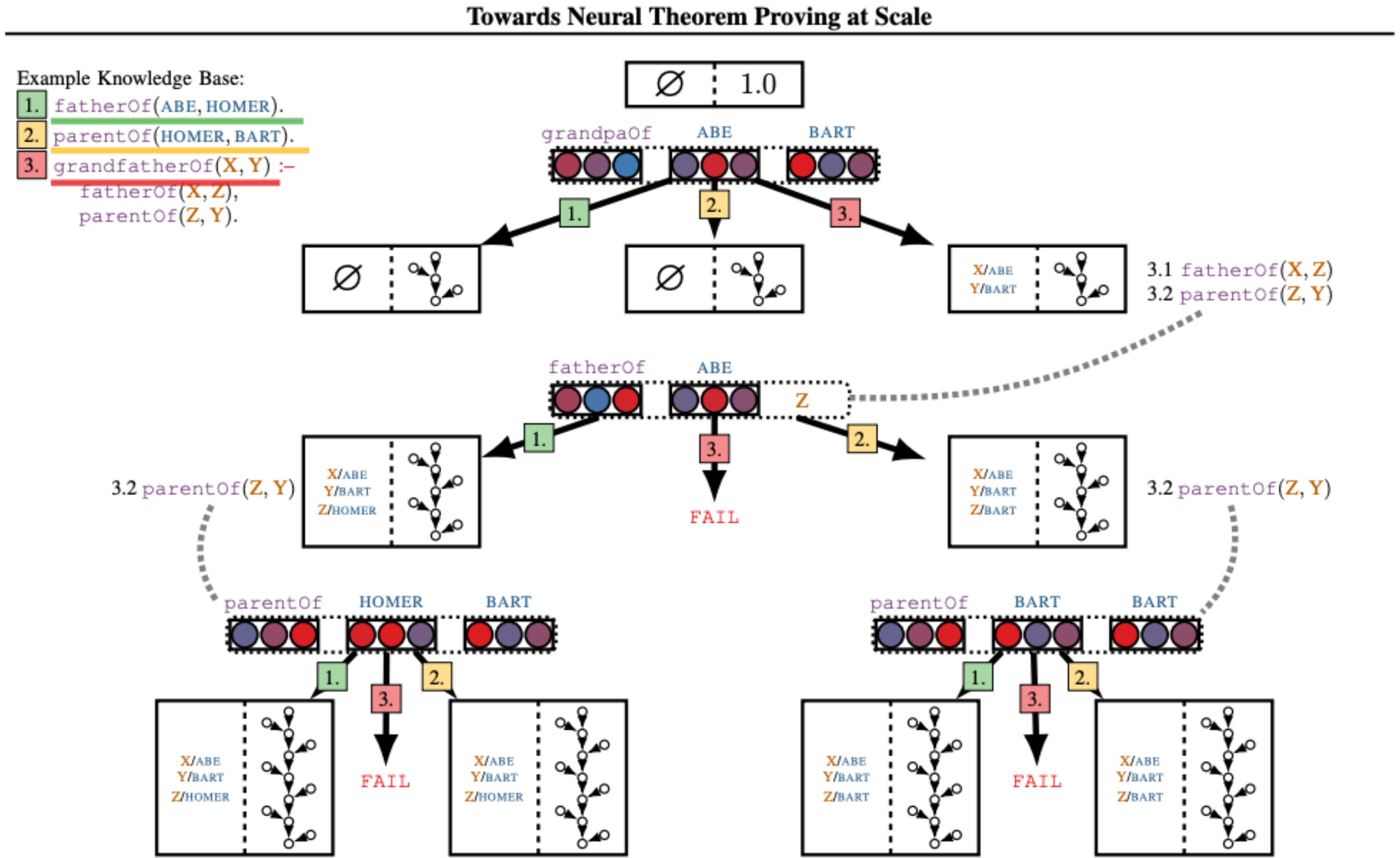
- Directed (fuzzy) NeSy
- similar in spirit to the Bayesian Logic Programs and Probabilistic Relational Models
- Of course, other kind of (fuzzy) operations for AND, OR and Aggregation (cf. later)



[Sourek, Kuzelka, et al JAIR]

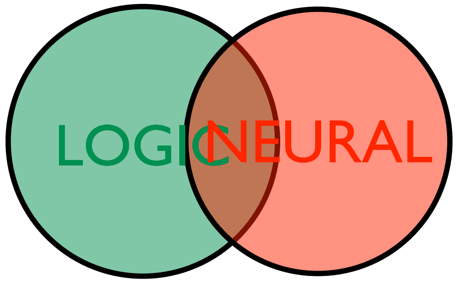
Neural Theorem Prover

directed StarAI approach and logic programs

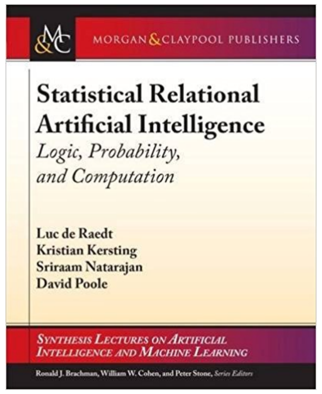


the logic is encoded in the network
how to reason logically ?

[Rocktäschel Riedel, NeurIPS 17; Minervini et al.]



2. Directed vs Undirected the NeSy dimension



MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational
Artificial Intelligence
*Logic, Probability,
and Computation*

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning

Ronald J. Brachman, William W. Cohen, and Peter Stone, *Series Editors*

Two types of Neural Symbolic Systems

Just like in StarAI

Logic as a kind of ***neural program***

directed StarAI approach and logic programs

Logic as the ***regularizer***
(reminiscent of Markov Logic Networks)

undirected StarAI approach and (soft) constraints

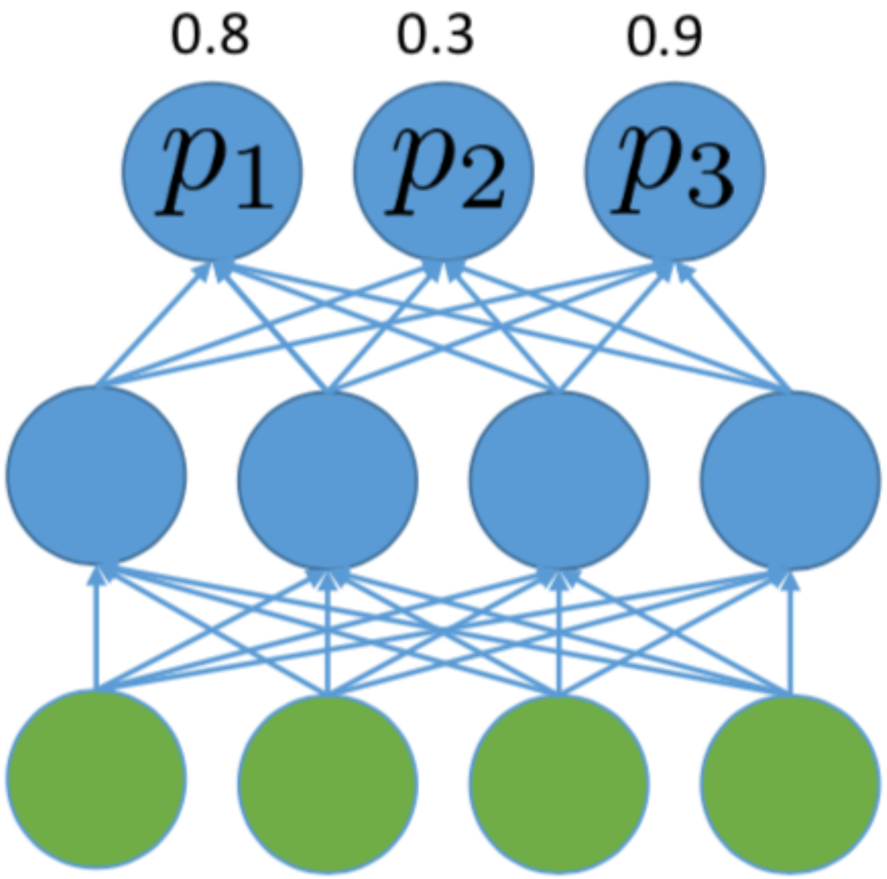
Also, many NeSy systems are doing
knowledge based model construction KBMC
where logic is used as a template

Just like in StarAI

Logic as constraints

undirected StarAI approach and (soft) constraints

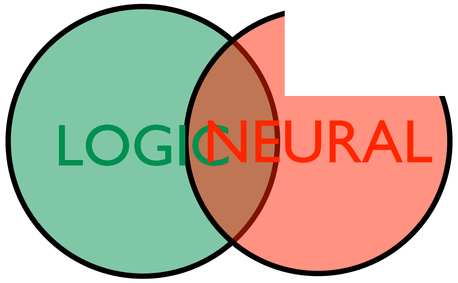
multi-class classification



This constraint should be satisfied

$$(\neg x_1 \wedge \neg x_2 \wedge x_3) \vee$$
$$(\neg x_1 \wedge x_2 \wedge \neg x_3) \vee$$
$$(x_1 \wedge \neg x_2 \wedge \neg x_3)$$

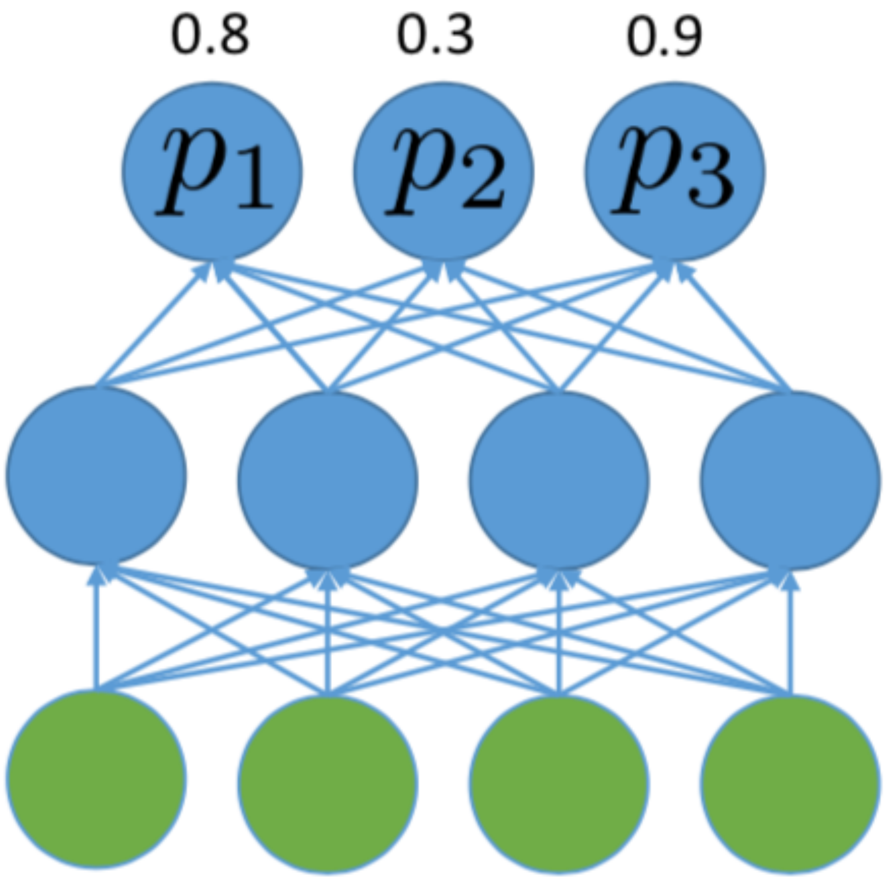
from Xu et al., ICML 2018



Logic as constraints

undirected StarAI approach and (soft) constraints

multi-class classification

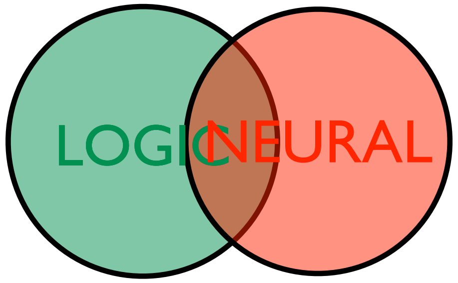


Probability that constraint is satisfied

$$(1-x_1)(1-x_2)x_3+$$
$$(1-x_1)x_2(1-x_3)+$$
$$x_1(1-x_2)(1-x_3)$$

basis for SEMANTIC LOSS

(weighted model counting)

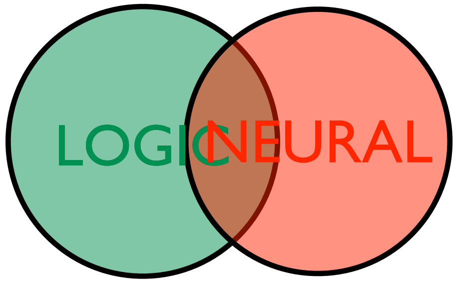


Logic as a regularizer

undirected StarAI approach and (soft) constraints

Semantic Loss:

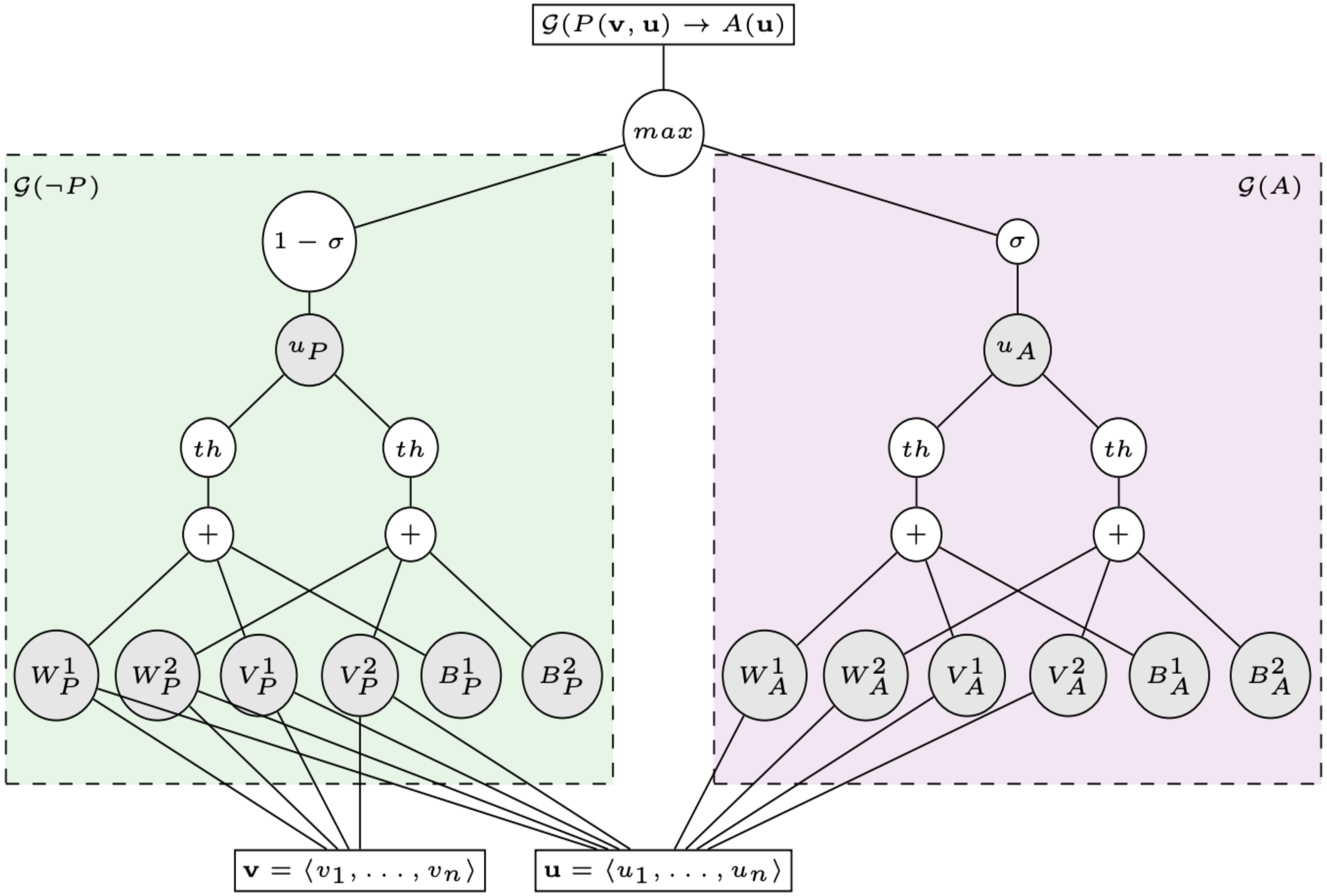
- Use logic as constraints (very much like “propositional MLNs)
- Semantic loss $SLoss(T) \propto -\log \sum_{X \models T} \prod_{x \in X} p_i \prod_{\neg x \in X} (1 - p_i)$
- Used as regulariser $Loss = TraditionalLoss + w.SLoss$
- Use weighted model counting , close to StarAI



Logic Tensor Networks

undirected StarAI approach and (soft) constraints

$$P(x,y) \rightarrow A(y), \text{with } \mathcal{G}(x) = \mathbf{v} \text{ and } \mathcal{G}(y) = \mathbf{u}$$



Serafini & Garcez

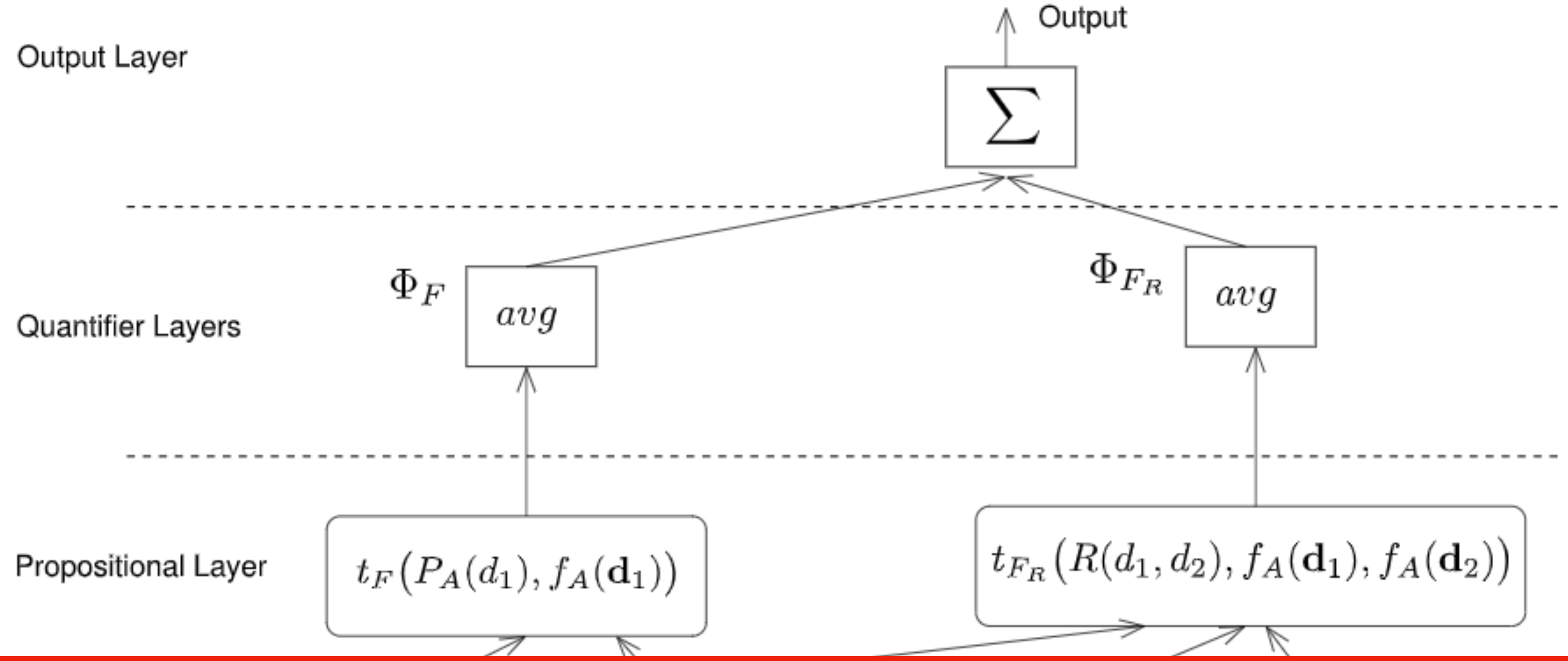
Semantic Based Regularization

undirected StarAI approach and (soft) constraints

$$\begin{aligned} F &:= \forall d\ P_A(d) \Rightarrow A(d) \\ F_R &:= \forall d\ \forall d'\ R(d,d') \Rightarrow ((A(d) \wedge A(d')) \vee (\neg A(d) \wedge \neg A(d'))) \\ C &= \{d_1, d_2\} \end{aligned}$$

Evidence Predicate Groundings

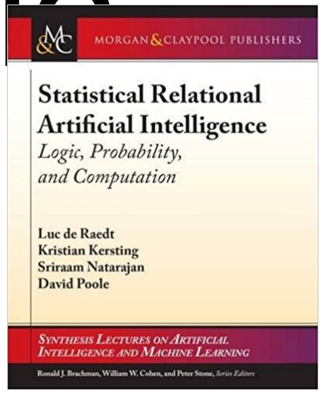
$P_A(d_1) = 1$
$R(d_1, d_2) = 1$



the logic is encoded in the network
how to reason logically ?

Diligenti et al. AIJ

Two types of Neural Symbolic Systems


MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational Artificial Intelligence
Logic, Probability, and Computation

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning

Ronald J. Brachman, William W. Cohen, and Peter Stone, *Series Editors*

Just like in StarAI

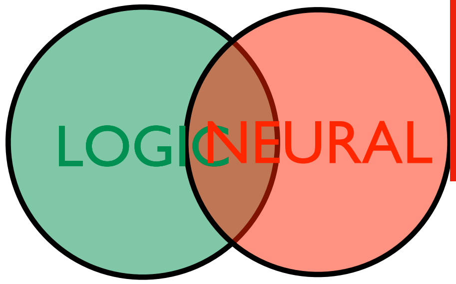
Logic as a kind of ***neural program***

directed StarAI approach and logic programs

Logic as the ***regularizer***
(reminiscent of Markov Logic Networks)

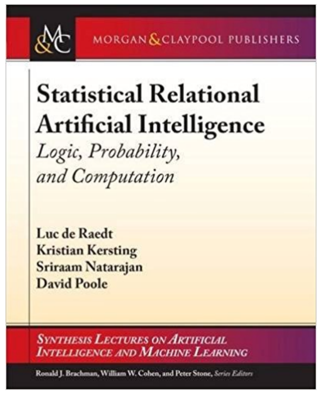
undirected StarAI approach and (soft) constraints

Consequence :
the logic is encoded in the network
the ability to logically reason is lost
logic is not a special case



2. Directed vs Undirected
the NeSy dimension

Two types of Neural Symbolic Systems


MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational Artificial Intelligence
Logic, Probability, and Computation

Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning

Ronald J. Brachman, William W. Cohen, and Peter Stone, *Series Editors*

Just like in StarAI

Logic as a kind of ***neural program***

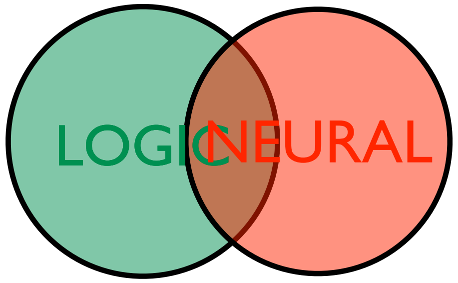
directed StarAI approach and logic programs

Logic as the ***regularizer***
(reminiscent of Markov Logic Networks)

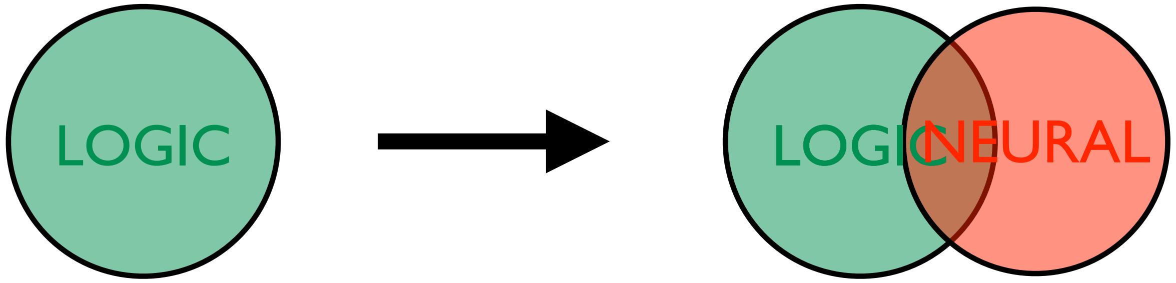
undirected StarAI approach and (soft) constraints

Also, many NeSy systems are doing
knowledge based model construction KBMC
where logic is used as a template

Just like in StarAI



3. Types of Logic



3. Types of Logic

Key Messages

- Different types of logic exist
- Different types of logic enable different functionalities

3. Types of Logic

LOGIC

Various flavours of logic

```
alarm :– earthquake.
alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.
calls_john :– alarm, hears_alarm_john.
```

Propositional logic

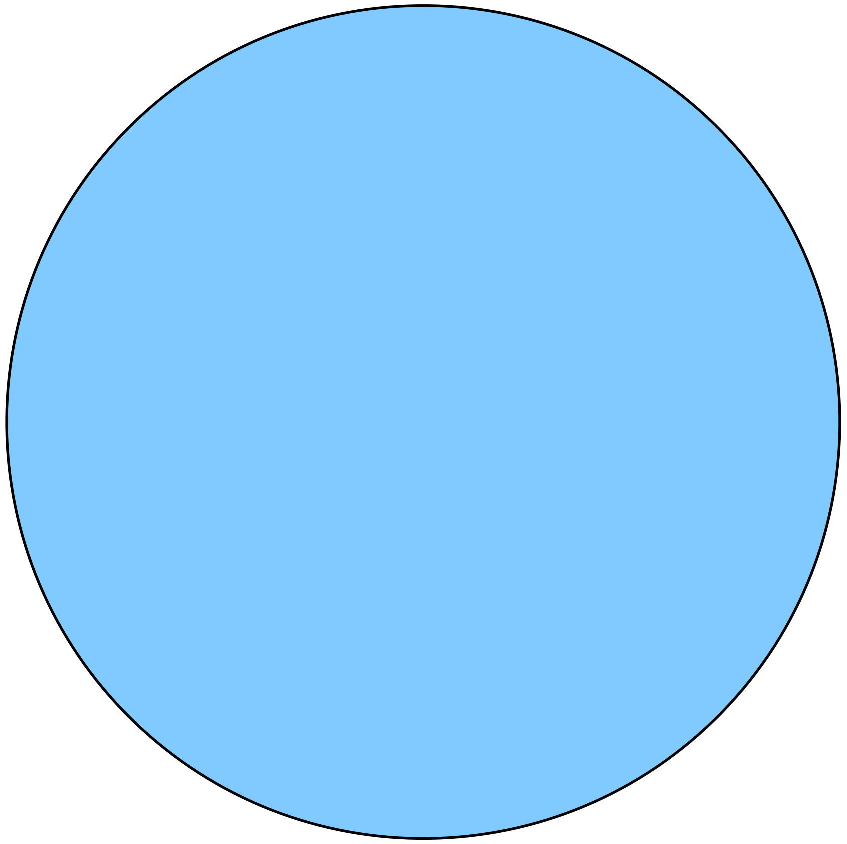
```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
    influences(Y,X),
    smokes(Y).
```

First-order logic

LOGIC

Various flavours of first-order logic

Logic programs
= programming language



LOGIC

Logic programming and Prolog

Full-fledged programming language

structured terms

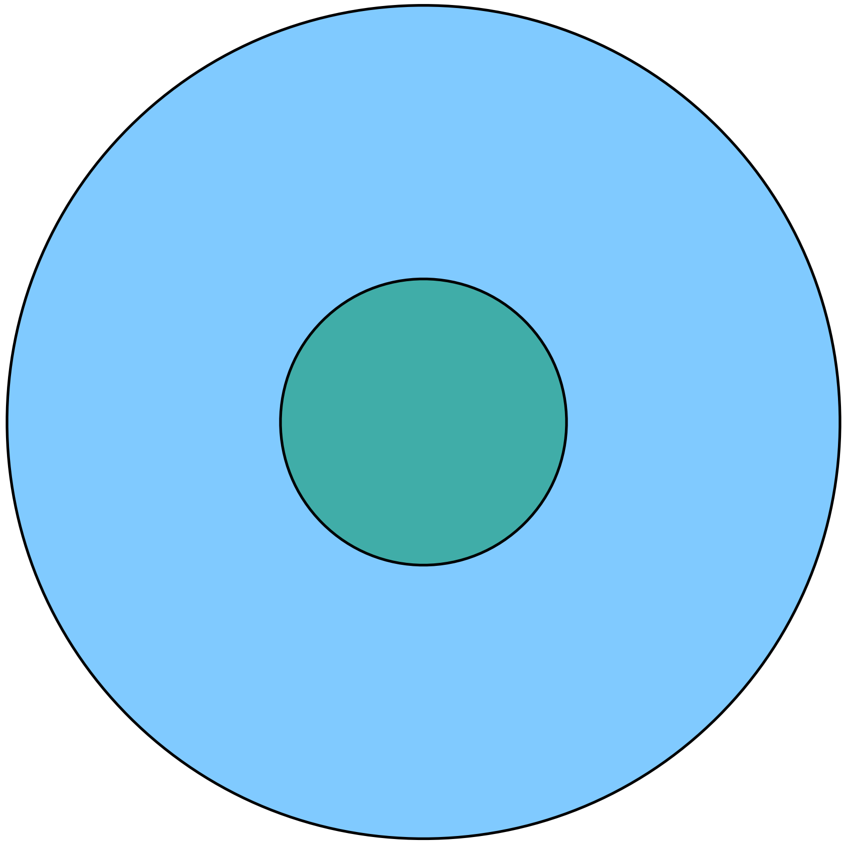
```
member(X, [X|_]).

member(X, [_|Tail]) :-
   member(X, Tail).
```

recursion

LOGIC

Various flavours of first-order logic

Logic programs
= programming language



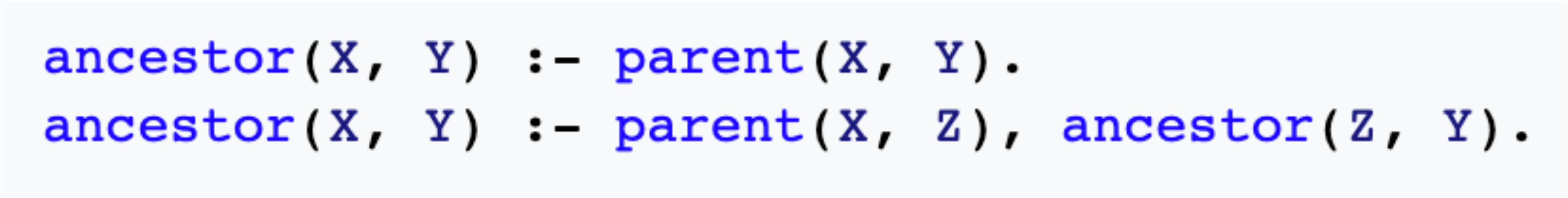
Datalog
= Logic programs
that always terminate

Datalog

Query language for deductive databases

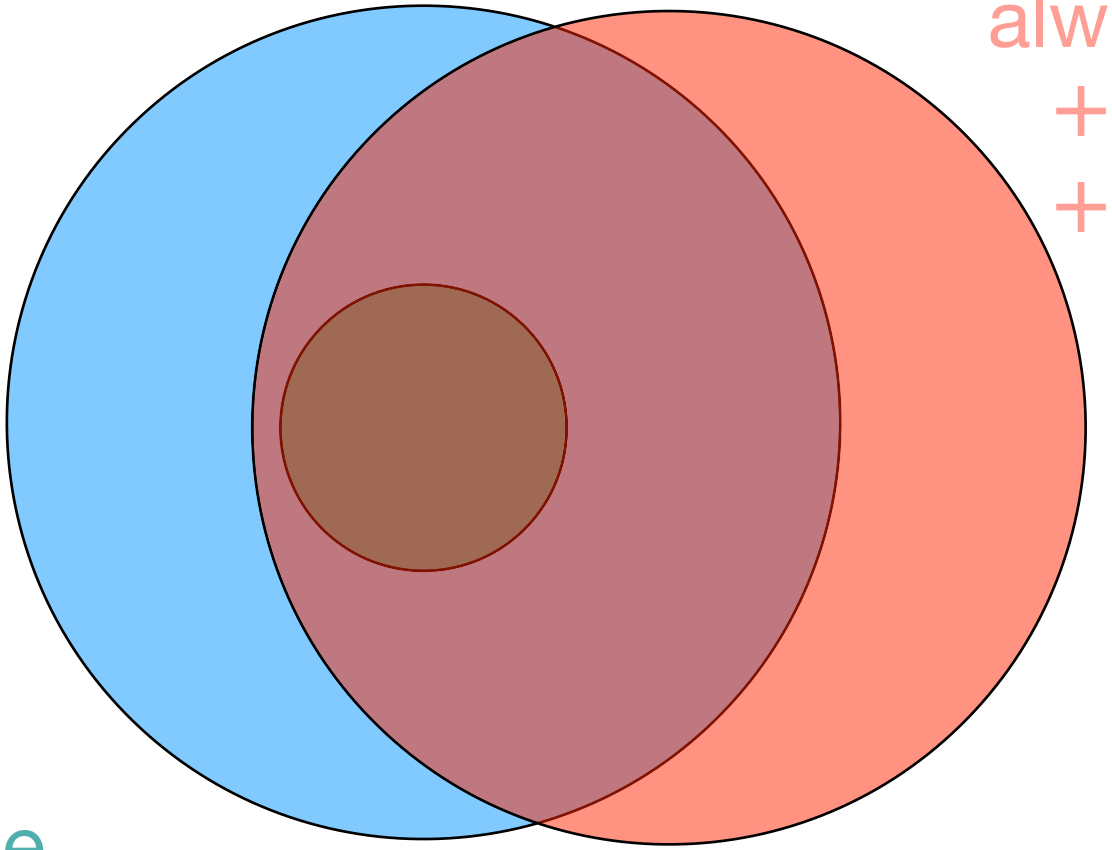
no structured terms

guaranteed to terminate



```
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

Various flavours of first-order logic



Logic programs
= programming language

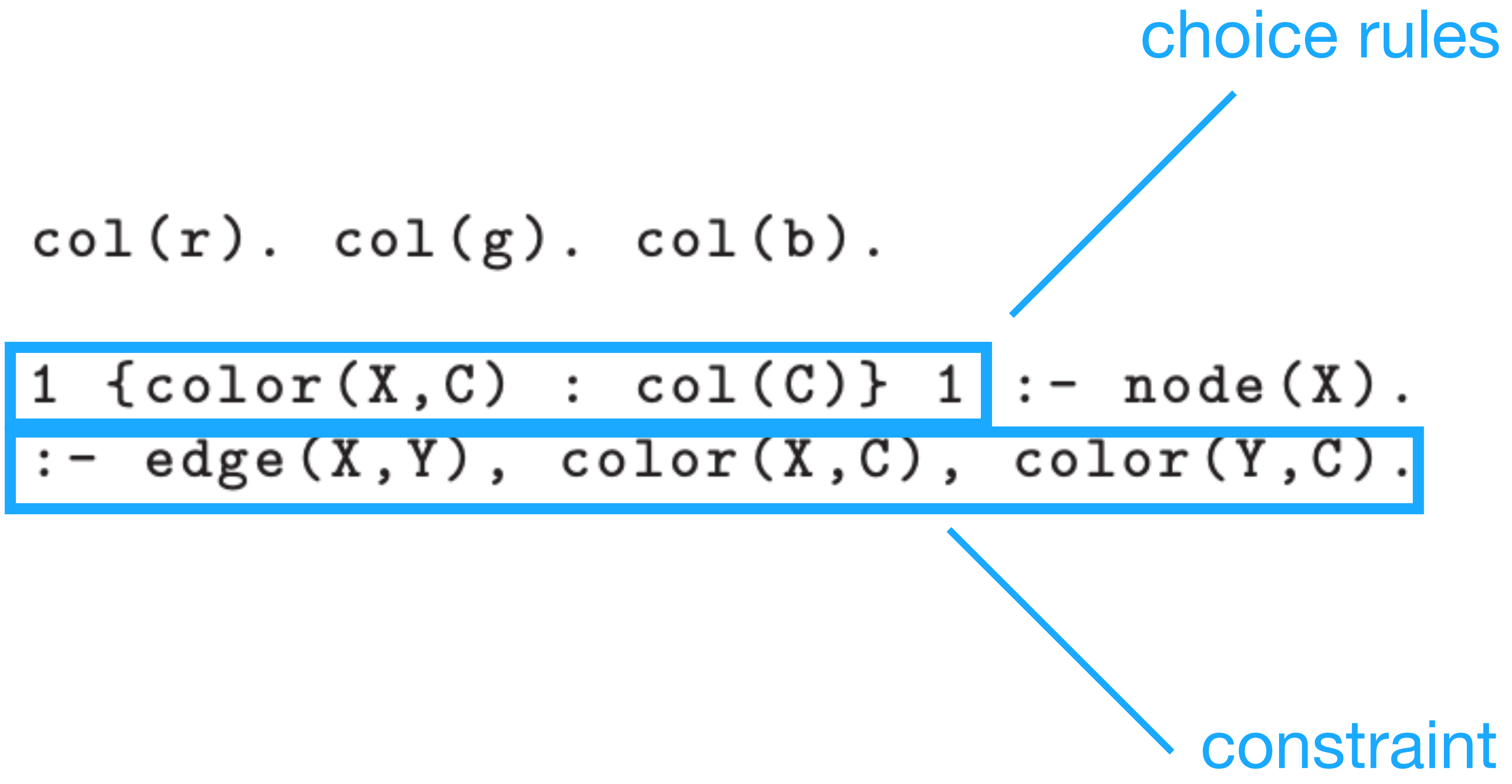
Answer-set programs
= Logic programs with
multiple models that
always terminate
+ soft/hard constraints
+ preferences

Datalog
= Logic programs
that always terminate

LOGIC

Answer-set programming

Prolog with multiple models + interesting features



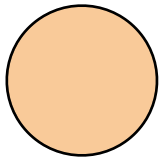
choice rules

```
col(r). col(g). col(b).

1 {color(X,C) : col(C)} 1 :- node(X).
:- edge(X,Y), color(X,C), color(Y,C).
```

constraint

LOGIC

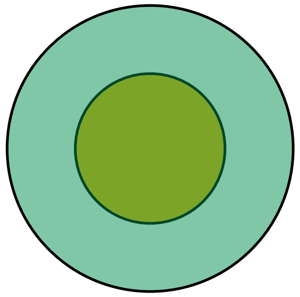
What can it do?



Propositional logic:
simple propositional reasoning

LOGIC

What can it do?

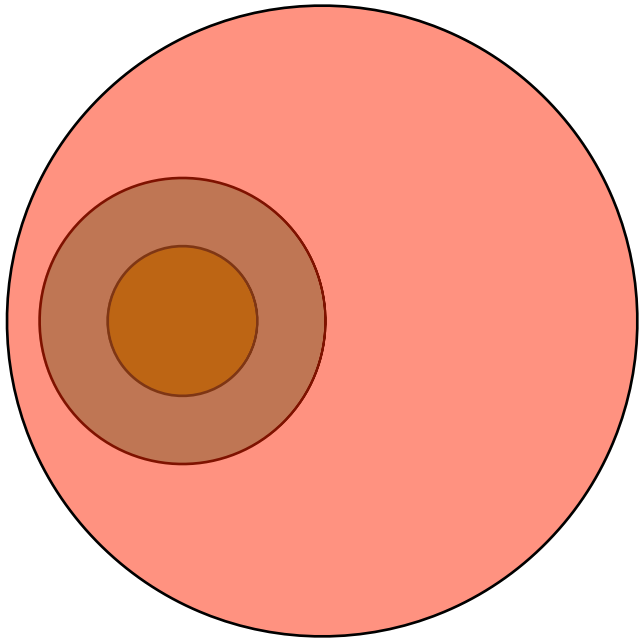


Datalog:
database queries

Propositional logic:
simple propositional reasoning

LOGIC

What can it do?



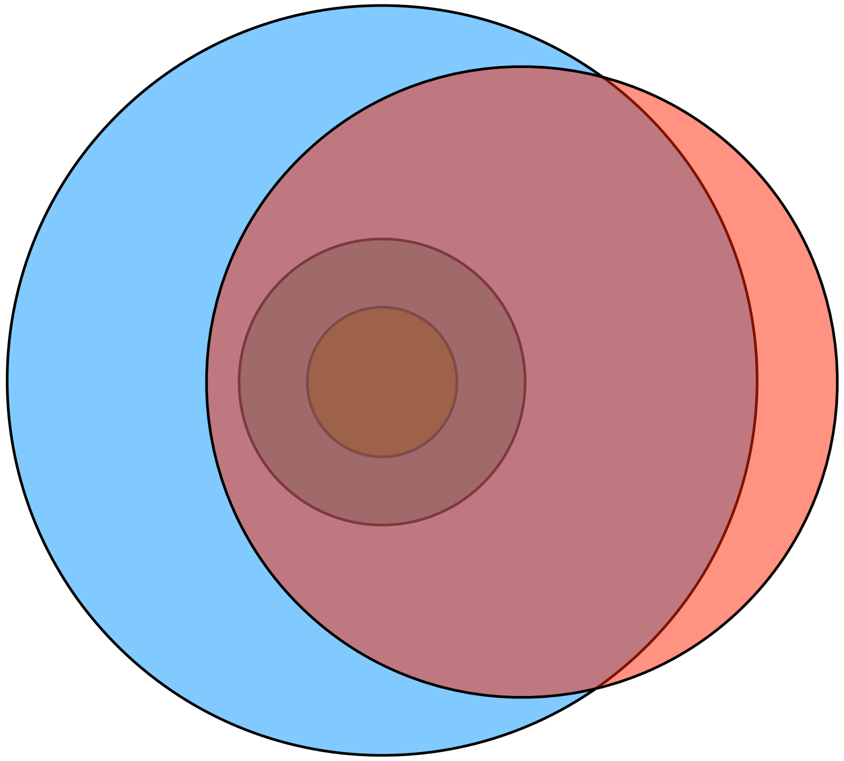
Answer-set programming:
database queries, common-sense reasoning, preferences

Datalog:
database queries

Propositional logic:
simple propositional reasoning

LOGIC

What can it do?



Logic programming:
programs manipulating structured objects, infinite domains, …

Answer-set programming:
database queries, common-sense reasoning, preferences

Datalog:
database queries

Propositional logic:
simple propositional reasoning

LOGIC

Logic program vs First-order logic

Issues with transitive closure in first-order logic

edge(1,2).
path(A,B) ← edge(A,B).
path(A,B) ← edge(A,C), path(C,B).

Logic programs always have one model

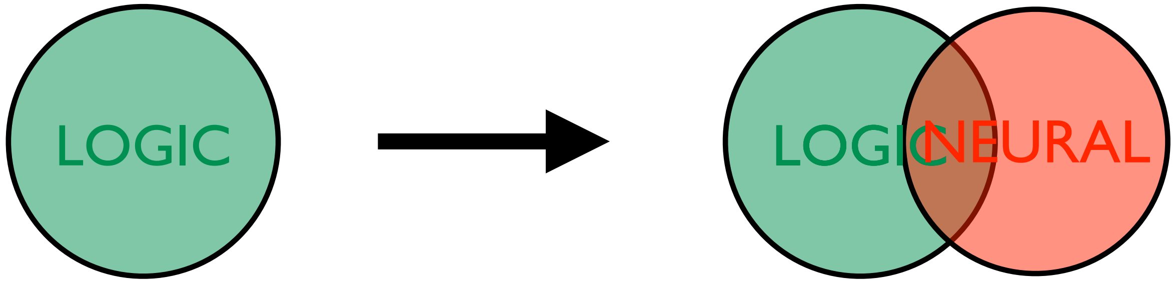
{edge(1,2), path(1,2)}

First-order logic can have many models

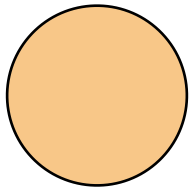
{edge(1,2), path(1,2)}
{edge(1,2), path(1,2), path(1,1)}
{edge(1,2), path(1,2), path(2,1)}

LOGIC

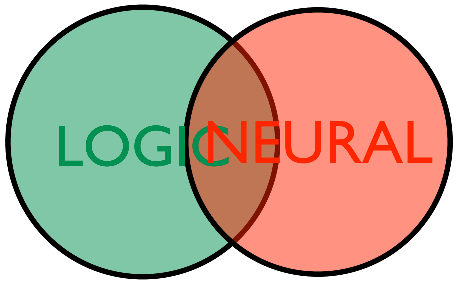
3. Types of Logic



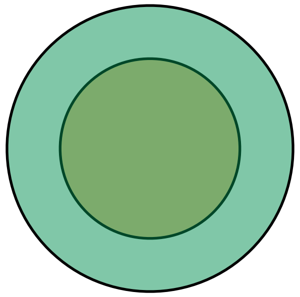
Logic in NeSy - Propositional logic



Semantic loss

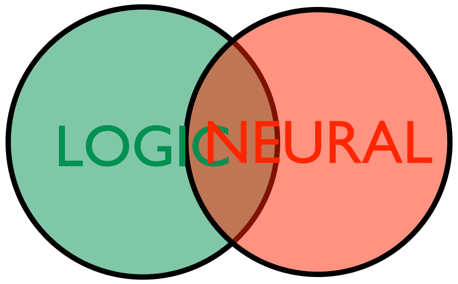


Logic in NeSy - Datalog

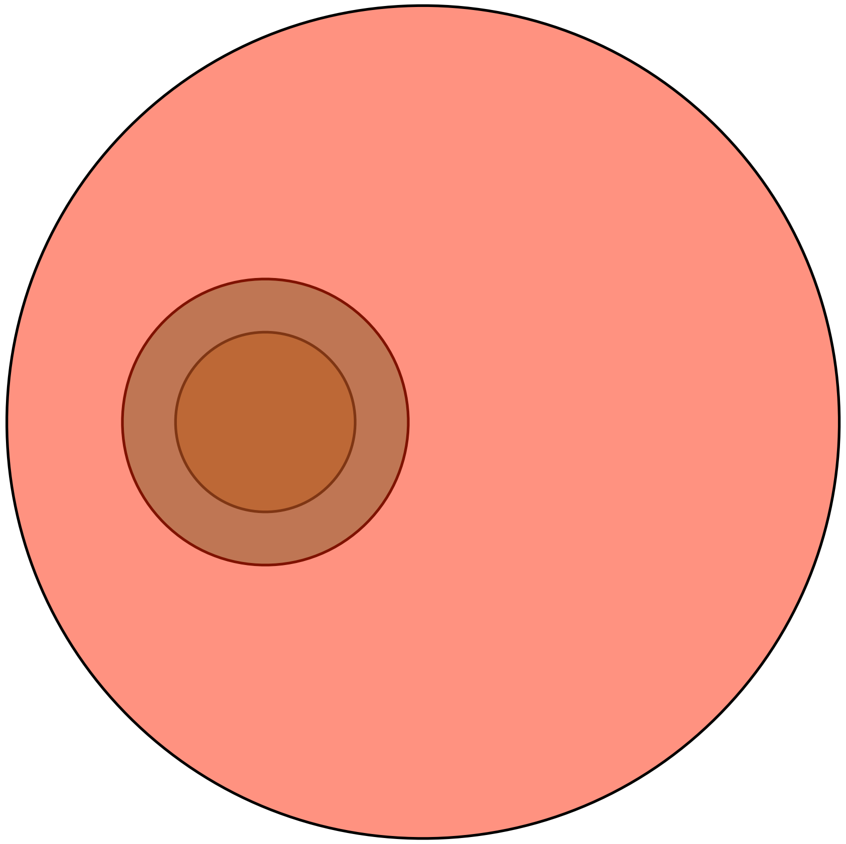


∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



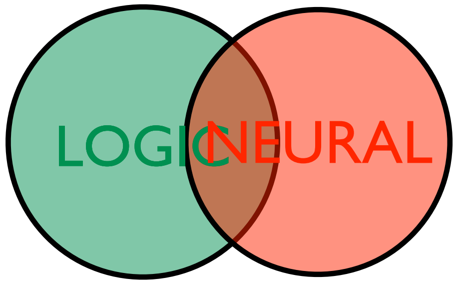
Logic in NeSy - Answer-set programming



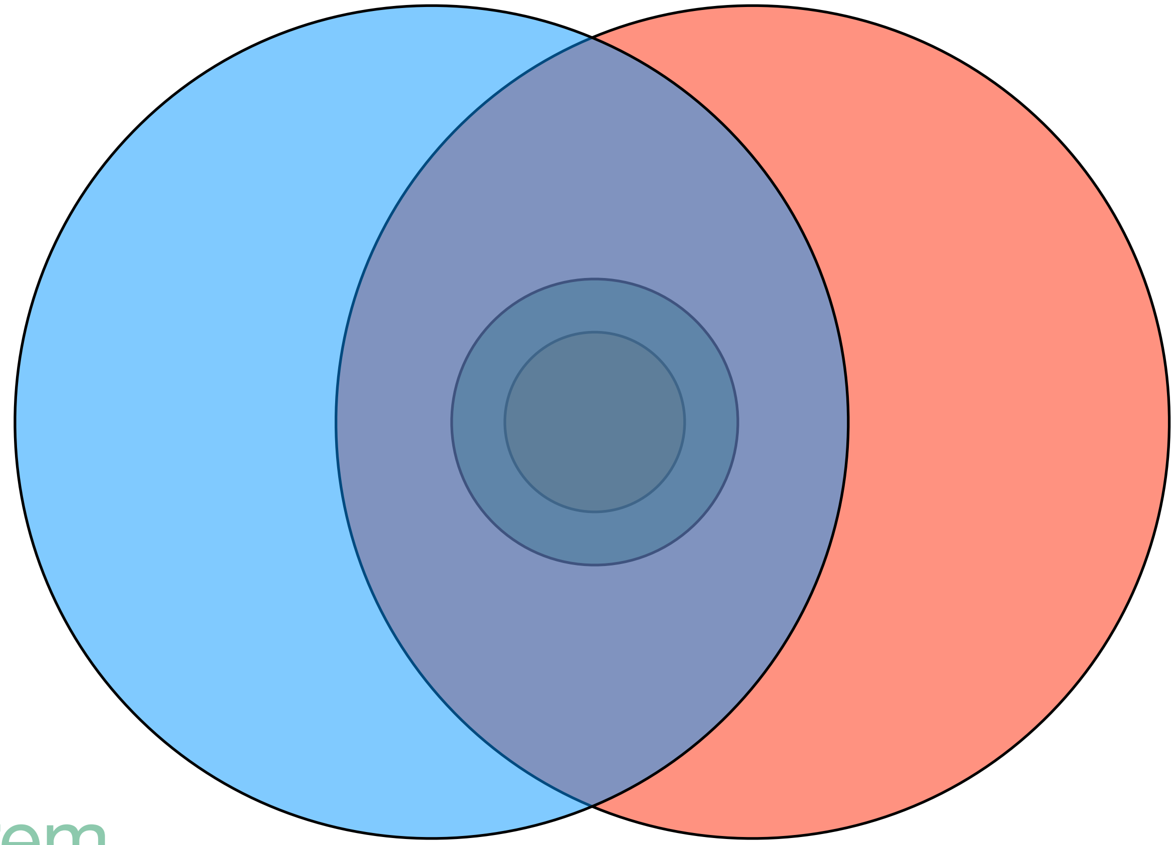
NeurASP

∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



Logic in NeSy - Logic programming

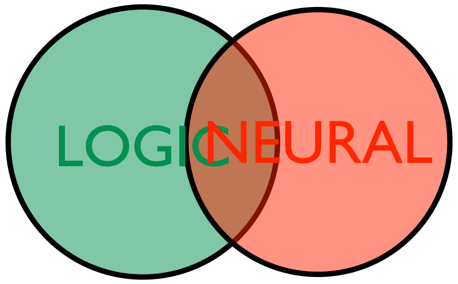


DeepProblog,
NLProlog

NeurASP

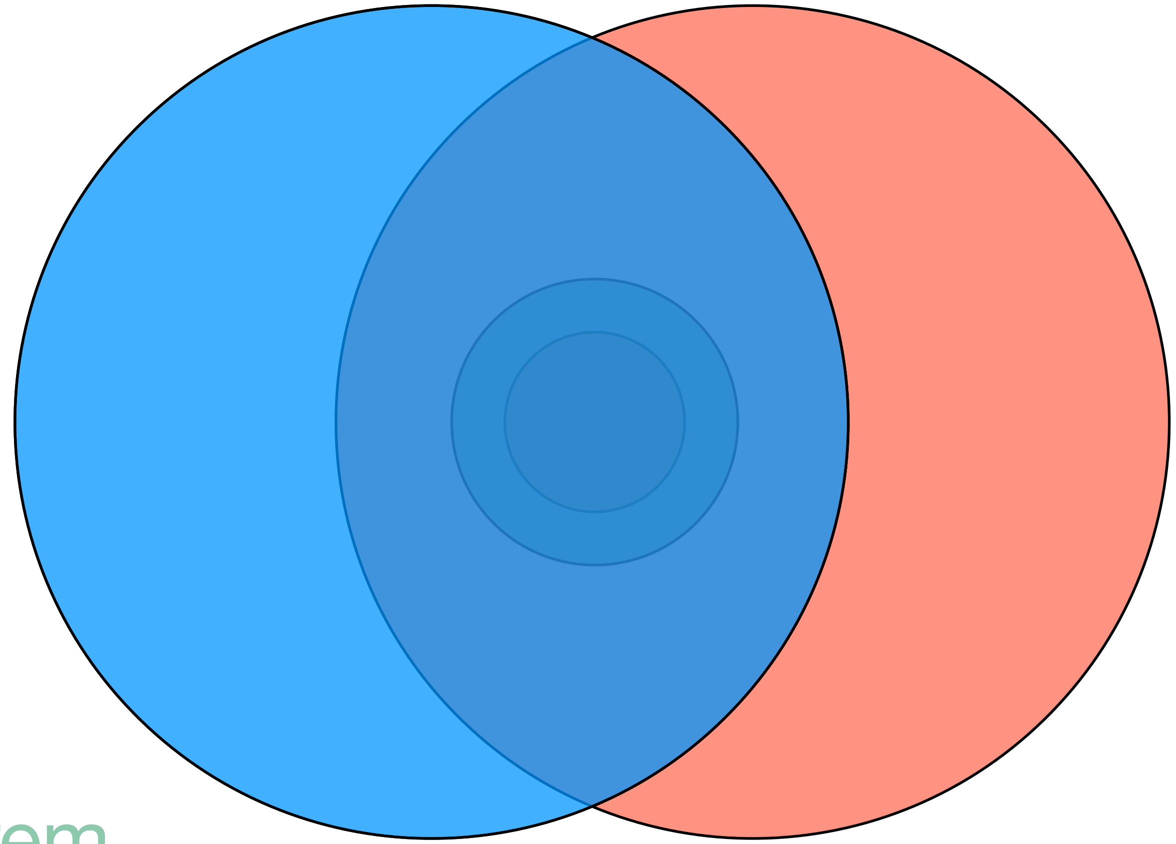
∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



Logic in NeSy - First-order logic

Logic tensor networks, NMLN,
SBT, RNM

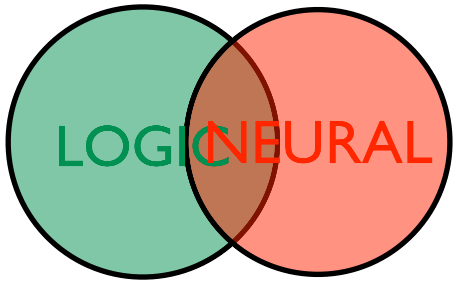


DeepProblog,
NLProlog

NeurASP

∂ILP, Neural Theorem
Provers, LRNN, DiffLog, …

Semantic loss



3. Types of Logic

Key Messages

- Different types of logic exist
- Different types of logic enable different functionalities

4. Symbolic vs sub-symbolic

4. Symbolic vs sub-symbolic

Key Messages

- Entities are represented very differently in symbolic and sub-symbolic systems, but they are complementary
- NeSy systems differ in how they integrate symbolic and sub-symbolic properties

4. Symbolic vs sub-symbolic

LOGIC

Entities in symbolic AI

- Atoms: an, bob
- Numbers: 4, -3.5
- Variables: X,Y

- Structured terms
  - mother(an,bob)
  - [1,3,5]
  - plus(3,times(2,5))

However, symbols have no inherent meaning

mother(an, bob)
brother(bob, charlie)
mother(X, charlie)
children(an, [bob, charlie])

vs

f(x, y)
g(y, z)
f(W, z)
h(x, [y, z])

LOGIC

Symbolic unification

- Powerful mechanism for symbol matching
  - basis for many AI systems

- Finds substitution θ such that both symbols match
  - mother(X, bob) = mother(an, Y)
  - θ = {X = an, Y = bob}
- Not useful to determine similarity
  - mother(an,bob) ≈ mother(an,charlie)?

LOGIC

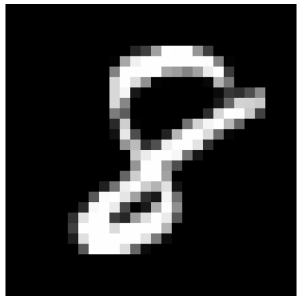
4. Symbolic vs sub-symbolic

NEURAL

Entities in sub-symbolic AI

Sub-symbolic systems require different representation
Let's call these non-symbolic representation sub-symbols

Entities are already sub-symbolic



0.1	-0.3	...
-0.9	-0.2	...
...	...	...

The transformation is straight-forward

The quick brown fox ...

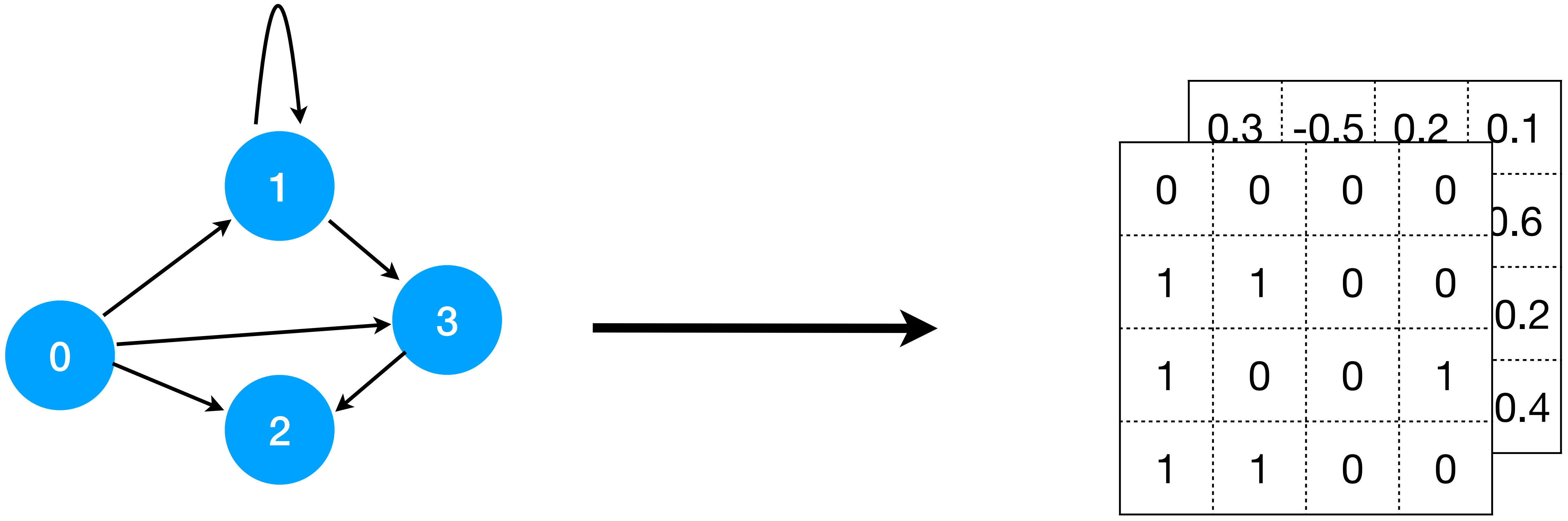
132	32	204	...

-0.8	0.4	0.6	...

NEURAL

Entities in sub-symbolic AI

The transformation is not straight-forward



0	0	0	0
1	1	0	0
1	0	0	1
1	1	0	0

NEURAL

Sub-symbols in StarAI

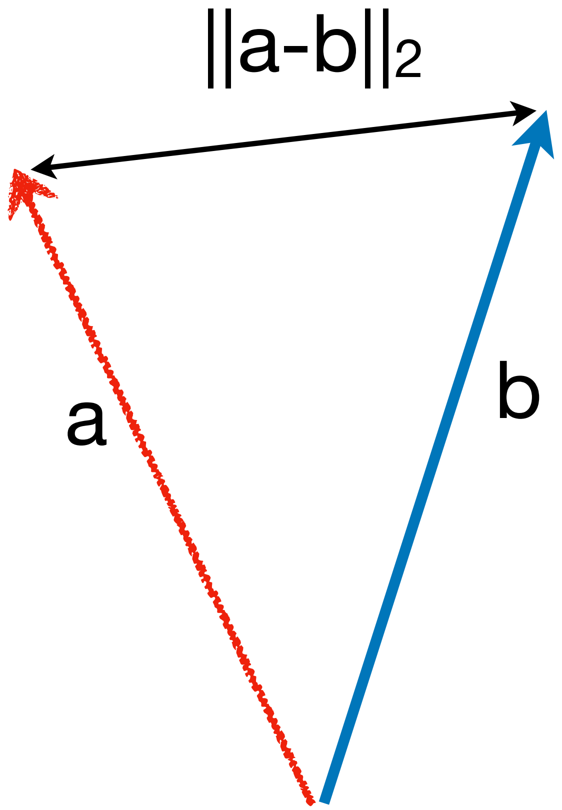
- It is possible to represent these sub-symbols in logic
  - vectors: [0.1, -0.5, 0.6]
  - matrices: [[0.2,0.4],
    [0.3, 0.1]]

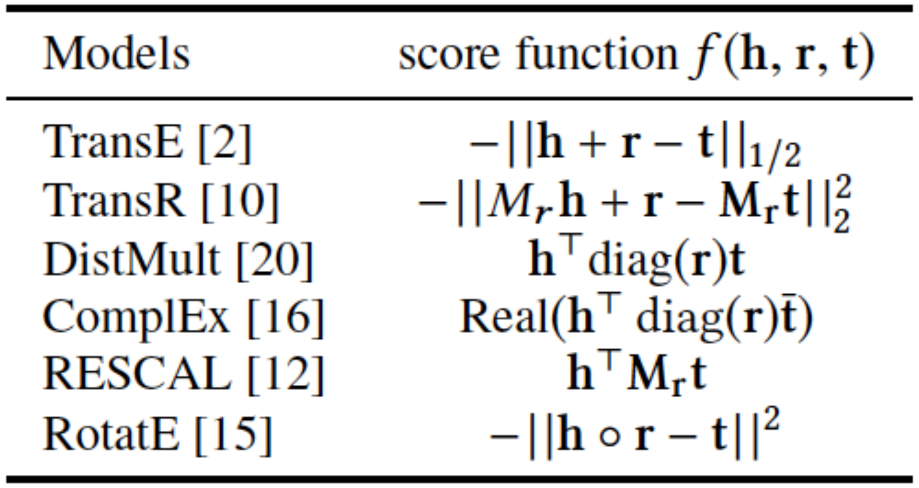
- However, they are not part of the computation mechanisms.
  - i.e. we cannot learn its parameters
- They are not first class citizens.

LOGIC

Comparing sub-symbols

- Similarity can be determined through various metrics
    - L1, L2, radial-basis function, ...
- Can only give a degree of similarity
- When is a ≠ b? When is a = b?
- Generalizability
- Encoding relations r(h,t)
    - Many ways to structure embedding space

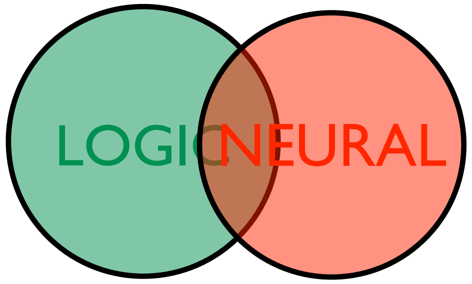




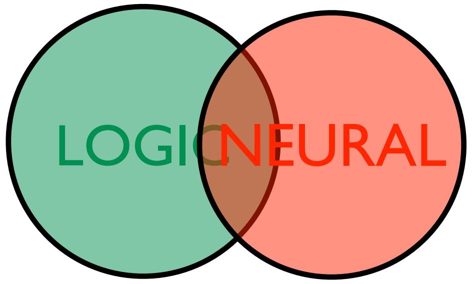
Models	score function $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$
TransE [2]	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|_{1/2}$
TransR [10]	$-\|\|M_r\mathbf{h}+\mathbf{r}-\mathbf{M_r}\mathbf{t}\|\|_2^2$
DistMult [20]	$\mathbf{h}^\top \mathrm{diag}(\mathbf{r})\mathbf{t}$
ComplEx [16]	$\mathrm{Real}(\mathbf{h}^\top \mathrm{diag}(\mathbf{r})\bar{\mathbf{t}})$
RESCAL [12]	$\mathbf{h}^\top \mathbf{M_r}\mathbf{t}$
RotatE [15]	$-\|\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|\|^2$

NEURAL

4. Symbolic vs sub-symbolic



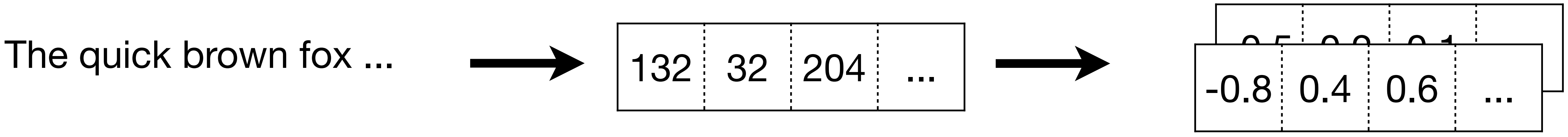
4. Symbolic vs sub-symbolic
Symbols as sub-symbols



Symbols as sub-symbols

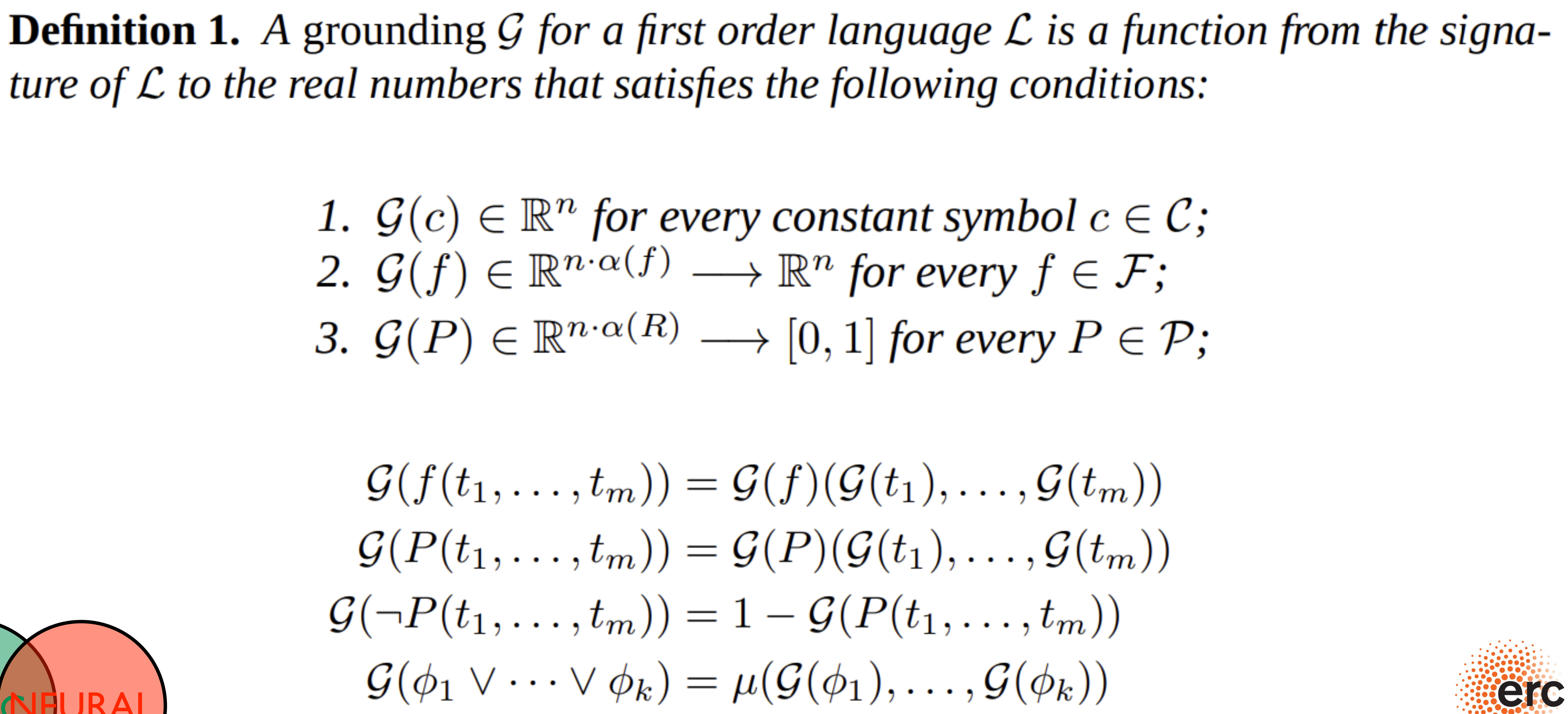
- Symbols are replaced with sub-symbols
  - One-hot encoding
  - Embeddings
  - Inherent numerical properties

- Natural in systems that are originate from a neural base
  - LTN, NLM, ...



Logic Tensor Network

- These translations are made explicit in Logic Tensor Networks



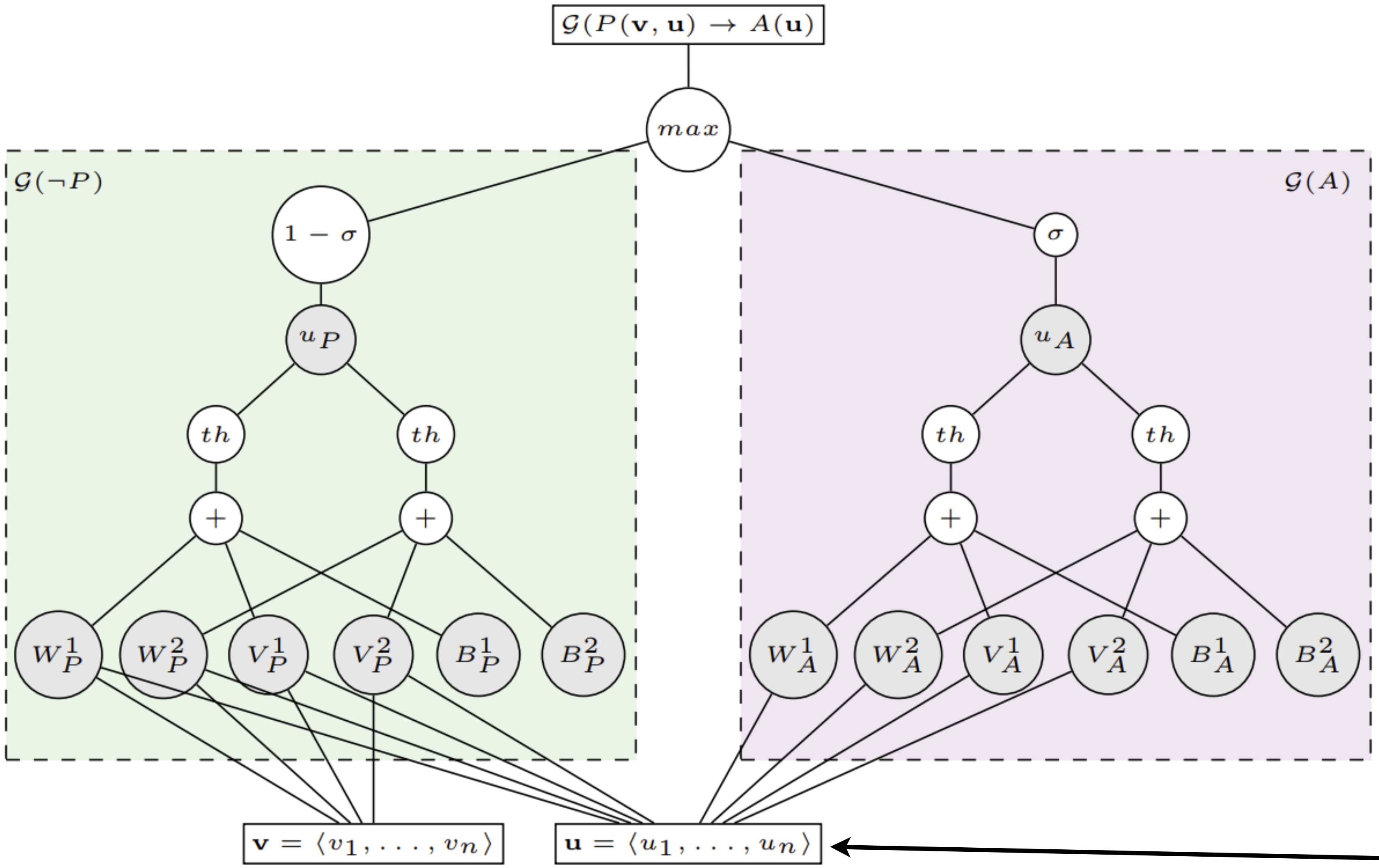
Definition 1. *A grounding $\mathcal{G}$ for a first order language $\mathcal{L}$ is a function from the signature of $\mathcal{L}$ to the real numbers that satisfies the following conditions:*

1. $\mathcal{G}(c) \in \mathbb{R}^n$ *for every constant symbol* $c \in \mathcal{C}$;
2. $\mathcal{G}(f) \in \mathbb{R}^{n \cdot \alpha(f)} \longrightarrow \mathbb{R}^n$ *for every* $f \in \mathcal{F}$;
3. $\mathcal{G}(P) \in \mathbb{R}^{n \cdot \alpha(R)} \longrightarrow [0, 1]$ *for every* $P \in \mathcal{P}$;

$$\begin{aligned}
\mathcal{G}(f(t_1, \ldots, t_m)) &= \mathcal{G}(f)(\mathcal{G}(t_1), \ldots, \mathcal{G}(t_m)) \\
\mathcal{G}(P(t_1, \ldots, t_m)) &= \mathcal{G}(P)(\mathcal{G}(t_1), \ldots, \mathcal{G}(t_m)) \\
\mathcal{G}(\neg P(t_1, \ldots, t_m)) &= 1 - \mathcal{G}(P(t_1, \ldots, t_m)) \\
\mathcal{G}(\phi_1 \vee \cdots \vee \phi_k) &= \mu(\mathcal{G}(\phi_1), \ldots, \mathcal{G}(\phi_k))
\end{aligned}$$

Luciano Serafini, Artur S. d'Avila Garcez: Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge. NeSy@HLAI 2016

Logic Tensor Network

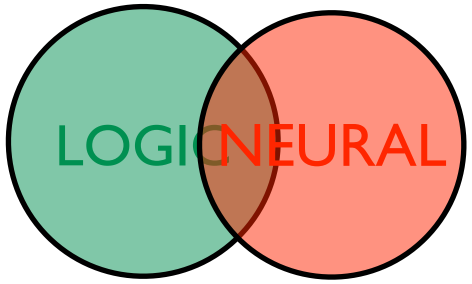


Sub-symbolic computation

Encoding symbols

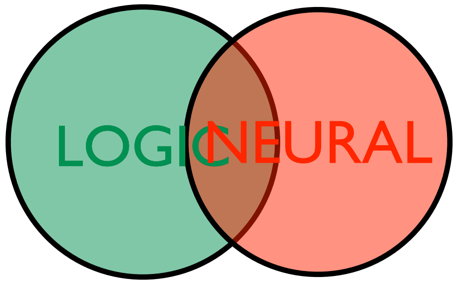
Luciano Serafini, Artur S. d'Avila Garcez: Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge. NeSy@HLAI 2016

4. Symbolic vs sub-symbolic
Sub-symbols as symbols



Sub-symbols as symbols

- The sub-symbolic nature is not considered in the logic
  - Tensors, vectors, … are treated as symbols
  - Sub-symbolic properties are not directly used in the logic
- Difference with StarAI systems
  - sub-symbolic properties are used on the neural side
  - usually differentiable / learnable
- Natural in systems that are originate from a logic base
  - DeepProbLog, NeurASP, ...



Sub-symbols as symbols: DeepProbLog

- DeepProbLog: interface between PLP (ProbLog) and neural networks.
- This interface takes the form of the neural predicate
  - Output of neural networks represented as probabilistic facts

```
nn(mnist_net, [D], N, [0 ... 9] ) :: digit(D,N).
addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```

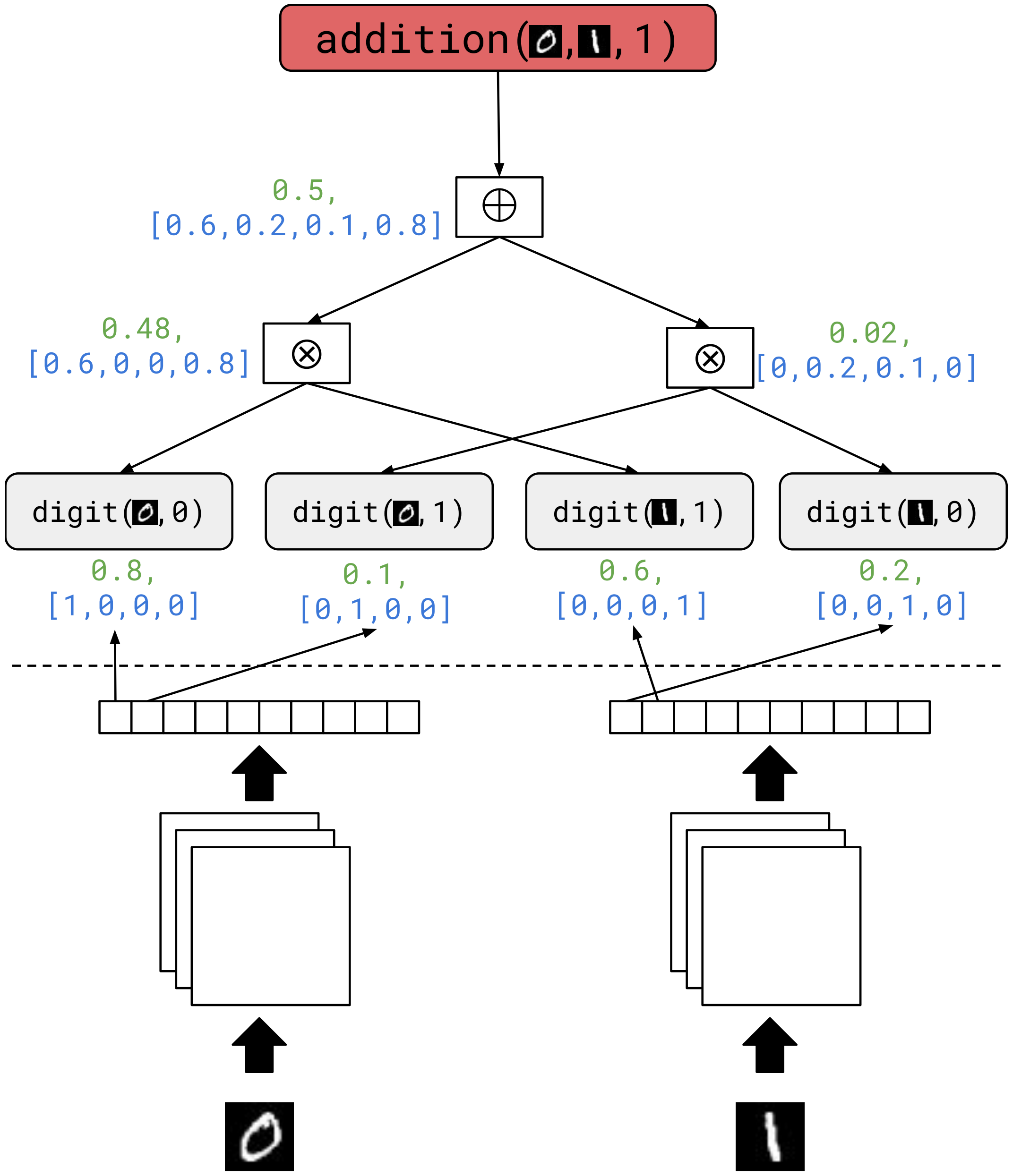
- In the logic, the images are represented as constants
- Sub-symbolic properties are used in the neural network to make predictions
- This may seem as a limitation, but isn't

Examples:

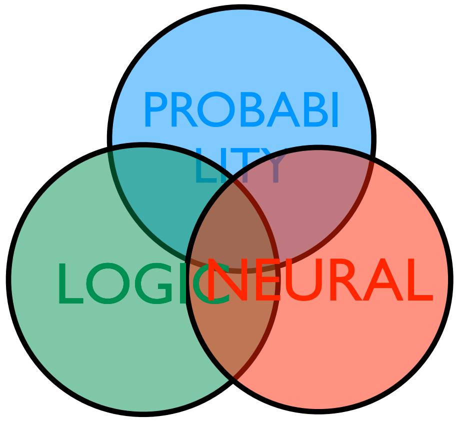
addition(, ,8), addition(,,4), addition(,,11), …

Sub-symbols as symbols: DeepProbLog

```
nn(mnist_net, [X], Y, [0 ... 9] ) ::
     digit(X,Y).

addition(X,Y,Z) :-
   digit(X,N1),
   digit(Y,N2),
   Z is N1+N2.
```

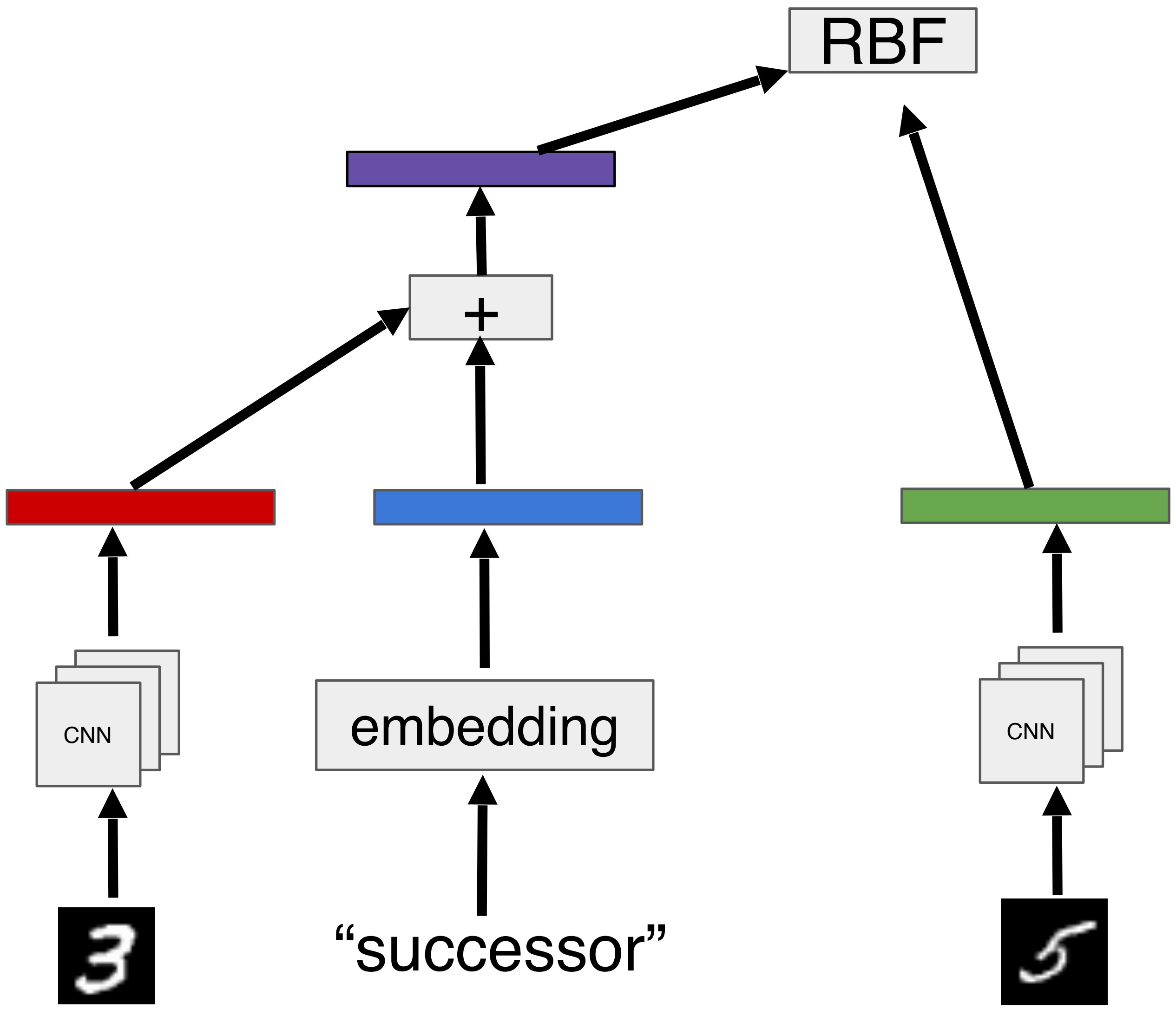


The ACs are differentiable and there is an interface with the neural nets



Embeddings as symbols

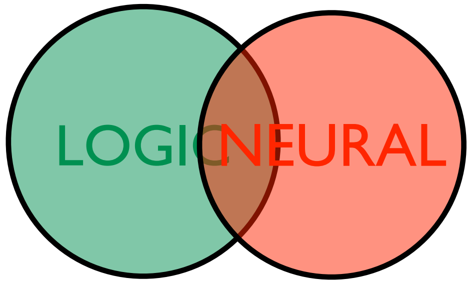
Computational Graph



```
succesor_n(3, 5) :-
        cnn_embed( 3,e1),
        cnn_embed( 5,e2),
        embed(“successor”,r),
        add(r,e1,e3),
        rbf(e2,e3).
```

Idea of TransE [Bordes et al]

4. Symbolic vs sub-symbolic
Sub-symbols as labels



Sub-symbols as labels
T-PRISM

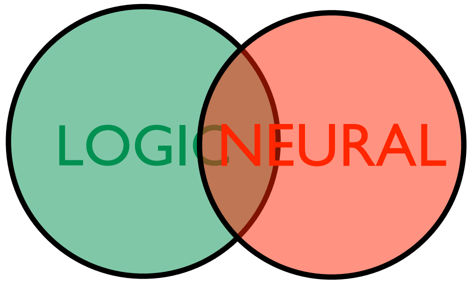
- StarAI
  - probabilites are used as labels
- labels are combined in inference (cfr. arithmetic circuit)
- In this integration, labels are sub-symbols instead
- Example: T-PRISM

```
rel(S,R,O):-
   tensor(v(S),[i]),
   tensor(v(O),[i]),
   tensor(r(R),[i]).
```

$$
\begin{aligned}
&\mathrm{label}(\mathrm{rel}(S,R,O)) \\
&= \mathrm{label}(S_i \wedge O_i \wedge R_i) \\
&= \textstyle\sum_i s_i o_i r_i \\
&= \mathrm{DistMult}(s,o,r)
\end{aligned}
$$

Ryosuke Kojima, Taisuke Sato: A tensorized logic programming language for large-scale data. CoRR abs/1901.08548 (2019)
Yang et al.: Embedding entities and relations for learning and inference in knowledge bases. (ICLR 2015)

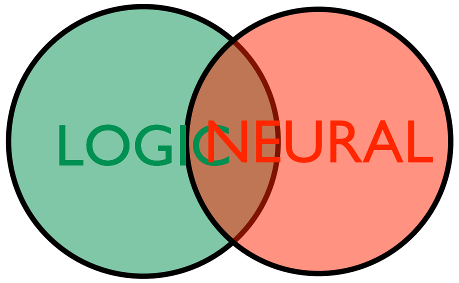
4. Symbolic vs sub-symbolic Neural Theorem Prover



Neural Theorem Prover

- The neural theorem prover uses both symbols and sub-symbols simultaneously
- Symbols retain their symbolic nature
- Each symbol has a learnable sub-symbol T

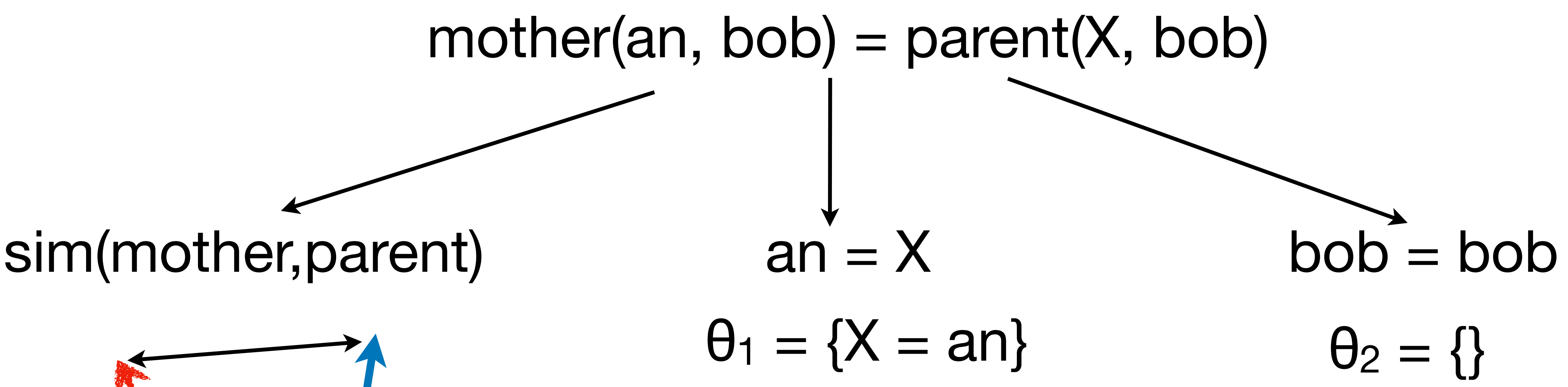
- Symbol comparison:
  - Normal unification
- Comparison of sub-symbols:
  - $\mathrm{sim}(x,y) = \exp(- ||T_x - T_y||_2)$



Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." Advances in Neural Information Processing Systems. 2017.

Soft unification

- Unify what can be unified
- Use similarity to compare other symbols and use it as a score



mother(an, bob) = parent(X, bob)

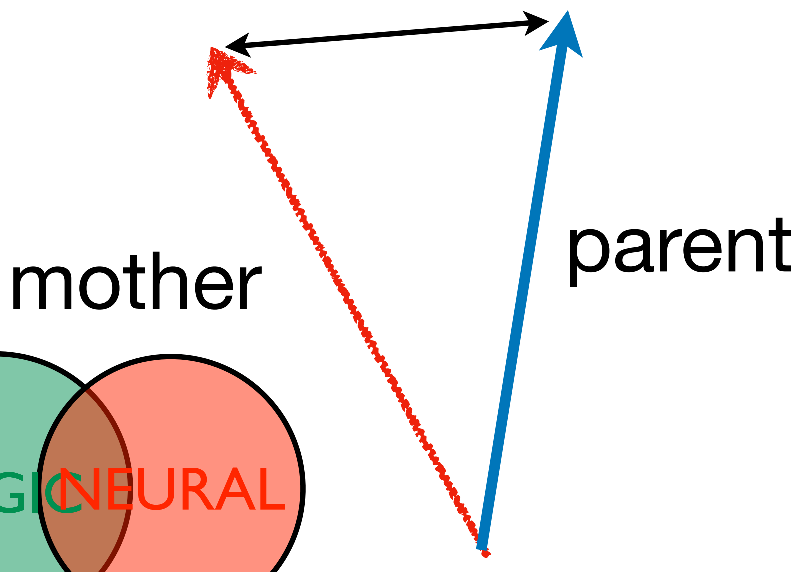
sim(mother,parent)

an = X

$\theta_1 = \{X = an\}$

bob = bob

$\theta_2 = \{\}$



Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." Advances in Neural Information Processing Systems. 2017.

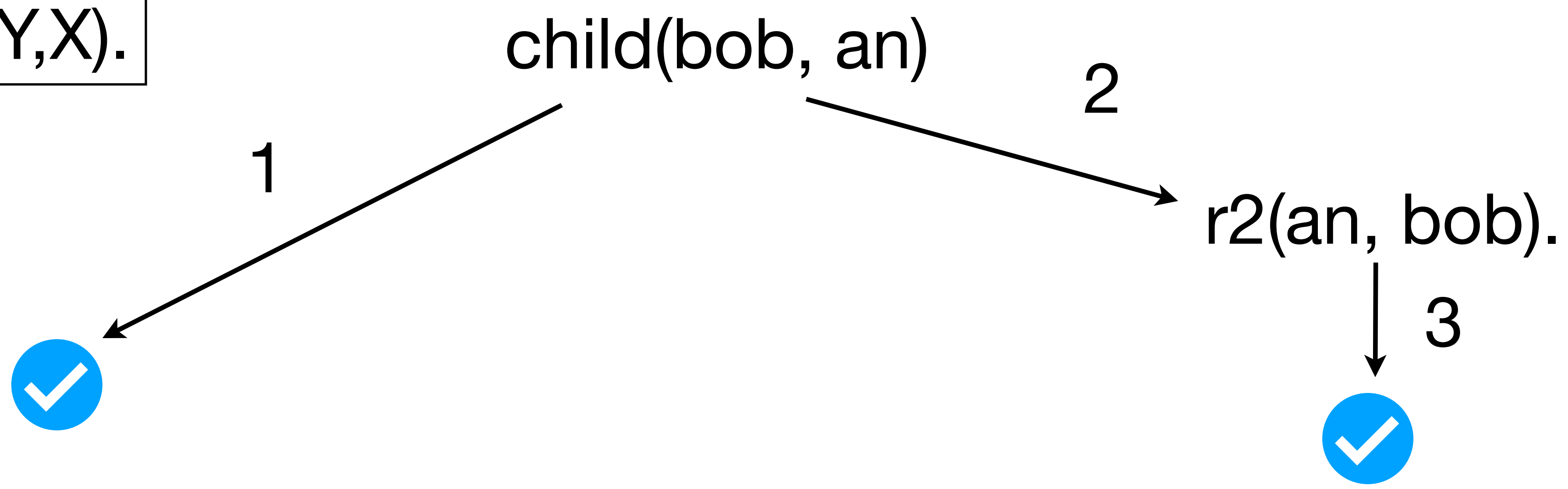
End-to-end differentiable proving

- OR module
  - Apply every rule whose head soft-unifies with the goal
  - Uses AND module to prove sub-goals in body
- AND module
  - Prove conjunction of sub-goals
  - Uses OR module to prove first goal
  - Uses AND module to recursively prove

Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." Advances in Neural Information Processing Systems. 2017.

Example

mother(an, bob).
r1(X,Y) :- r2(Y,X).



child(bob, an)

1

✓

2

r2(an, bob).

3

✓

Unifications

1) mother(an,bob) = child(bob,an)
sim(mother,child)
sim(an,bob)

2) r1(X,Y) = child(bob,an)
sim(r1,child)
X = bob
Y = an

3) r2(an, bob) = mother(an, bob)
sim(r2,mother)

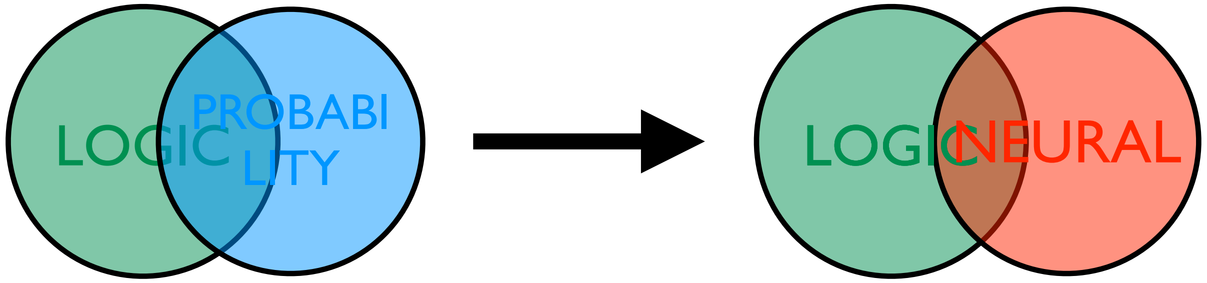
Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." Advances in Neural Information Processing Systems. 2017.

4. Symbolic vs sub-symbolic

Key Messages

- Entities are represented very differently in symbolic and sub-symbolic systems, but they are complementary
- NeSy systems differ in how they integrate symbolic and sub-symbolic properties

5. Structure vs parameter learning

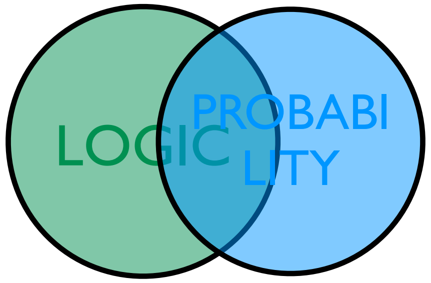


5. Learning

Key Messages

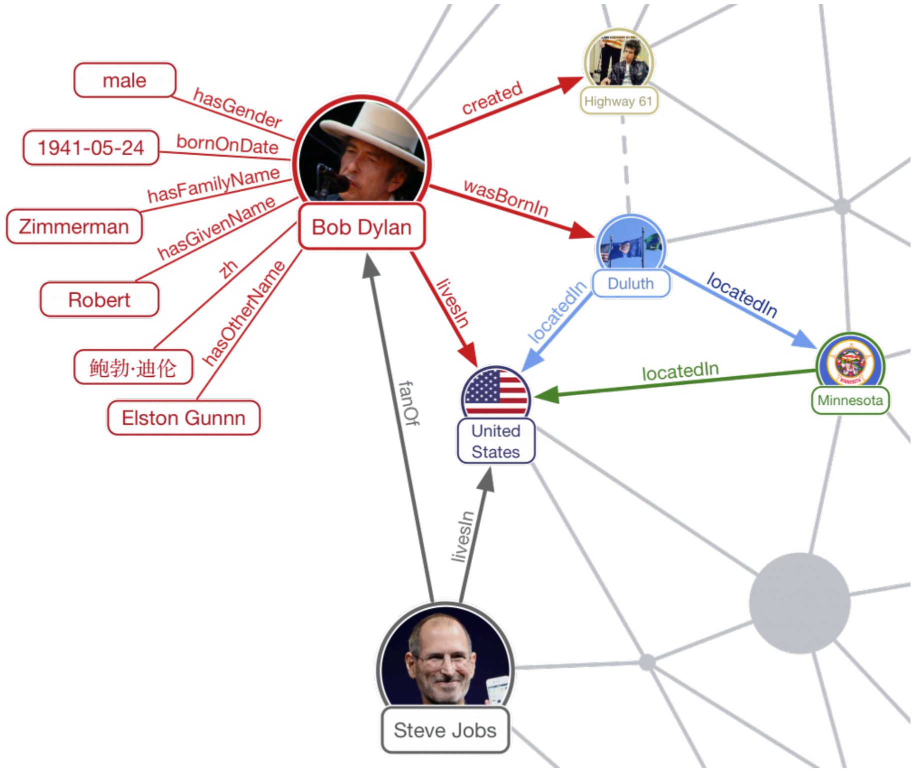
- Learning: finding logical formulas and estimating probabilities
- Structure learning: both formulas and probabilities
- Parameter learning: only probabilities
- Many flavours of learning in NeSy

5. Structure vs parameter learning



Learning in StarAI

Obtaining models from data



↦

0.7::nationality(X,Y) :-
livesIn(X,Y).

0.7::nationality(X,Y) :-
livesIn(X,Z), locatedIn(Z,Y).

0.9::nationality(X,Y) :-
bornIn(X,Y).

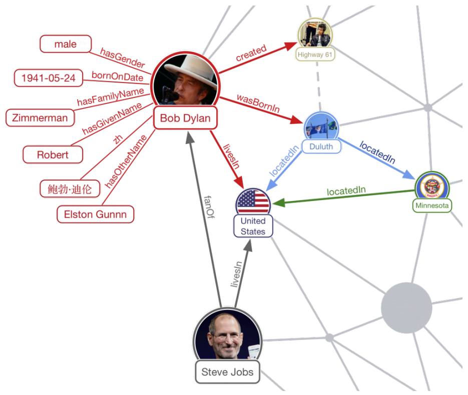
StarAI learning paradigms

	Structure learning	**Parameter learning**
What is provided?	Data	Data and discrete structure
What is the learning goal?	Structure and parameters	Parameters

Learning types: Parameter learning

Learning the probabilities/weights of a specified model

Model (the formulas) are given



nationality(X,Y) :-
livesIn(X,Y).

nationality(X,Y) :-
livesIn(X,Z), locatedIn(Z,Y).

nationality(X,Y) :-
bornIn(X,Y).

↦

the goal of learning

0.7::nationality(X,Y) :-
livesIn(X,Y).

0.7::nationality(X,Y) :-
livesIn(X,Z), locatedIn(Z,Y).

0.9::nationality(X,Y) :-
bornIn(X,Y).

Learning types: Parameter learning

Learning the probabilities/weights of a specified model

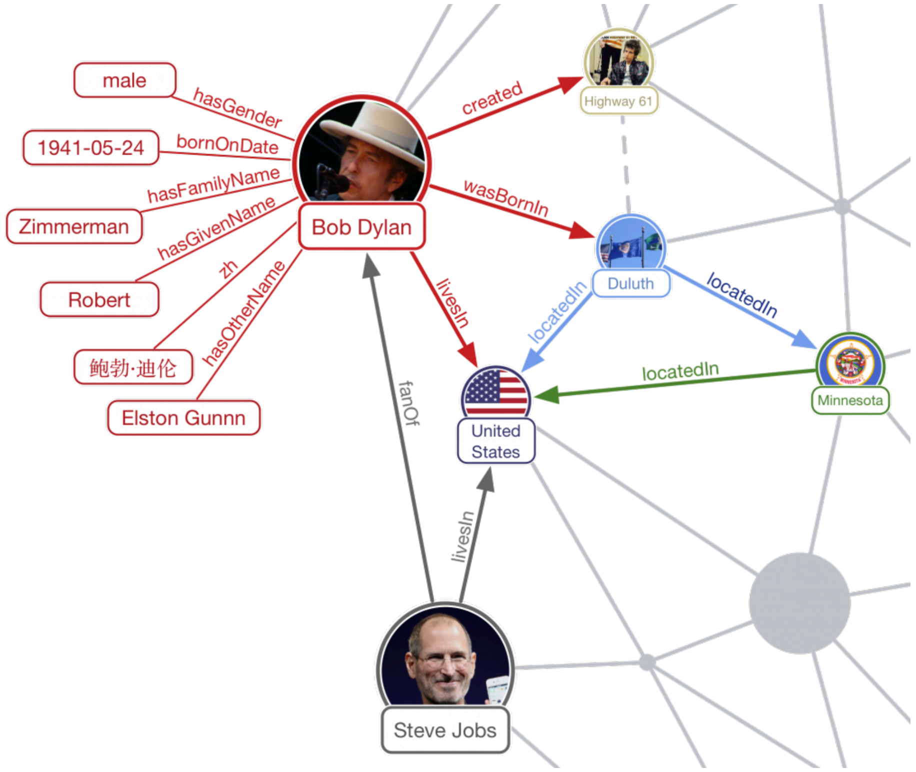
Model (the formulas) are given

Learning principles: identical to learning parameters of any parametric model

- gradient descent [Lowd & Domingos, 2007]
- least squares [Gutmann et al, 2008]
- Expectation Maximisation [Gutmann et al, 2011]

Learning types: Structure learning

Finding the clauses/logical formulas of a model



$\mapsto$

the goal of learning

0.7::nationality(X,Y) :-
livesIn(X,Y).

0.7::nationality(X,Y) :-
livesIn(X,Z), locatedIn(Z,Y).

0.9::nationality(X,Y) :-
bornIn(X,Y).

Learning types: Structure learning

Two types of structure learning

Discriminative

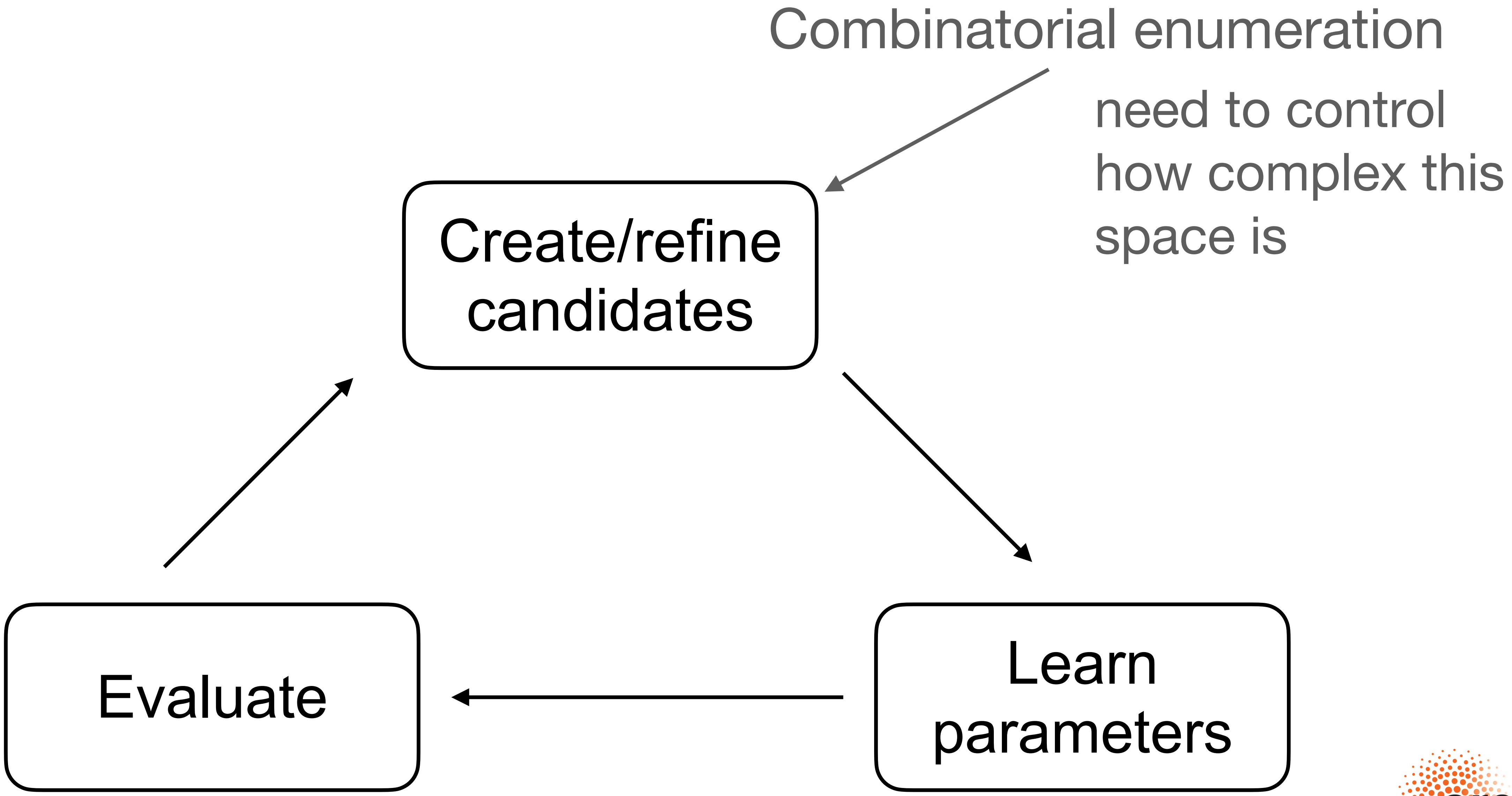
- specific target relation
- separate background knowledge

Generative

- no specific target relation
- learning generative process behind data

Learning types: Structure learning

Learning by searching



Combinatorial enumeration

need to control how complex this space is

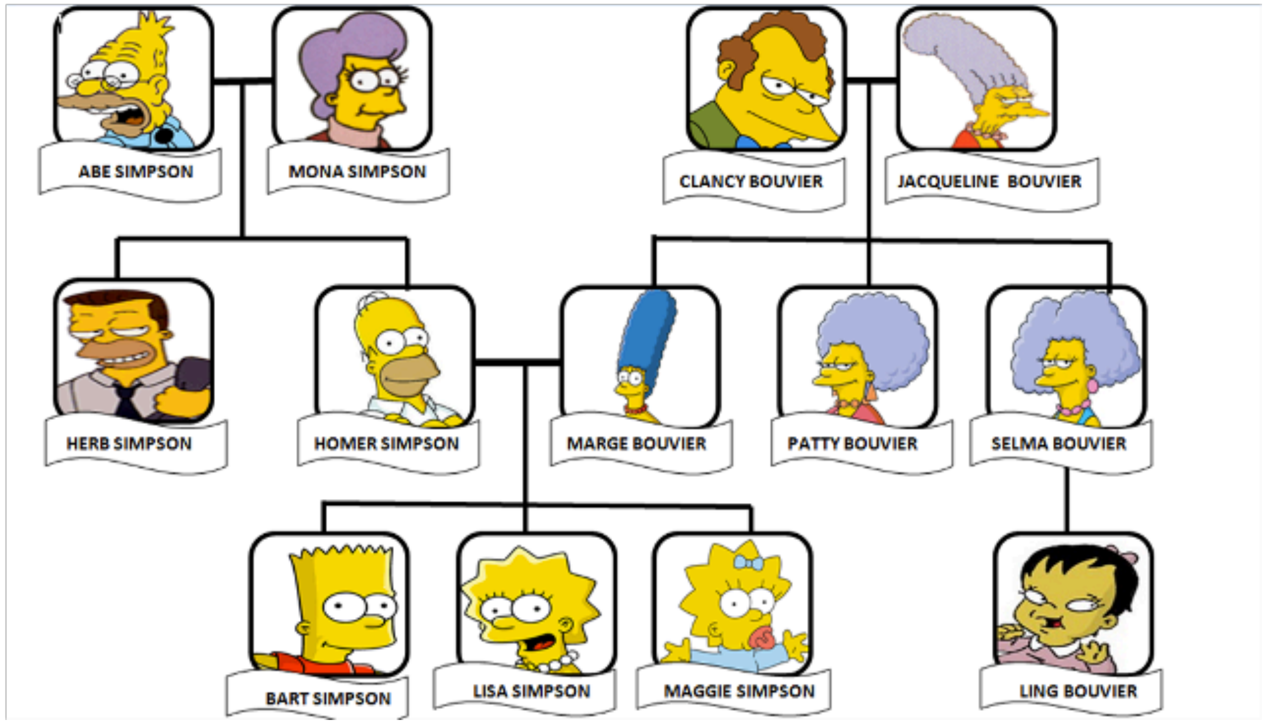
Create/refine candidates

Learn parameters

Evaluate

Learning via enumeration - Probfoil+

[De Raedt et al, 2015]



grandparent(abe,lisa).
grandparent(abe,bart).
grandparent(jacqueline,lisa).
grandparent(jacqueline,maggie.)

Learning via enumeration - Probfoil+

[De Raedt et al, 2015]

Model: {}

{1.0:: grandparent(X,Y) ← mother(X,Z), father(Z,Y)}

start again with a single rule!

if not good enough, refine!

Learn one rule:

p:: grandparent(X,Y) ← *true*

~~p:: grandparent(X,Y) ← *true*~~

p:: grandparent(X,Y) ← mother(X,Y)

~~p:: grandparent(X,Y) ← mother(X,Y)~~

p:: grandparent(X,Y) ← mother(Y,X)

~~p:: grandparent(X,Y) ← mother(Y,X)~~

p:: grandparent(X,Y) ← mother(X,Z)

~~p:: grandparent(X,Y) ← mother(X,Z)~~

p:: grandparent(X,Y) ← father(X,Y)

~~p:: grandparent(X,Y) ← father(X,Y)~~

~~.....~~

....

p:: grandparent(X,Y) ← mother(X,Y),father(X,Z)

....

p:: grandparent(X,Y) ← mother(X,Z),father(Z,Y)

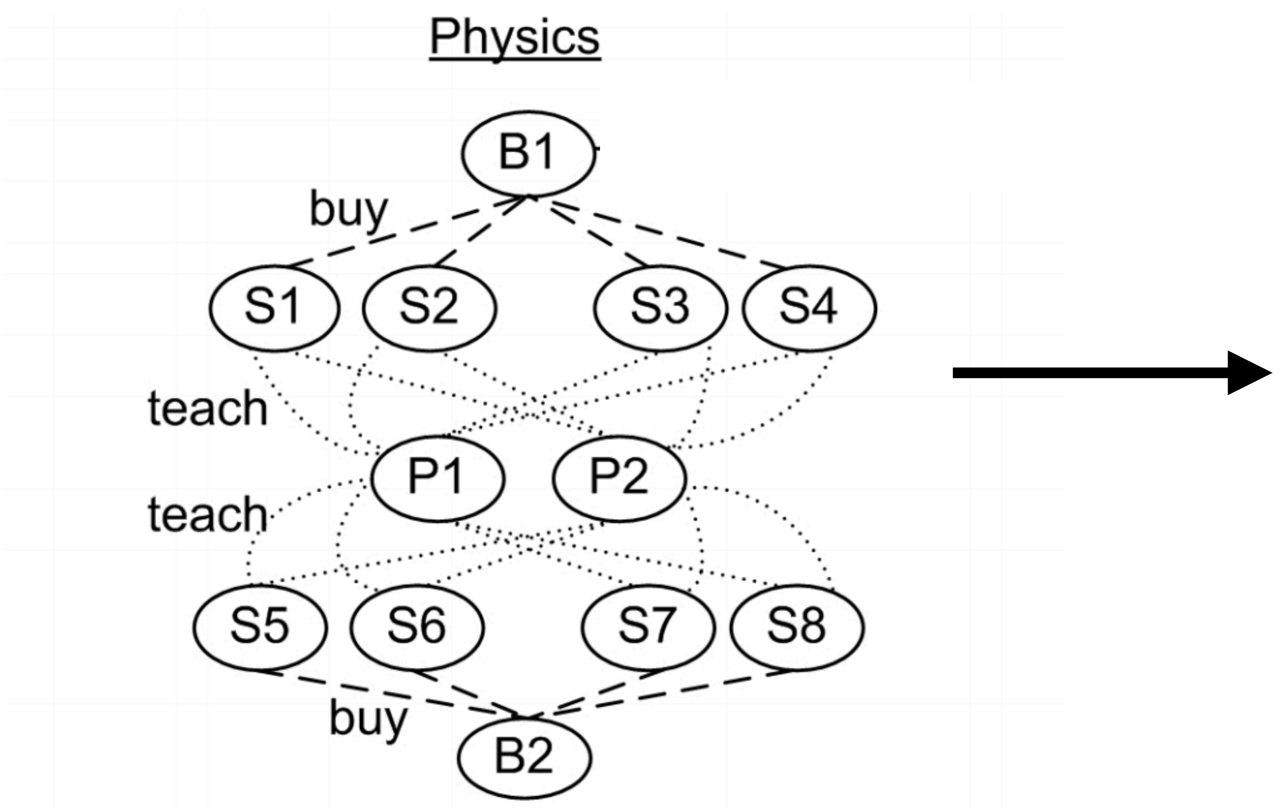
p:: grandparent(X,Y) ← mother(X,Z),mother(Z,Y)

p:: grandparent(X,Y) ← father(X,Y),mother(X,Y)

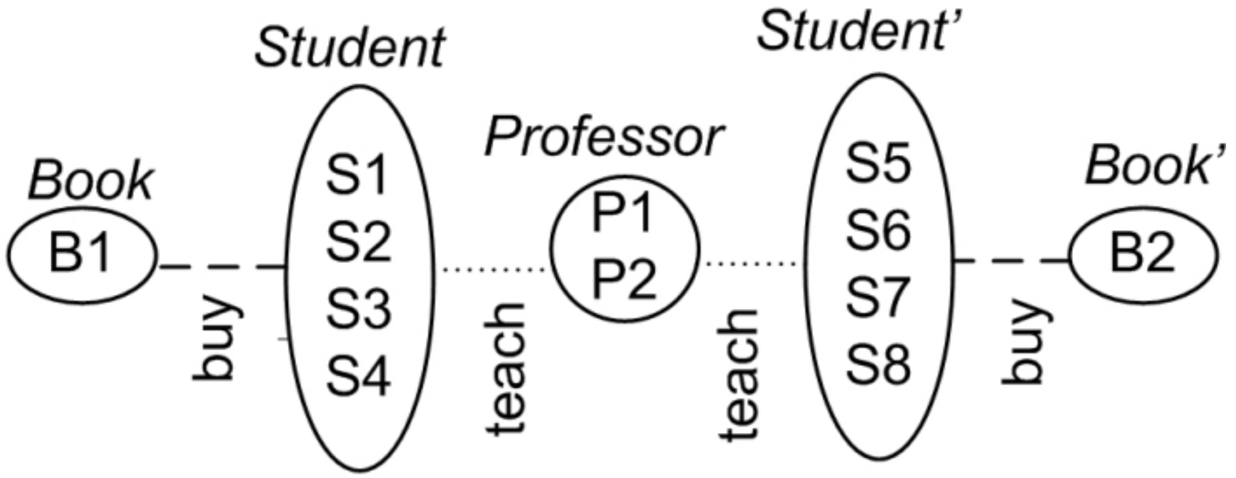
.....

Learning via random walks

[Kok & Domingos, 2009]



“Lift” a knowledge graph by identifying nodes with the same role



Traverse the lifted knowledge graph
and
turn every path into a clause/rule

LOGIC PROBABILITY

Learning in StarAI - overview

Structure learning

- ⊕ Starts directly from data

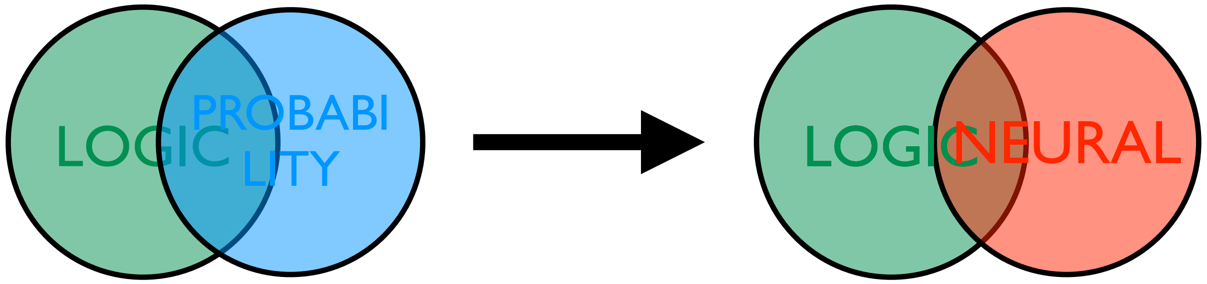
- ⊖ Combinatorial problem
- ⊖ User needs to design a language

Parameter learning

- ⊕ Learning is easier
- ⊕ Scales better

- ⊖ An expert needs to provide the rules
- ⊖ Sensitive to the choice of rules

5. Structure vs parameter learning



Spectrum of learning paradigms

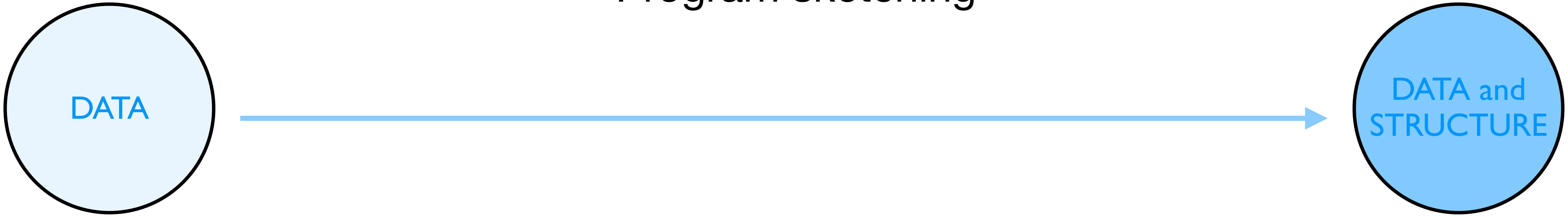
Soft patterns

Neural generation

Structure via
parameter learning

Neurally-guided
learning

Program sketching



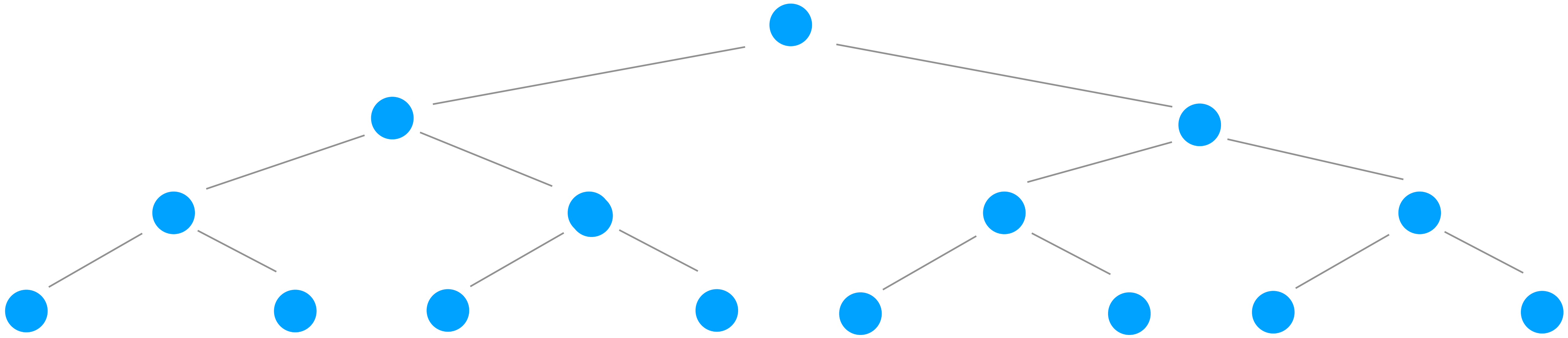
Structure learning

Parameter learning

LOGIC NEURAL

DeepCoder

[Balog et al, 2017]

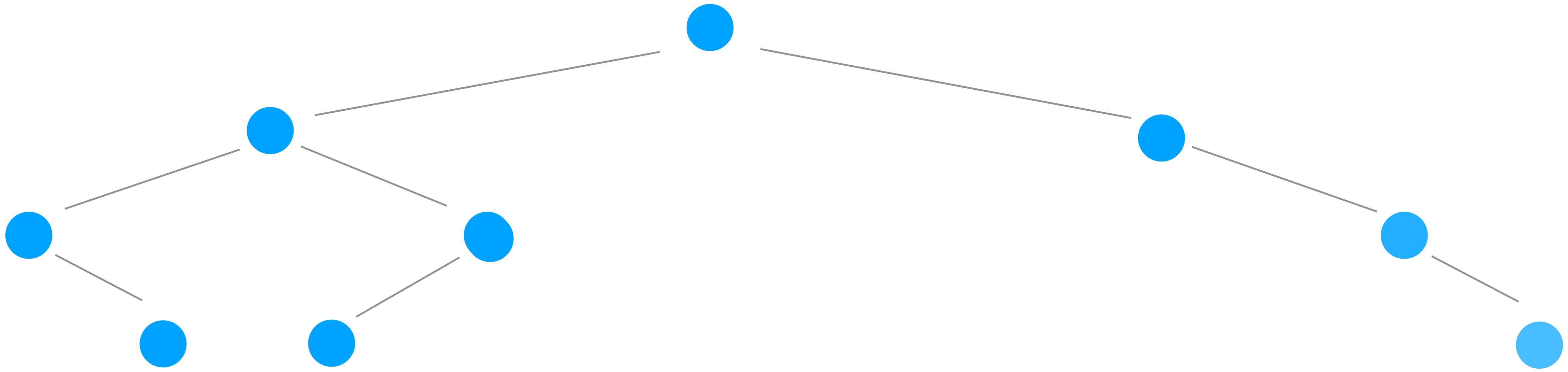


StarAI techniques search for clauses/rules systematically

DeepCoder

[Balog et al, 2017]

Preferences of learning ‘primitives’



Explore the subpart of the space with primitives that are likely to solve the problem

likely to solve a problem = learned from data

DeepCoder

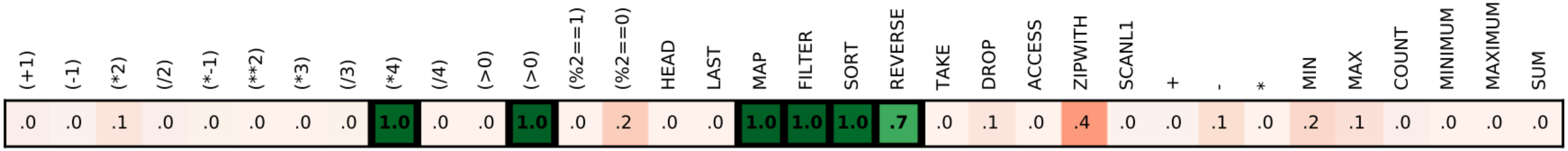
[Balog et al, 2017]

Preferences of learning ‘primitives’

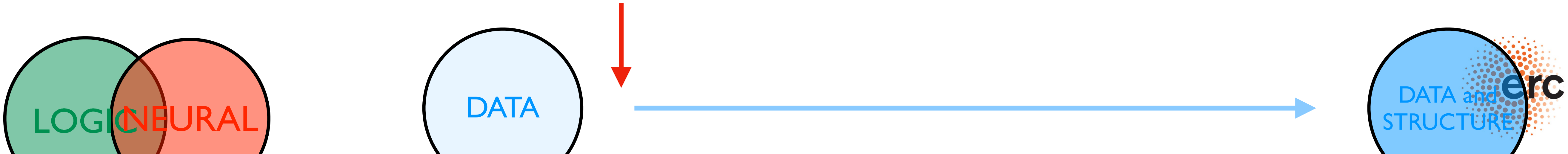
Learn from pairs
(examples, program)

```
a ← [int]
b ← FILTER (<0) a
c ← MAP (*4) b
d ← SORT c
e ← REVERSE d
```

An input-output example:
Input:
`[-17, -3, 4, 11, 0, -5, -9, 13, 6, 6, -8, 11]`
Output:
`[-12, -20, -32, -36, -68]`



(+1)	(-1)	(*2)	(/2)	(*-1)	(**2)	(*3)	(/3)	(*4)	(/4)	(>0)	(>0)	(%2==1)	(%2==0)	HEAD	LAST	MAP	FILTER	SORT	REVERSE	TAKE	DROP	ACCESS	ZIPWITH	SCANL1	+	-	*	MIN	MAX	COUNT	MINIMUM	MAXIMUM	SUM
.0	.0	.1	.0	.0	.0	.0	.0	1.0	.0	.0	1.0	.0	.2	.0	.0	1.0	1.0	1.0	.7	.0	.1	.0	.4	.0	.0	.1	.0	.2	.1	.0	.0	.0	.0



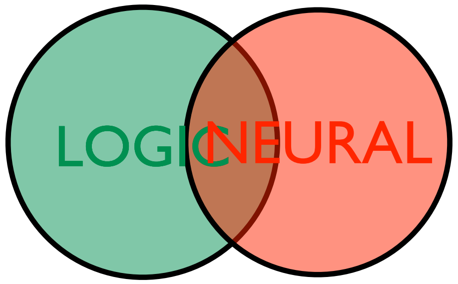
DreamCoder

[Ellis et al, 2018]

Distribution of primitives defines a generative model of programs

$$q(\text{programs} \mid \text{examples})$$

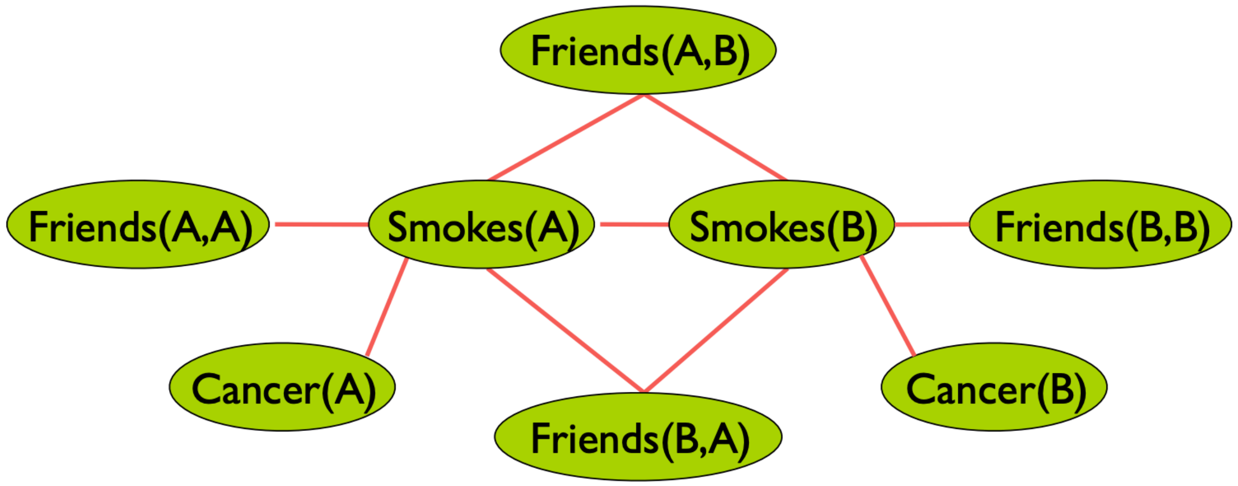
Neural network outputs the posterior distribution over programs likely to solve a specific task



Neural Markov Logic Networks

[Marra et al, 2020]

MLNs can be interpreted as log-linear models



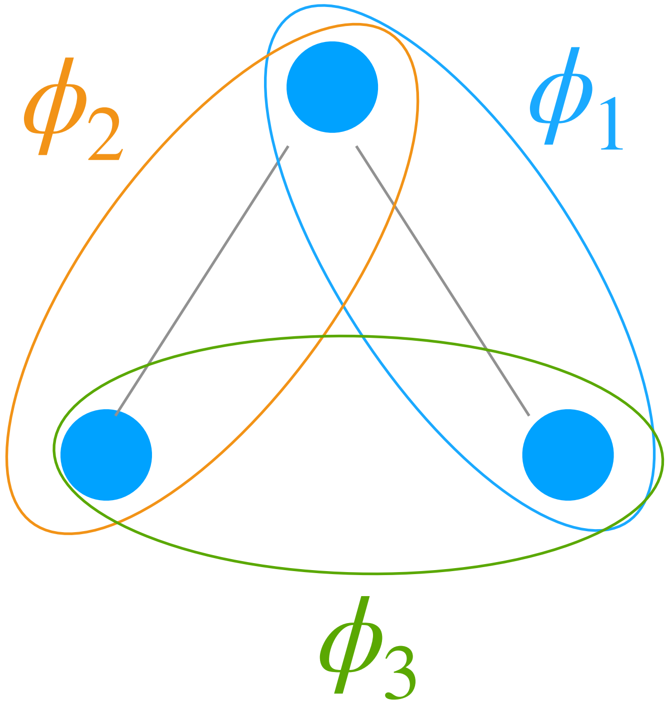
$$P(X = x) = \frac{1}{Z}\prod_i \phi_i(x_{\{i\}})^{n_i(x)}$$

potentials come from formulas provided by the expert (cliques in Markov network)

Neural Markov Logic Networks

[Marra et al, 2020]

Learn neural potentials from fragments of data



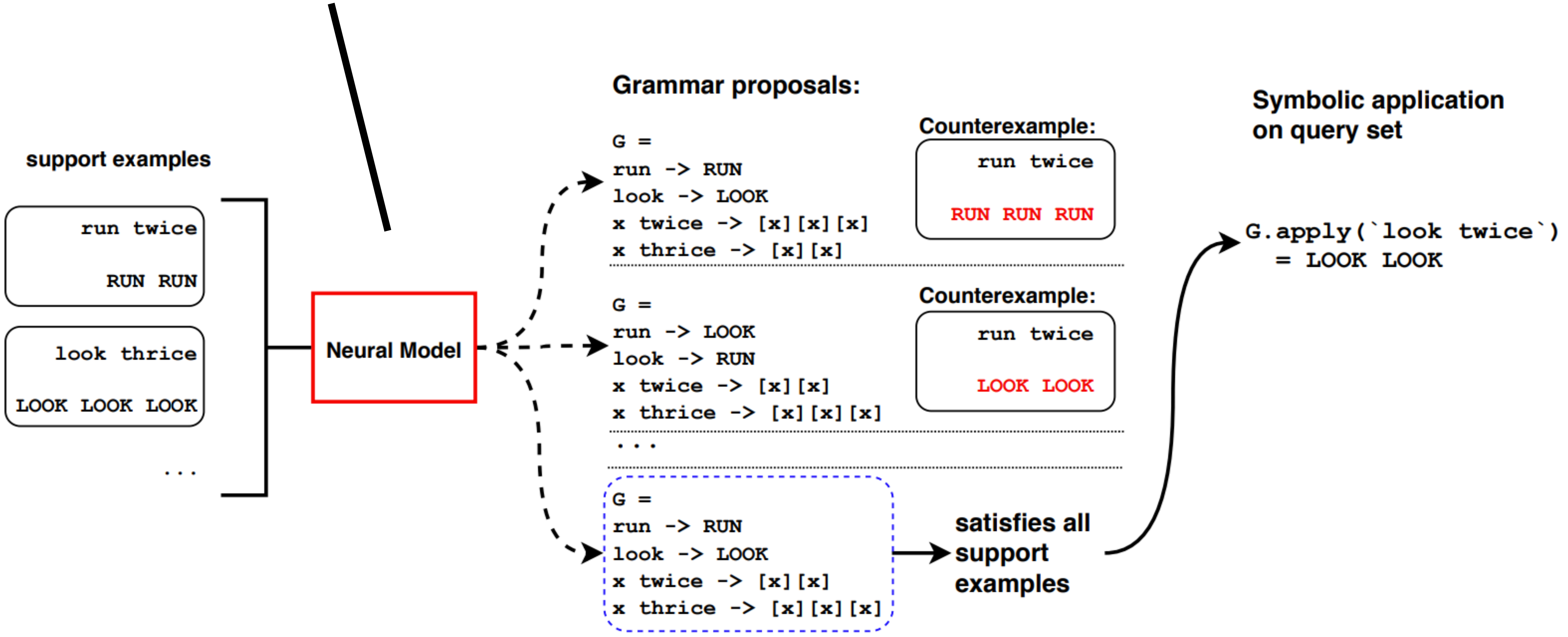
$$P(X = x) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)}$$

potentials come from fragments of data (knowledge graph)

Neural Generation

[Nye et al, 2020]

Neural model generates discrete structure



support examples

```
run twice
RUN RUN
```

```
look thrice
LOOK LOOK LOOK
```

...

Neural Model

Grammar proposals:

```
G =
run -> RUN
look -> LOOK
x twice -> [x][x][x]
x thrice -> [x][x]
```

Counterexample:

```
run twice
RUN RUN RUN
```

```
G =
run -> LOOK
look -> RUN
x twice -> [x][x]
x thrice -> [x][x][x]
```

Counterexample:

```
run twice
LOOK LOOK
```

...

```
G =
run -> RUN
look -> LOOK
x twice -> [x][x]
x thrice -> [x][x][x]
```

satisfies all support examples

Symbolic application on query set

```
G.apply(`look twice`)
  = LOOK LOOK
```

Program sketching

[Bosnjak et al, 2018; Manhaeve et al, 2018]

Provide partial code

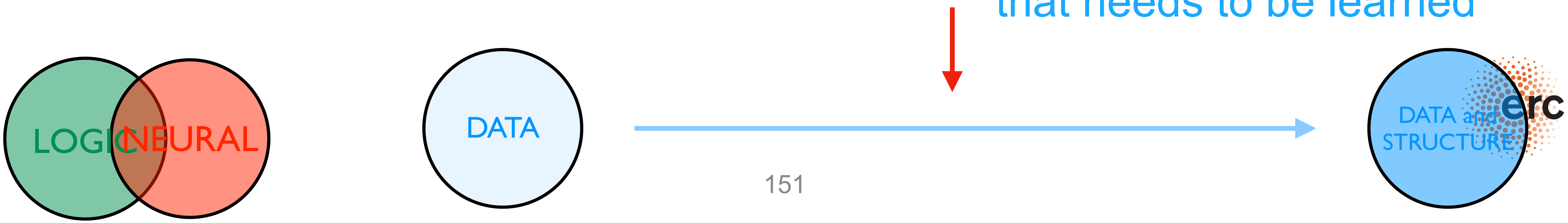
Fill in the missing functionality with neural networks

Examples:

[1,4,5] ↦ [1,16,25]
[2,2,5,1] ↦ [4,4,25,1]

```
def target_function(input_array):
    rarray = []

    for element in input_array:
        rarray.append(??(element))

    return rarray
```

partial functionality
that needs to be learned

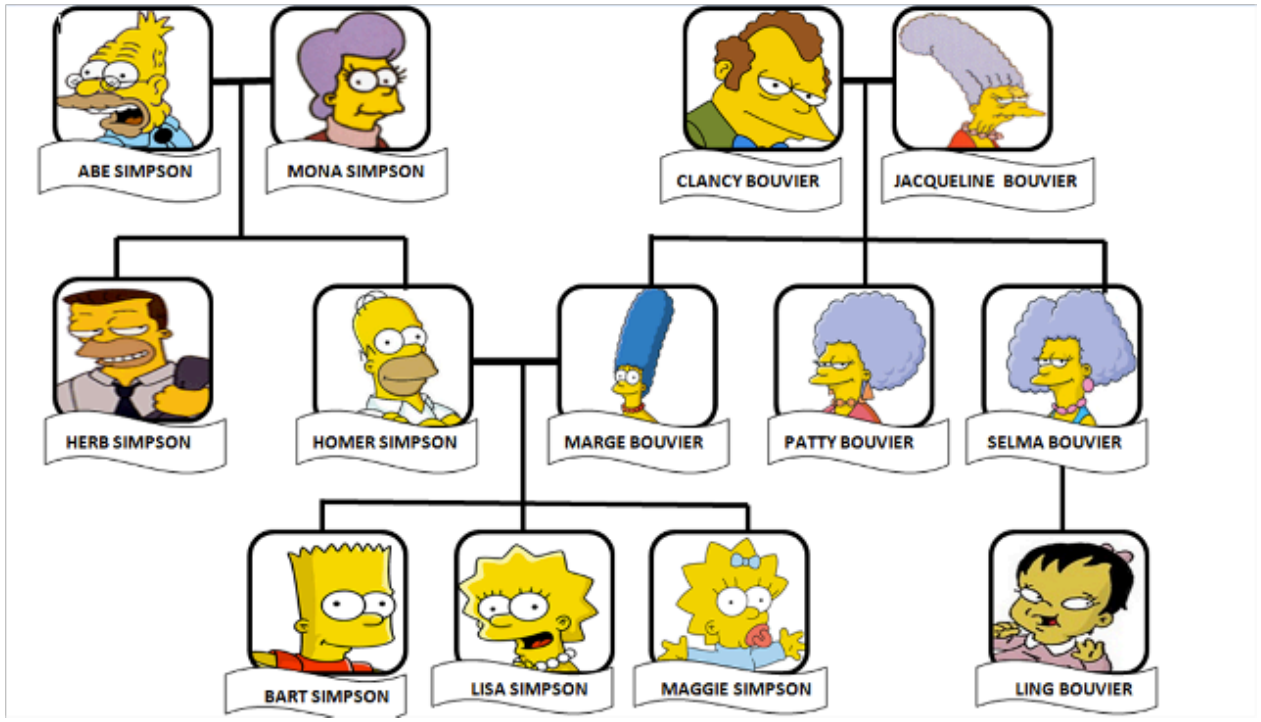


Structure learning via parameter learning

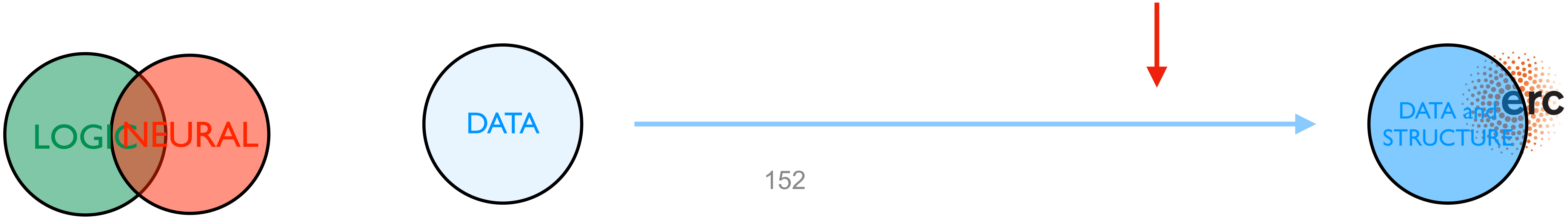
[Su et al, 2019]

Enumerate (lots of) logical formulas from templates

and learn their probabilities/weights



grandparent(abe,lisa).
grandparent(abe,bart).
grandparent(jacqueline,lisa).
grandparent(jacqueline,maggie.)



LOGIC

NEURAL

DATA

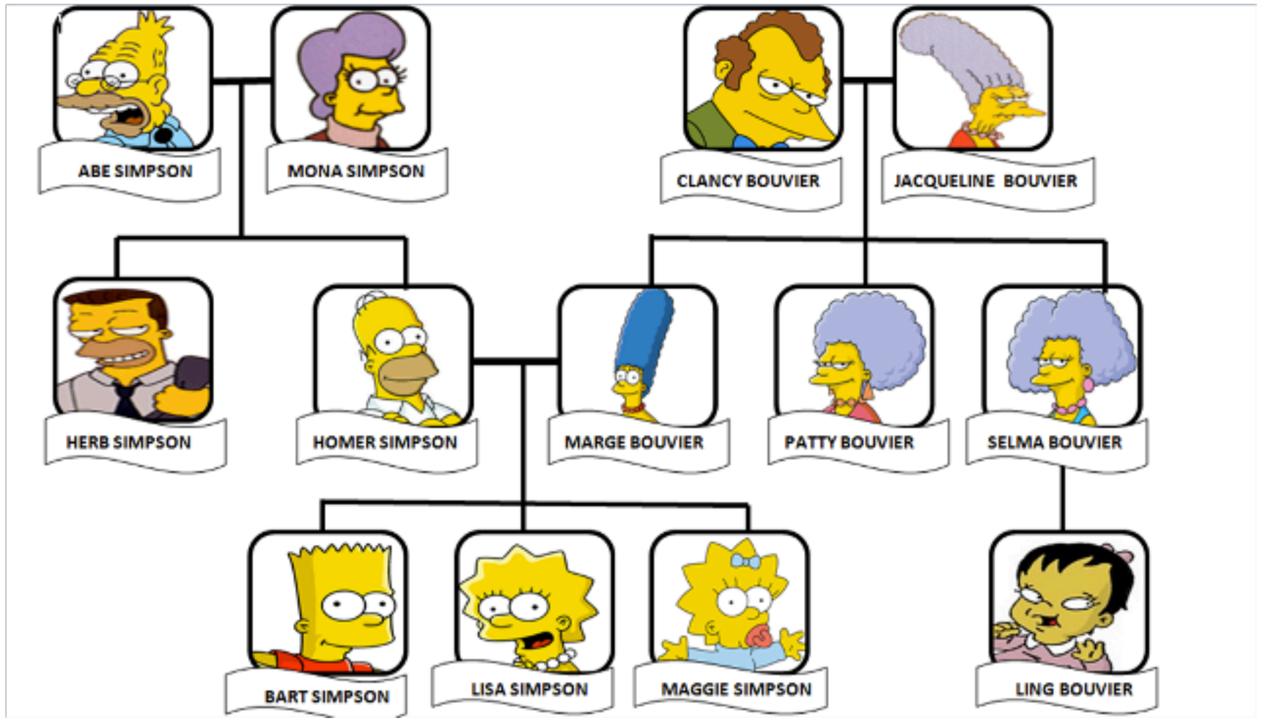
DATA and STRUCTURE

Program sketching

[Su et al, 2019]

Enumerate (lots of) logical formulas from templates

and learn their probabilities/weights



Program templates

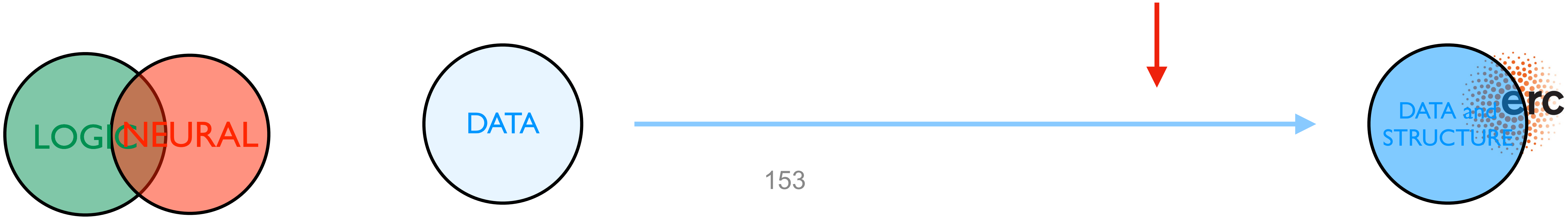
$$T(X,Y) \leftarrow P(X,Y).$$

$$T(X,Y) \leftarrow P(Y,X).$$

$$T(X,Y) \leftarrow P(X,Z), Q(Z,Y).$$

Target: grandparent

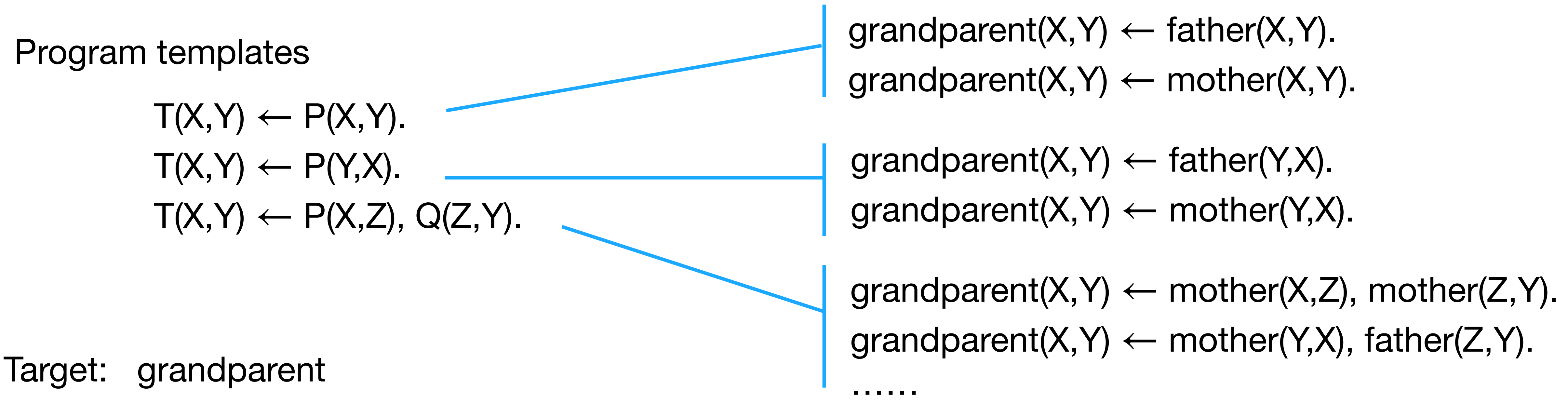
Other predicates: father, mother



Program sketching

[Su et al, 2019]

Enumerate (lots of) logical formulas from templates
and learn their probabilities/weights



Program templates

$T(X,Y) \leftarrow P(X,Y).$
$T(X,Y) \leftarrow P(Y,X).$
$T(X,Y) \leftarrow P(X,Z), Q(Z,Y).$

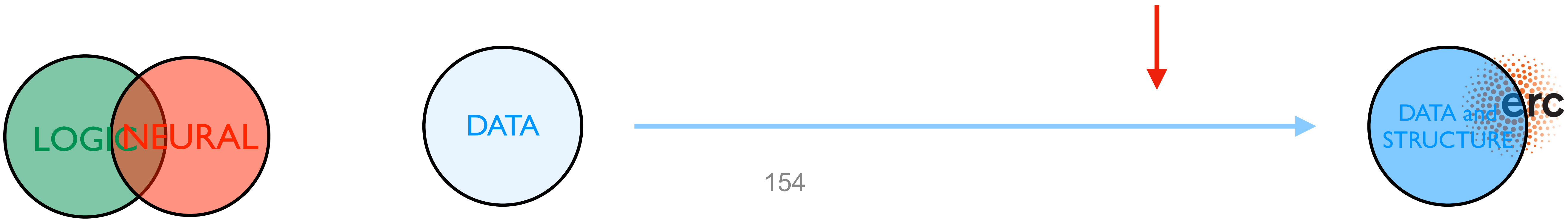
grandparent(X,Y) ← father(X,Y).
grandparent(X,Y) ← mother(X,Y).

grandparent(X,Y) ← father(Y,X).
grandparent(X,Y) ← mother(Y,X).

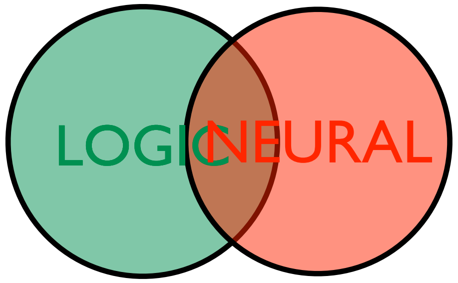
grandparent(X,Y) ← mother(X,Z), mother(Z,Y).
grandparent(X,Y) ← mother(Y,X), father(Z,Y).
……

Target: grandparent

Other predicates: father, mother



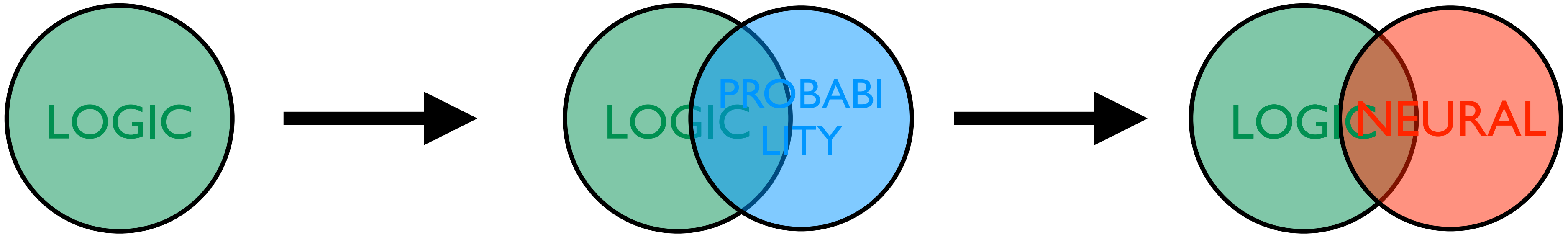
	Pros	Cons
Neural guidance	makes discrete search tractable	lots of training data
Soft patterns	efficient learning	no explicit structure
Neural generation	focused combinatorial search	lots of training data
Sketching	reduces combinatorial search	significant user effort
Structure via params	removes combinatorial search	spurious interactions



5. Learning
Key Messages

- Learning: finding logical formulas and estimating probabilities
- Structure learning: both formulas and probabilities
- Parameter learning: only probabilities
- Many flavours of learning in NeSy

6. Semantics



LOGIC

LOGIC PROBABILITY

LOGIC NEURAL

6. Semantics

Key Messages

- StarAI and NeSy share the same underlying semantics
- Semantics can be described in terms of parametric circuits
- Differentiable semantics/circuits allows an easy integration
- NeSy models can be seen as neural reparameterization of StarAI models

Semantics

- In Logic, semantics is connected to the interpretations of logical sentences
- An interpretation assigns a denotation or a value to each symbol in that language.

“*42(47)*”

Semantics

- In Logic, semantics is connected to the interpretations of logical sentences
- An interpretation assigns a denotation or a value to each symbol in that language.

“*human(socrates)*”

- Given a propositional language L, a labelling function is a function:

$$\ell : L \rightarrow V$$

6. Semantics

Boolean logic

LOGIC

Semantics in Boolean Logic

- Defining a semantics for a propositional language L is about assigning a truth value to all the sentences of the logic

- The labelling function ℓ_B is:

$$\ell_B : L \rightarrow \{True, False\}$$

Three steps:

1. Labels for propositions
2. Labels for operators
3. Labels for formulas

LOGIC

Semantics in Boolean Logic

1. Providing the labels for propositions

A,B,C.

$$\tilde{\ell}_B(A) = True$$
$$\tilde{\ell}_B(B) = False$$
$$\tilde{\ell}_B(C) = True$$

LOGIC

Semantics in Boolean Logic

2. Providing the semantics for operators

$\ell_B^{\rightarrow}$

p	q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

LOGIC

Semantics in Boolean Logic

3. The labels of formulas is defined recursively on the semantics of its components

$$\ell_B(A \to B) = \ell_B^{\to}(\tilde{\ell}_B(A), \tilde{\ell}_B(B))$$

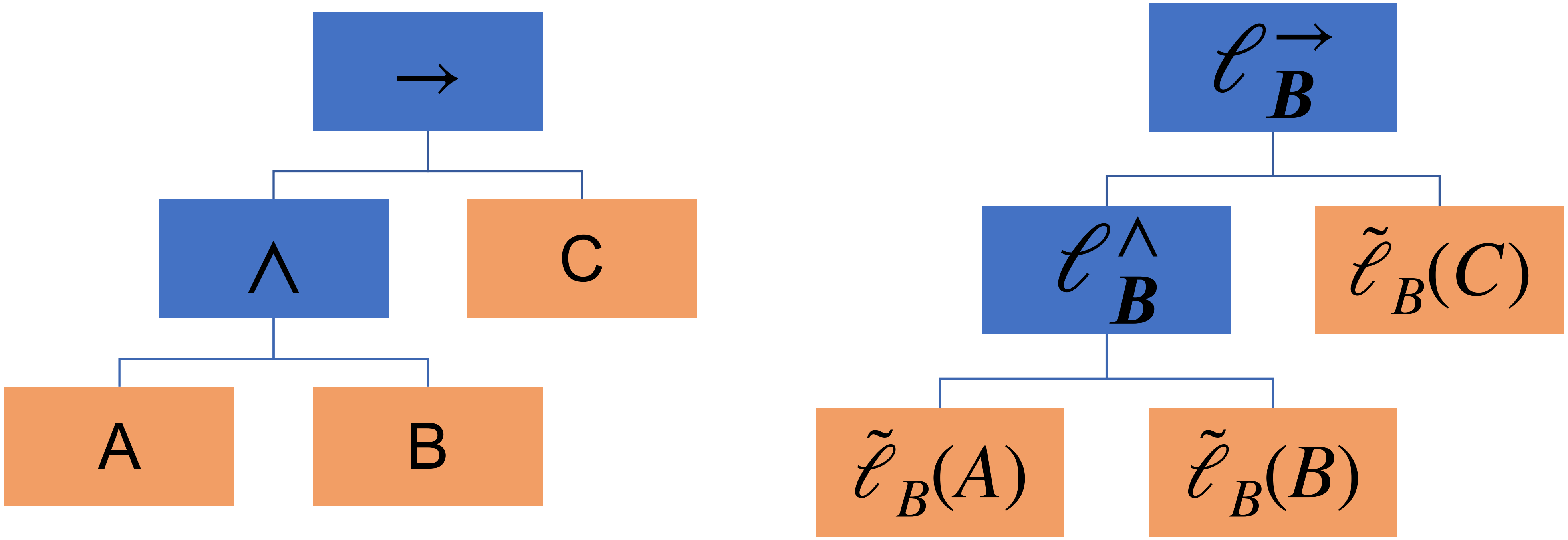
This recursive evaluation of formulas is said to be extensional approach.

LOGIC

Semantics in Boolean Logic

- Consider:

$(A \wedge B) \rightarrow C$



LOGIC

6. Semantics

Fuzzy logic

LOGIC

Fuzzy Logic Semantics

- There are many fuzzy logics
- Here we are interested in a subclass, in particular *t-norm fuzzy logic*

LOGIC

Fuzzy Logic Semantics

- Defining a semantics for a propositional fuzzy language L is again about assigning a truth degree to all the sentences of the logic

- We define a labeling function:

$$\ell_F : L \to [0,1]$$

Three steps:

1. Labels for propositions
2. Labels for operators
3. Labels for formulas

LOGIC

Fuzzy Logic Semantics

1. Providing the labels for propositions

A,B,C.

$$\tilde{\ell}_B(A) = 0.9$$
$$\tilde{\ell}_B(B) = 0.3$$
$$\tilde{\ell}_B(C) = 0.5$$

LOGIC

Fuzzy Logic Semantics

2. Providing the labels for operators: t-norm theory

- A t-norm is a binary function that extends the conjunction to the continuous case

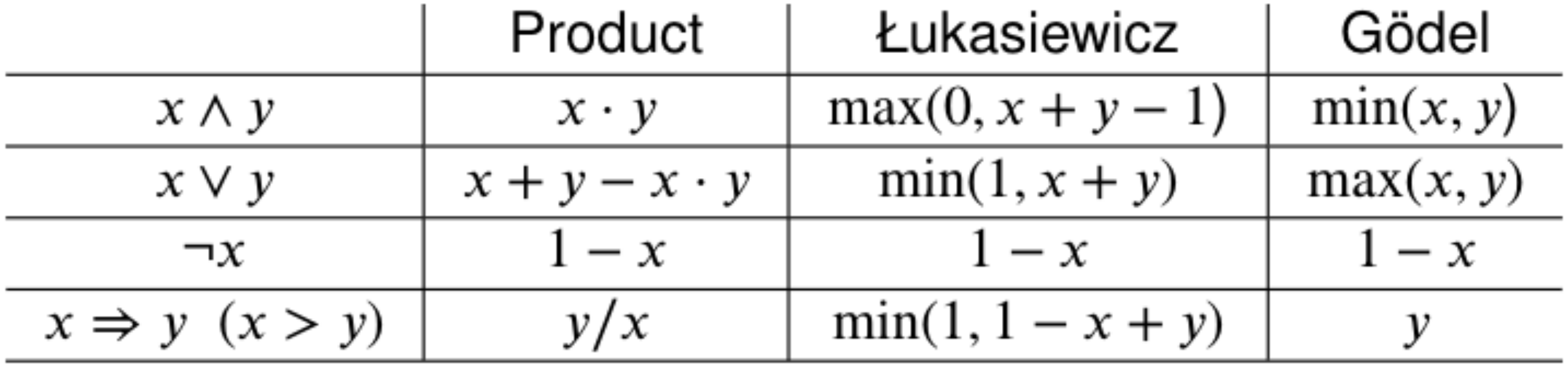
$$t : [0,1] \times [0,1] \rightarrow [0,1]$$

- There are 3 fundamental t-norms:
  - Lukasiewicz t-norm: $t_L(x, y) = \max(0, x + y - 1)$
  - Goedel t-norm: $t_G(x, y) = \min(x, y)$
  - Product t-norm: $t_P(x, y) = x \cdot y$

LOGIC

Fuzzy Logic Semantics

- All the other operators can be derived from the t-norm (and its residuum)



	Product	Łukasiewicz	Gödel
$x \wedge y$	$x \cdot y$	$\max(0, x + y - 1)$	$\min(x, y)$
$x \vee y$	$x + y - x \cdot y$	$\min(1, x + y)$	$\max(x, y)$
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \Rightarrow y \;\; (x > y)$	y/x	$\min(1, 1 - x + y)$	y

LOGIC

Fuzzy Logic Semantics

3. The labels of formulas is defined recursively on the semantics of its components

$$\ell_F(A \to B) = \ell_F^{\to}(\tilde{\ell}_F(A), \tilde{\ell}_F(B))$$

This recursive evaluation of formulas is said to be extensional approach.

e.g.

$\tilde{\ell}_F(A) = 0.9$, $\tilde{\ell}_F(B) = 0.3$, $\ell_F^{\to} = \min(1, 1 - x + y)$

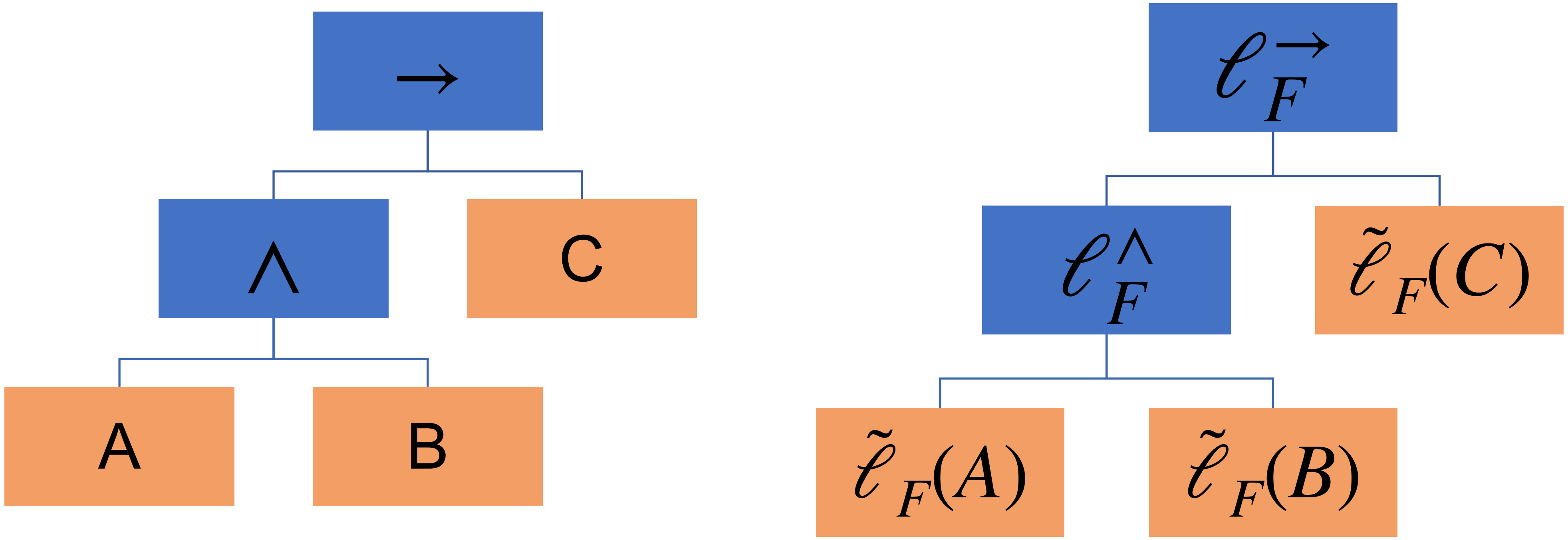
LOGIC

$$\ell_F(A \to B) = \min(1, 1 - 0.9 + 0{,}3) = 0.4$$

Fuzzy Logic Semantics

- Consider:

$(A \wedge B) \to C$



LOGIC

Fuzzy Logic Semantics

Properties of t-norms

- Most common t-norms are:
  - **Continuous**
  - **Differentiable** -> This turns to be one of the reason of their adoption in NeSY

- Convex fragments of the logic can be defined (Giannini et al, 2019)

LOGIC

Fuzzy vs Boolean

- Fuzzy and Boolean have different properties
- When fuzzy is used as a “relaxation” (**fuzzification)** of Boolean **undesired effects** can happen.

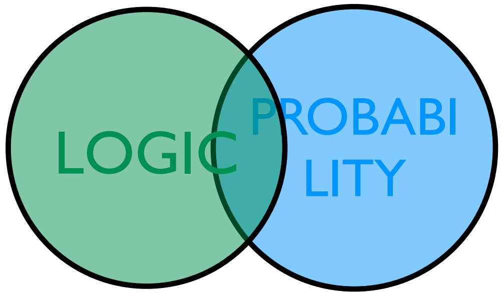
- Consider the rule:

1. $\ell_B(A \vee B \vee C) = True$
2. $\ell_F(A \vee B \vee C) = \min(1, \ell_F(A) + \ell_F(B) + \ell_F(C)) = 1$
3. $\ell_F(A) = \ell_F(B) = \ell_F(C) = 0.35$
4. $\ell_B(A) = \ell_B(B) = \ell_B(C) = False$

LOGIC

Semantics

Probabilistic logic



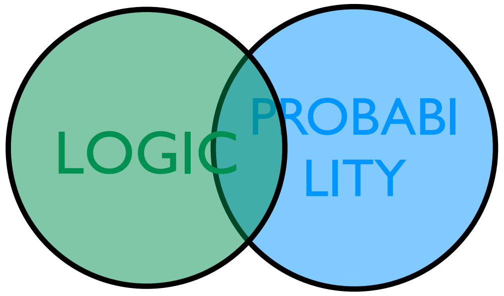
Probabilistic Logic Semantics

Given a proposition language L, the basic idea is to introduce a probability function ℓ_P:

$$\ell_P\colon L \rightarrow [0,1]$$

Three steps:

1. Labels for propositions / formulas
2. Distribution over possible interpretations
3. Labels for formulas = Weighted Model Count using distribution



Probabilistic Logic Semantics

1. Provide
   A. the labels for propositions (e.g. ProbLog)

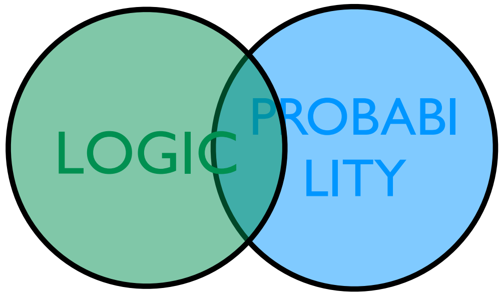
A, B

$$\tilde{\ell}_P(A) = 0.1$$
$$\tilde{\ell}_P(B) = 0.7$$

   B. the labels for formulas of interest (e.g. Markov Logic)

$$A \land B \qquad \tilde{\ell}_P(A \land B) = 1.5$$

$$(\neq \ell_P(A \land B))$$

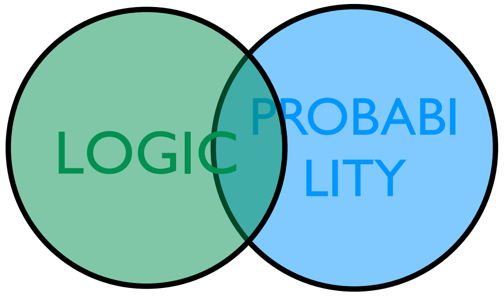


Probabilistic Logic Semantics

2. Usually ℓ_P is defined in terms of a probabilistic distribution p over truth assignments or interpretations of the propositional variables.

$$p(\ell_B(x_1), \ldots, \ell_B(x_n))$$

e.g. p(A=True, B=False) = ?



Probabilistic Logic Semantics

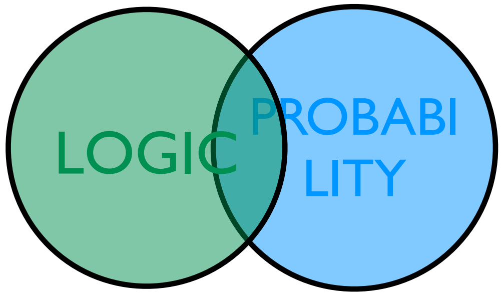
e.g. in ProbLog:

$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \prod_{i:\ell_B(x_i)=True} \tilde{\ell}_P(x_i) \prod_{i:\ell_B(x_i)=False} (1 - \tilde{\ell}_P(x_i))$$

0.1 :: burglary. (B)
0.05 ::earthquake. (E)
0.6 ::hears_alarm(john). (H)
alarm :– earthquake.
alarm :– burglary.

B	E	H	p(B,E,H)
F	F	F	0.342
F	F	T	0.513
F	T	F	0.018
F	T	T	0.027
T	F	F	0.038
T	F	T	0.057
T	T	F	0.002
T	T	T	0.003

0.1 x 0.05 x (1-0.6)



Probabilistic Logic Semantics

e.g. Markov Logic

$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} \tilde{\ell}_P(\alpha) \sum_{x} \ell_B(\alpha(x)) \right)$$

Weight formula

Number of true groundings
(Each true grounding
contributes with 1)

stress(X) -> smokes(X): **1.5**

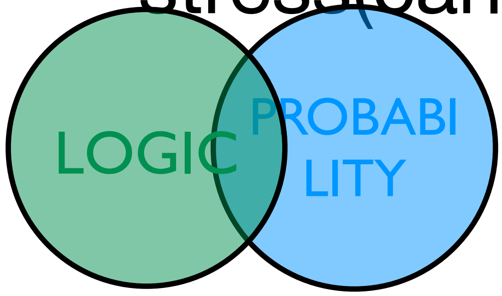
stress(ann) -> smokes(ann): **True (1)**

stress(bob) -> smokes(bob): **False (0)**

stress(carl) -> smokes(carl): **True (1)**

$$\propto exp(1.5 \cdot (1 + 0 + 1))$$

B	E	H	p(B,E,H)
F	F	F	0.342
F	F	T	0.513
F	T	F	0.018
F	T	T	0.027
T	F	F	0.038
T	F	T	0.057
T	T	F	0.002
T	T	T	0.003



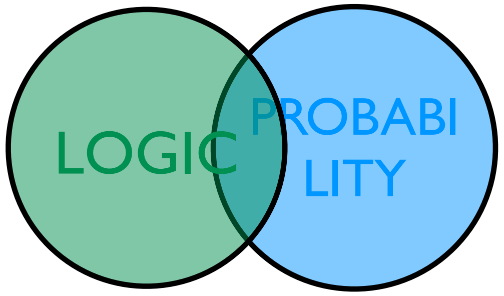
Probabilistic Logic Semantics

3. Given any sentence Q of the propositional language L, with variables $x_1, \ldots, x_n$:

$$\ell_P(Q) = \sum_{\ell_B(x_1),\ldots,\ell_B(x_n) \vDash Q} p(\ell_B(x_1), \ldots, \ell_B(x_n))$$

WMC - Weighted Model Counting

(for both ProbLog and Markov Logic)



Probabilistic Logic Semantics

For example:

B	E	H	p(B,E,H)
F	F	F	0.342
F	F	T	0.513
F	T	F	0.018
F	T	T	0.027
T	F	F	0.038
T	F	T	0.057
T	T	F	0.002
T	T	T	0.003

```
0.1 :: burglary.   (B)
0.05 ::earthquake. (E)
0.6 ::hears_alarm(john).  (H)
alarm :– earthquake.
alarm :– burglary.
```

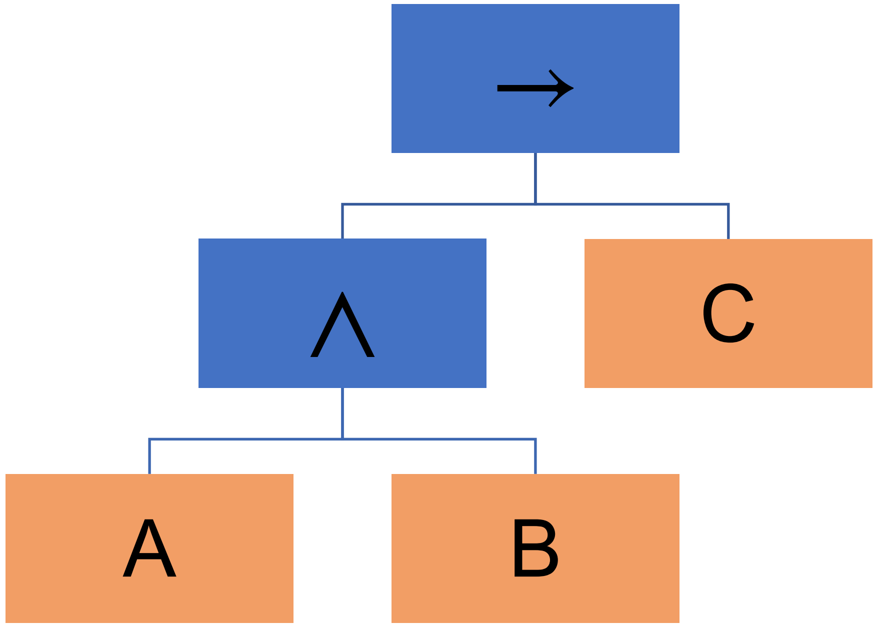
$$Q = B \wedge H$$

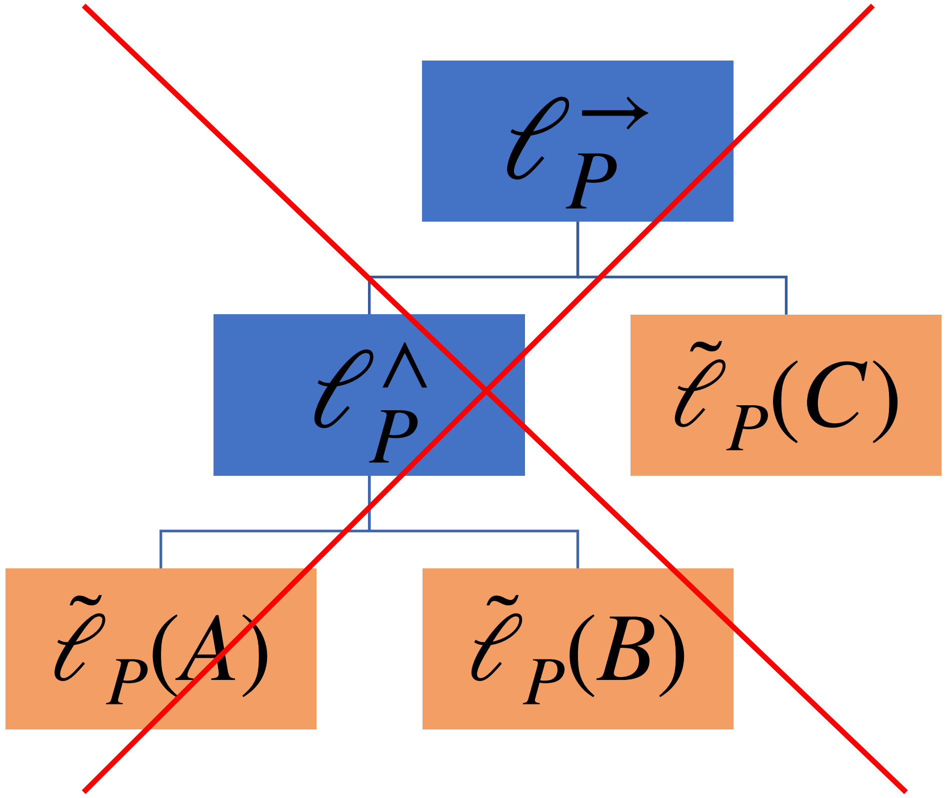
$$\ell_P(Q) = 0.06$$

Probabilistic Logic Semantics

- Consider:

$(A \wedge B) \to C$



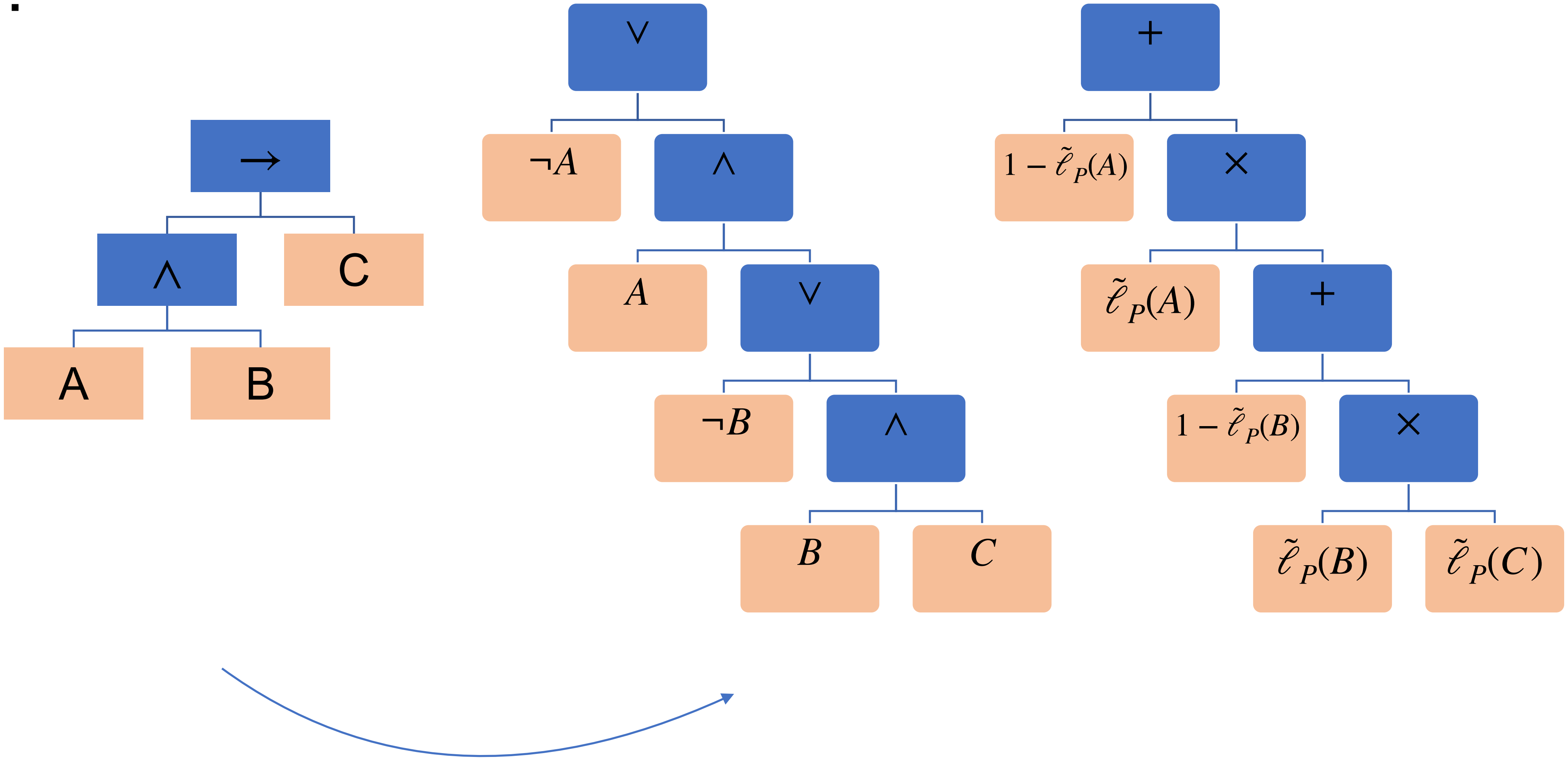


(not always at least)

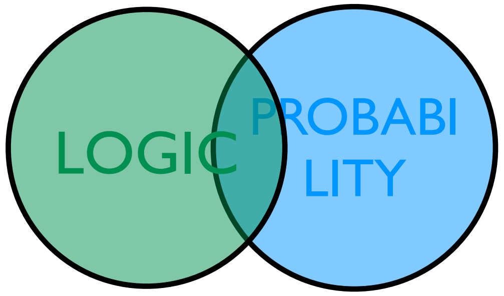
Probabilistic Logic Semantics

- Consider:

$(A \wedge B) \rightarrow C$



Knowledge Compilation



The probabilistic structure is now explicit in the compiled formula.

Probabilistic Soft Logic (PSL)

Bach, Stephen H., et al. *JMLR* 2017

- Let's start by an example of a Markov Logic Network:

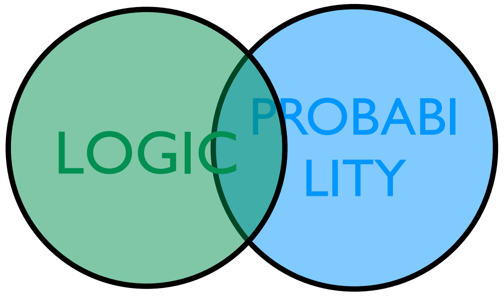
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} \tilde{\ell}_P(\alpha) \sum_{x} \ell_B(\alpha(x)) \right)$$

- In PSL, we relax the Boolean semantics ℓ_B to a fuzzy semantics ℓ_F

$$p(\ell_F(x_1), \ldots, \ell_F(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} \tilde{\ell}_P(\alpha) \sum_{x} \ell_F(\alpha(x)) \right)$$

Weight formula

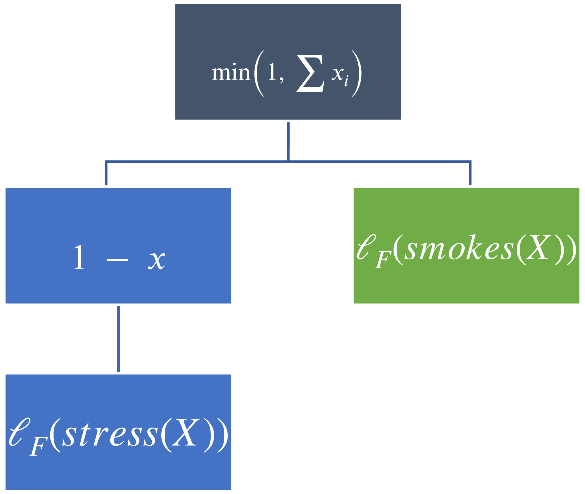
Each grounding contributes with a value in [0,1]



Probabilistic Soft Logic (PSL)

$$\alpha(X) : stress(X) \rightarrow smokes(X)$$

$$\ell_F(\alpha(X)) = \min(1,1-\ell_F(stress(X)) + \ell_F(smokes(X)))$$



$\min\left(1, \sum x_i\right)$

$1 - x$

$\ell_F(smokes(X))$

$\ell_F(stress(X))$

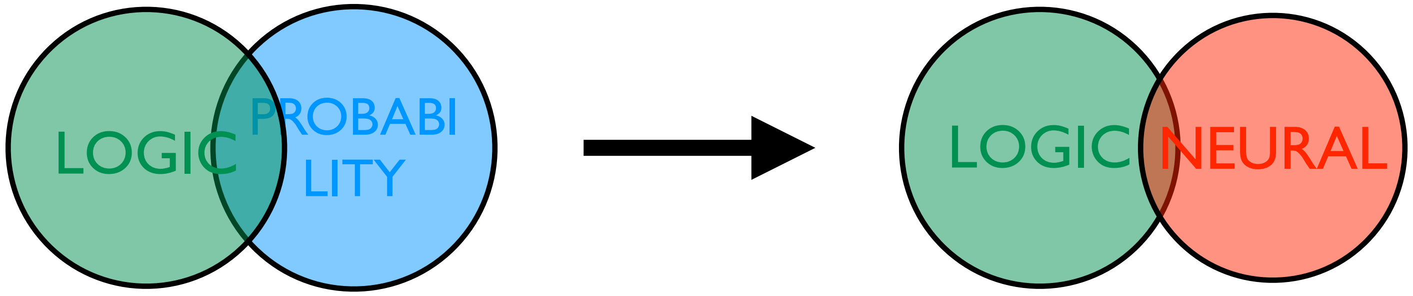
MPE:

$$\max_{\ell_F(stress(X)),\ell_F(smokes(X))} \ell_P(\alpha) \sum_X \ell_F(\alpha(X))$$

$$\ell_F(stress(X)) = \ell_F(stress(X)) + \lambda \frac{\partial \ell_P(\alpha) \sum_X \ell_f(X)}{\partial \ell_F(stress(X))}$$

6. Semantics

Neural Symbolic



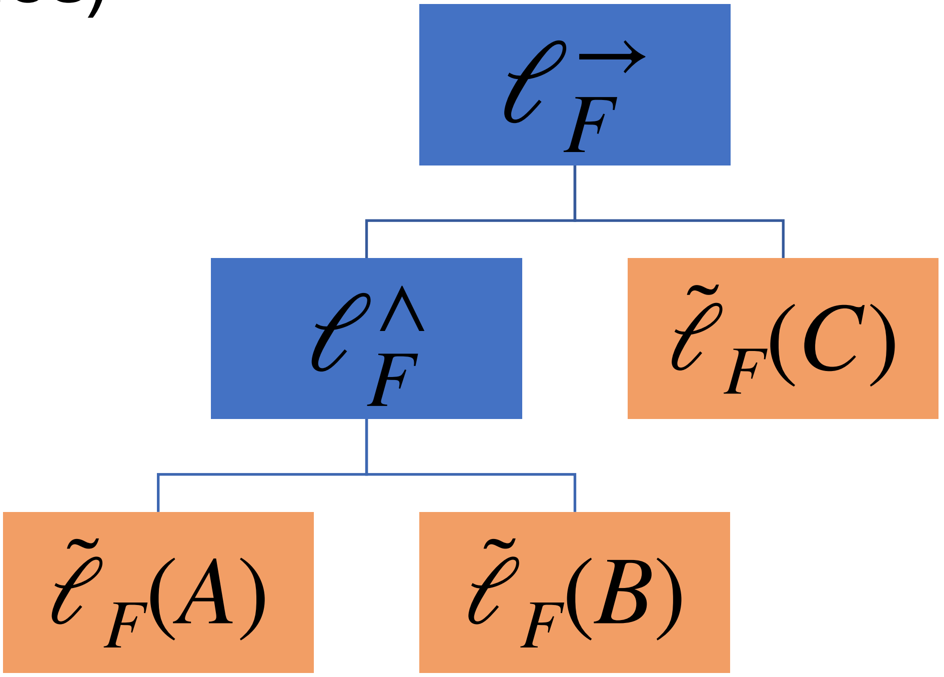
Neural Symbolic

How to carry over concepts from the semantics of StarAI to neural symbolic?

$$\ell(Q)$$

Labelling functions (semantics) = Parametric circuit

$\ell_F((A \land B) \rightarrow C)$



The query Q determine the structure (potentially after knowledge compilation)

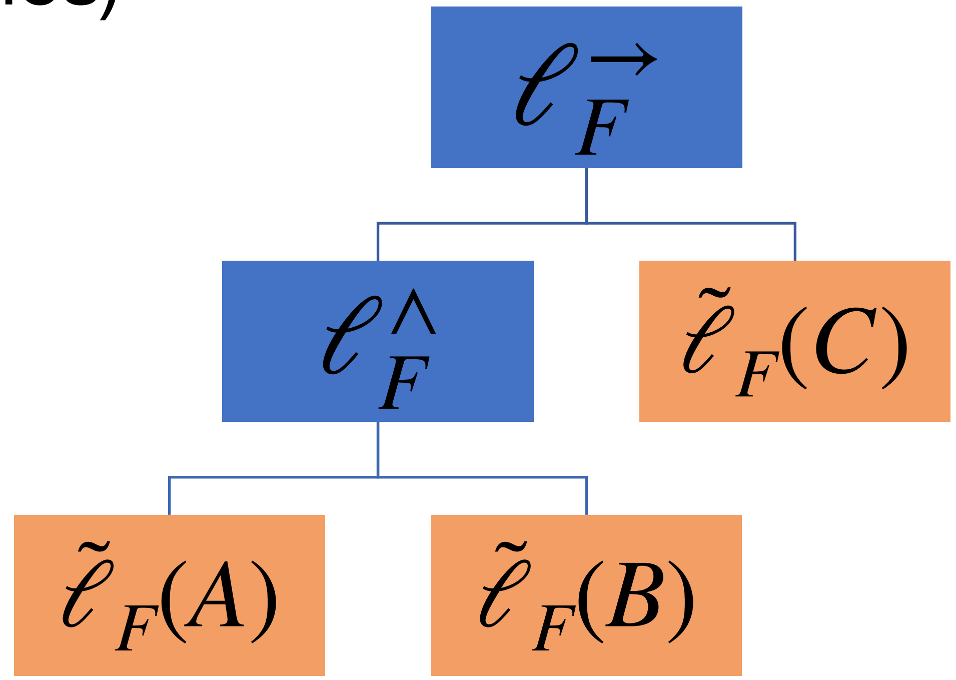
Neural Symbolic

How to carry over concepts from the semantics of StarAI to neural symbolic?

$$\ell(Q)$$

Labelling functions (semantics) = Parametric circuit

$\ell_F((A \wedge B) \rightarrow C)$



$\ell_F^{\rightarrow}$

$\ell_F^{\wedge}$

$\tilde{\ell}_F(C)$

$\tilde{\ell}_F(A)$

$\tilde{\ell}_F(B)$

The leaves represent the scalar parameters $\tilde{\ell}$

Neural Symbolic

How to carry over concepts from the semantics of StarAI to neural symbolic?

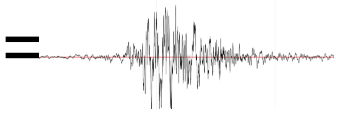
- Atomic labels $\tilde{\ell}$ are just scalar tables of parameters

```
0.1 :: burglary.   (B)
0.05 ::earthquake. (E)
0.6 ::hears_alarm(john).  (H)
alarm :– earthquake.
alarm :– burglary.
```

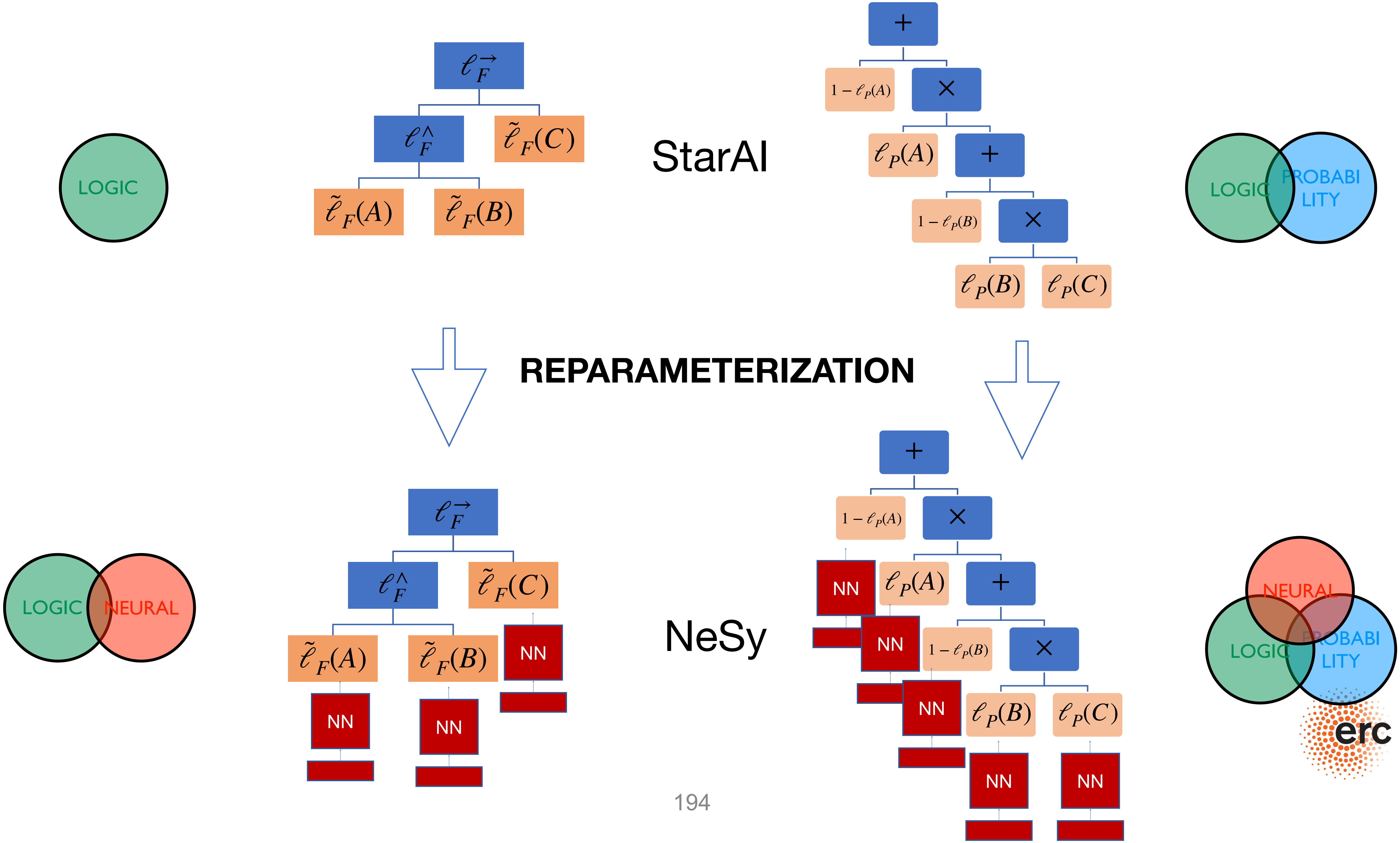
L	$\tilde{\ell}$
Burglary	0.1
Earthquake	0.05
…	

Neural Symbolic

- What if we turn scalar parameters $\tilde{\ell}$ to neural networks?

- Two main reasons:
    - **Perceptive queries** (burglary =  , earthquake= )
    - **Semantic sub-symbolic queries** (burglary=[0.33,0.56,7.45])

StarAI to Neural Symbolic

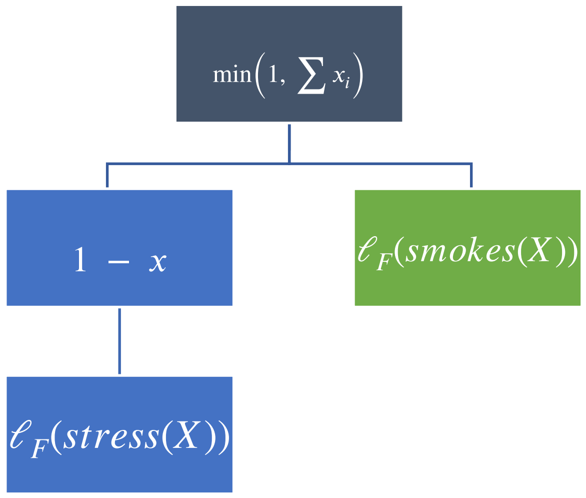


StarAI

REPARAMETERIZATION

NeSy

Fuzzy Reparameterization

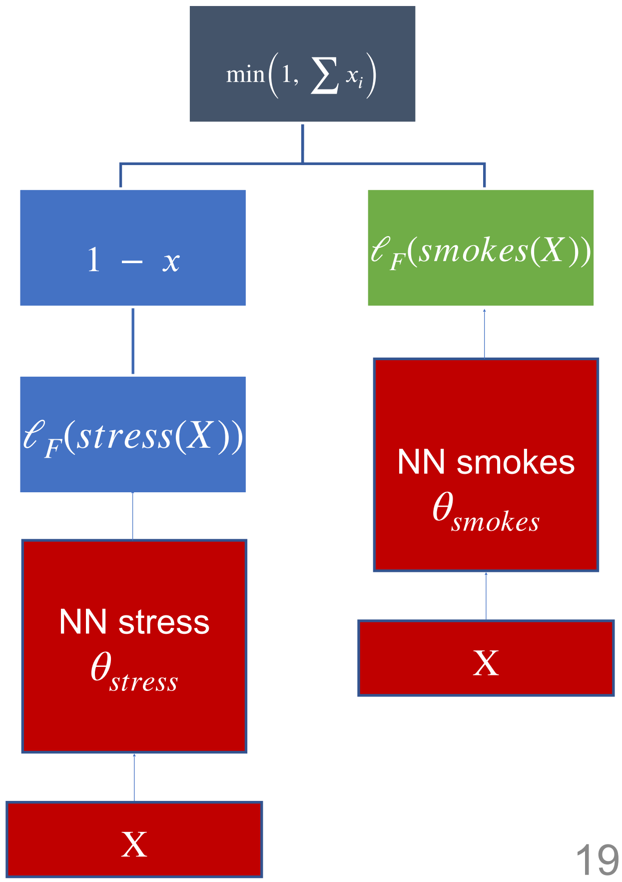


StarAI (PSL)

$$\max_{\ell_F(stress(X)),\ell_F(smokes(X))} \sum_{\alpha} \ell_P(\alpha)\ell_F(\alpha(X))$$

Semantic Based Regularization (Diligenti et al, AI 2017)

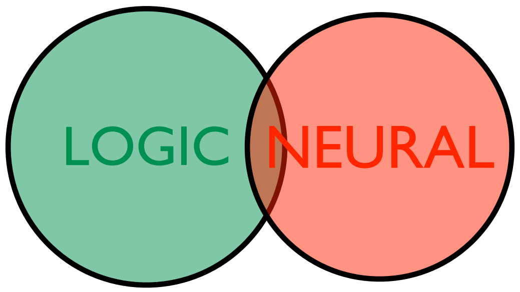
Logic Tensor Network (Donadello et at, IJCAI 2017)



NeSy (SBR, LTN)

$$\max_{\theta_{stress},\theta_{smokes}} \sum_{\alpha} \ell_P(\alpha)\ell_F(\alpha(X))$$

Parameters of the neural nets



Probabilistic Reparameterization

🟧 Probabilistic parameters

- ProbLog:

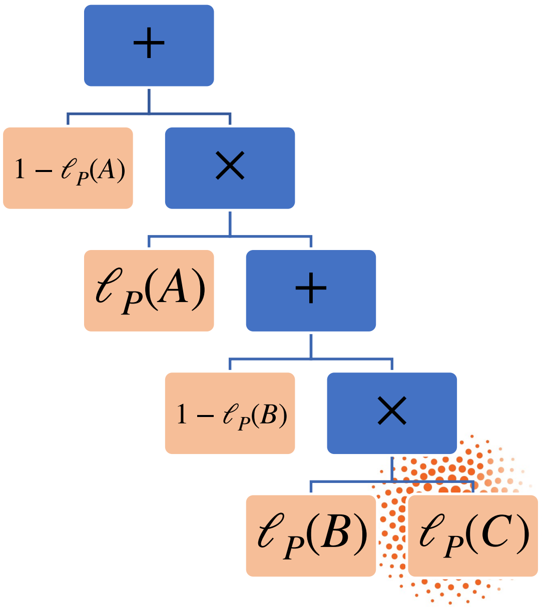
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \prod_{i:\ell_B(x_i)=True} \tilde{\ell}_P(x_i) \prod_{i:\ell_B(x_i)=False} (1-\tilde{\ell}_P(x_i))$$

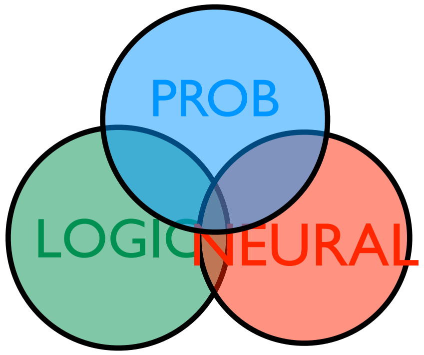
- Markov Logic:

$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} \tilde{\ell}_P(\alpha) \sum_{x} \ell_B(\alpha(x)) \right)$$

WMC

$$\ell_P(Q) = \sum_{\ell_B(x_1),\ldots,\ell_B(x_n) \models Q} p(\ell_B(x_1), \ldots, \ell_B(x_n))$$





Probabilistic Reparameterization

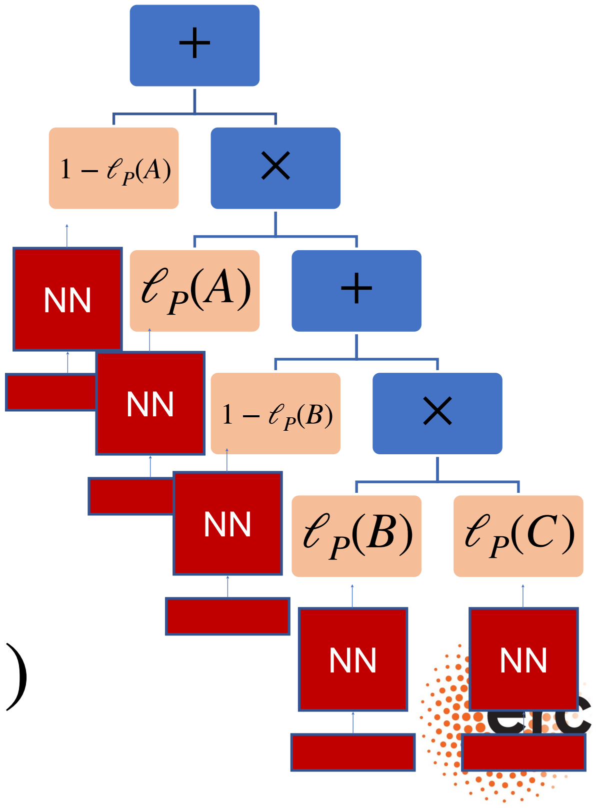
🟥 Neural parameters

- **DeepProbLog** (Manhaeve et al, NeurIPS (2018))

$$p(\ell_B(x_1), \dots, \ell_B(x_n)) = \prod_{i:\ell_B(x_i)=True} \tilde{\ell}_P(x_i) \prod_{i:\ell_B(x_i)=False} (1-\tilde{\ell}_P(x_i))$$

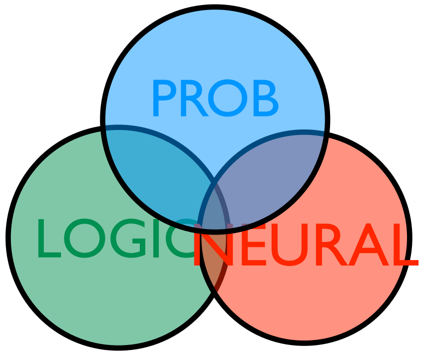
- **Relational Neural Machines** (Marra et al, ECAI 2020)

$$p(\ell_B(x_1), \dots, \ell_B(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} \tilde{\ell}_P(\alpha) \sum_{x} \ell_B(\alpha(x)) \right)$$



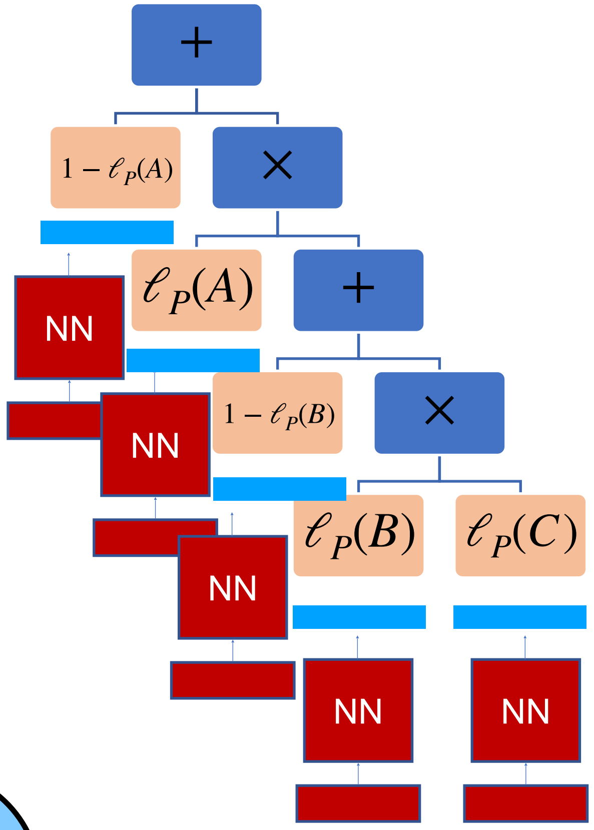
WMC

$$\ell_P(Q) = \sum_{\ell_B(x_1),\dots,\ell_B(x_n) \models Q} p(\ell_B(x_1), \dots, \ell_B(x_n))$$



Probabilistic Reparameterization

- **DeepProbLog** (Manhaeve et al, NeurIPS (2018))



Interface ←

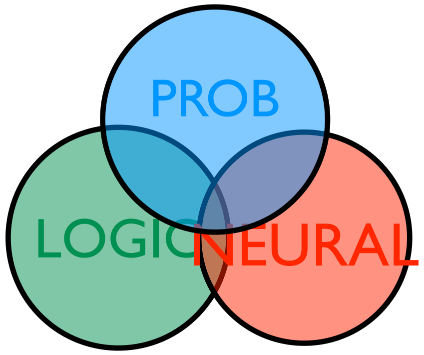
Probabilistic fact

0.01 :: burglary.

↓

Neural Predicate

nn(mnist_net, [X], Y, [0 ... 9]) :: digit(X,Y).



6. Semantics

Key Messages

- StarAI and NeSy share the same underlying semantics
- Semantics can be described in terms of parametric circuits
- Differentiable semantics/circuits allow an easy integration
- NeSy models can be seen as neural reparameterization of StarAI models

7. Logic vs Probability vs Neural

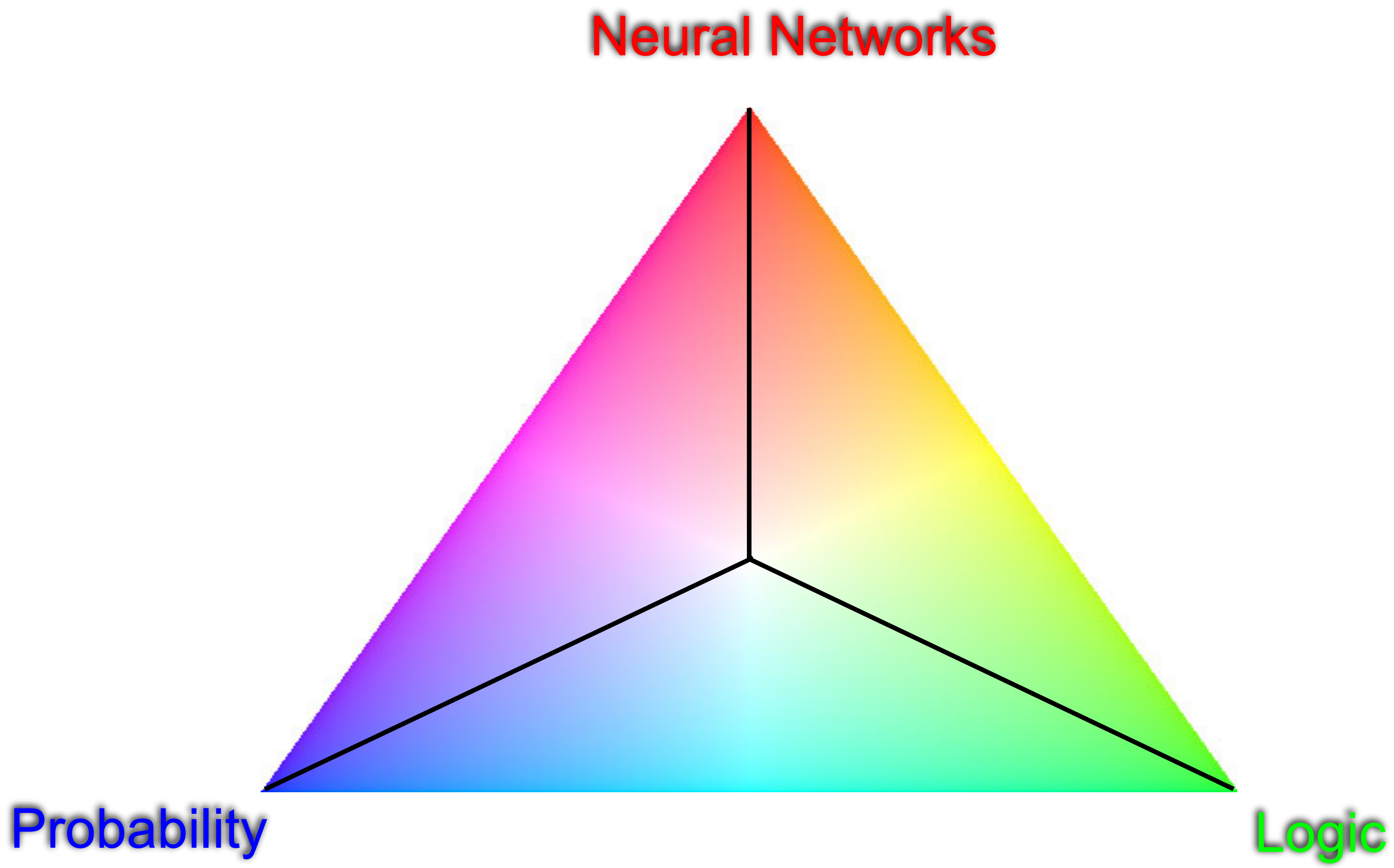
7. Logic vs Probability vs Neural
Key Messages

- We have three paradigms in the NeSy spectrum: Logic, Probability and Neural Networks

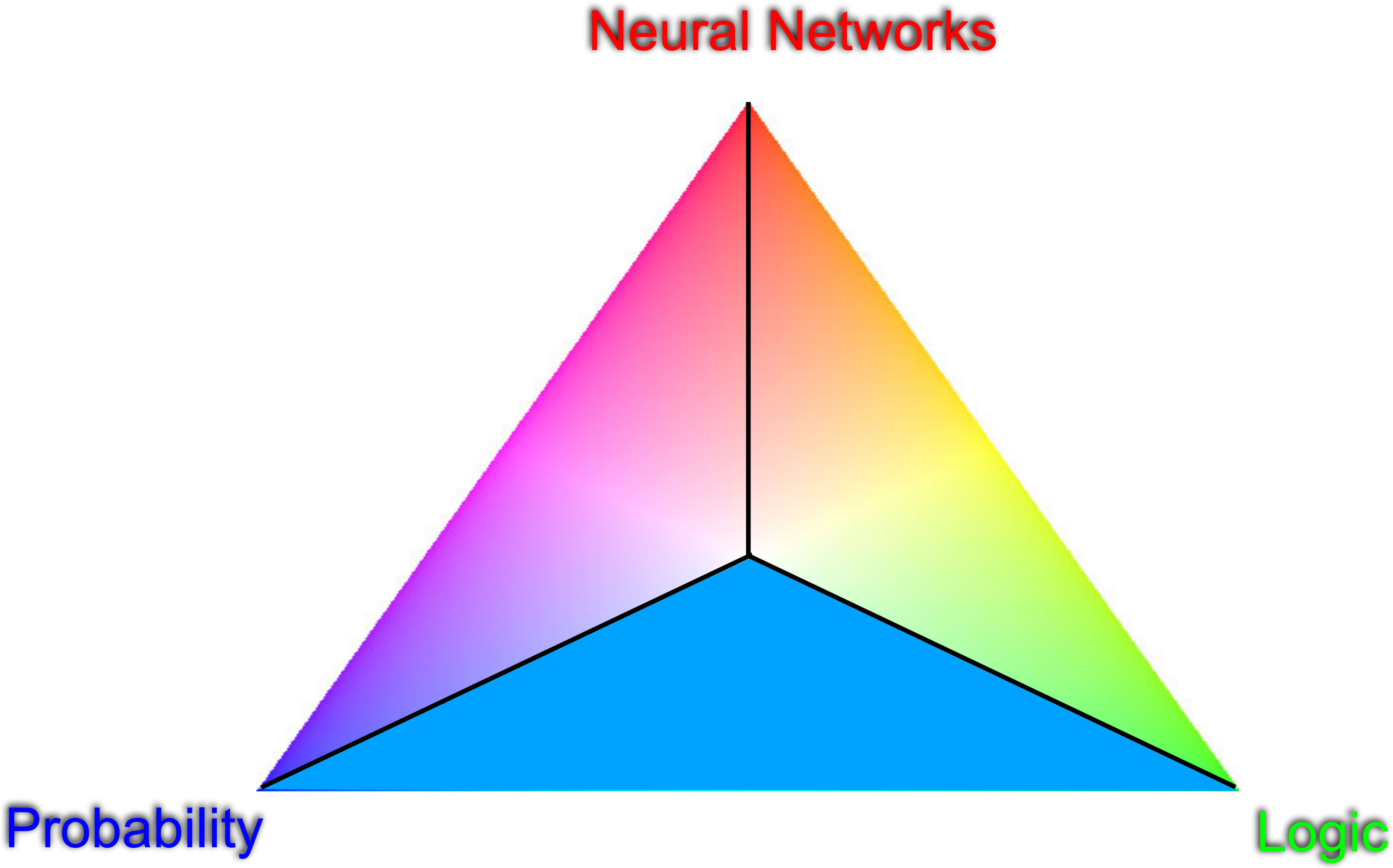
- An integration of the three should have the original paradigms as special cases
  - Computationally complex

- The integration is usually achieved by sacrificing the base paradigms
  - More scalable

About integration in neural symbolic

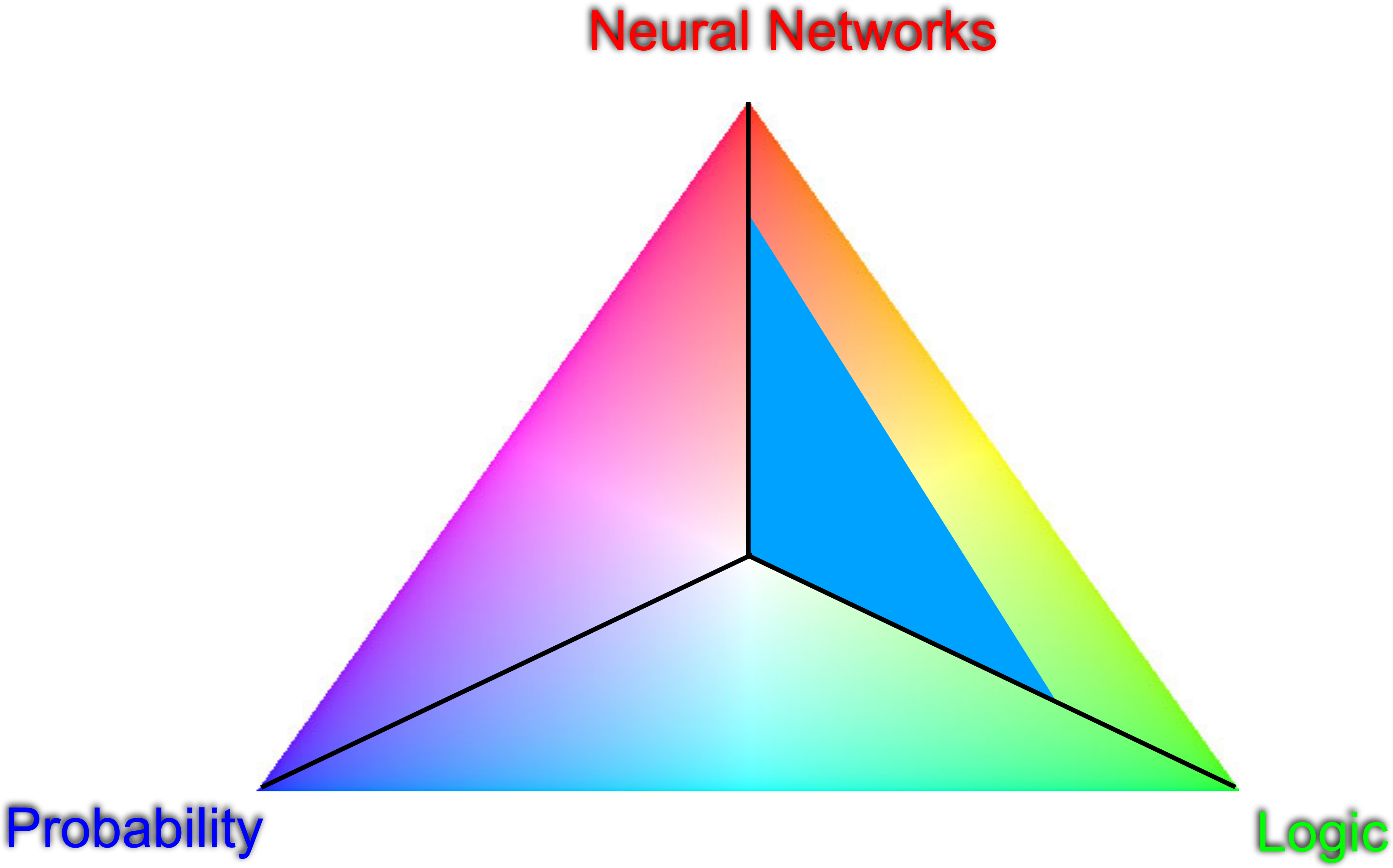


Statistical Relational AI



They perfectly integrate probability theory (Probabilistic Graphical Models) and Logic.

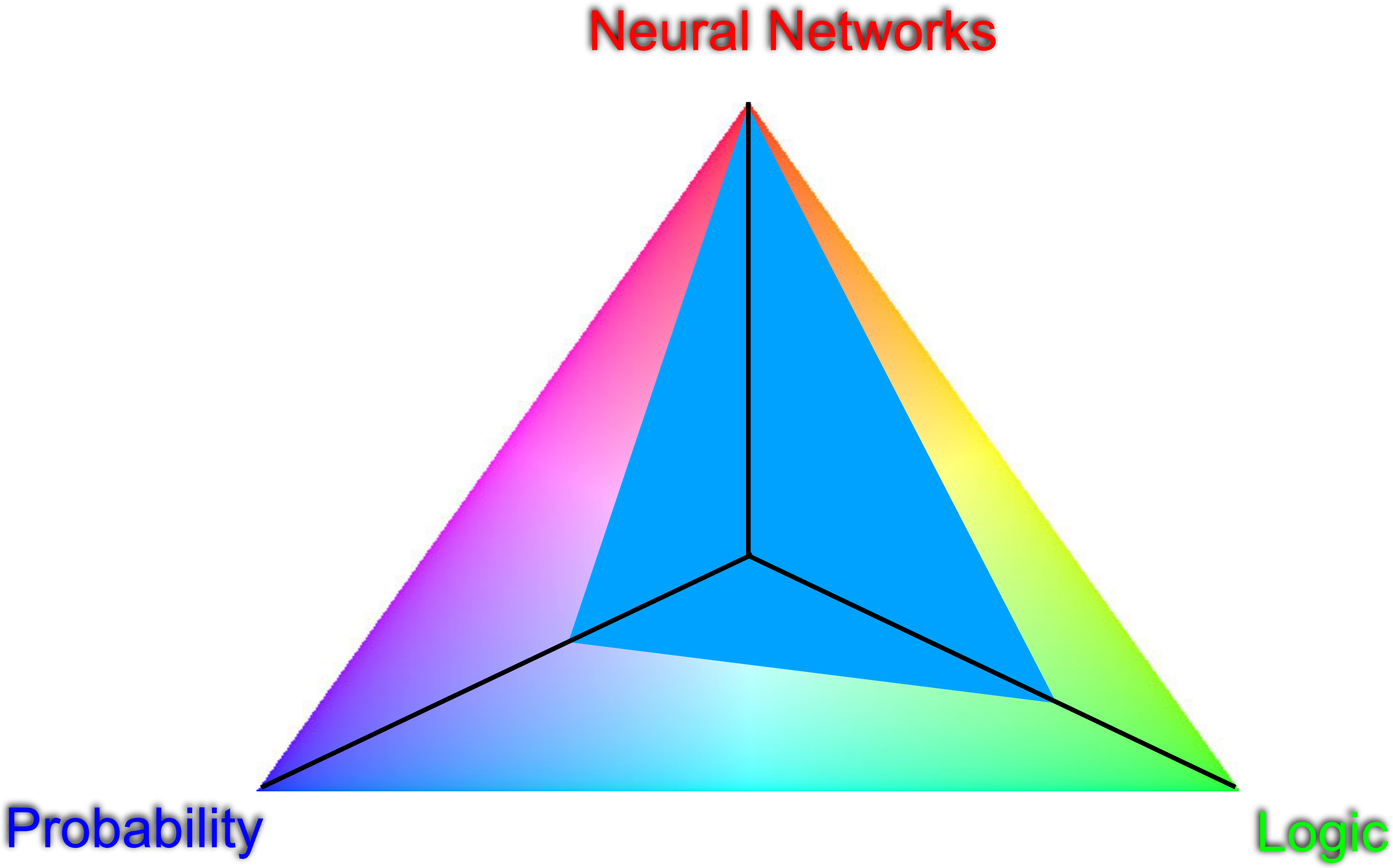
Relaxed theorem provers



They sacrifice a bit the pure boolean semantics to obtain some soft neural capabilities (weighted reasoning, embeddings).

KBANN (Tawell 1994)
LRNN (Sourek, 2017)
NTPs (Rocktäschel, 2017)
DiffLog (Si et al, 2018)
NN for Relational Data (2019)

Regularization methods

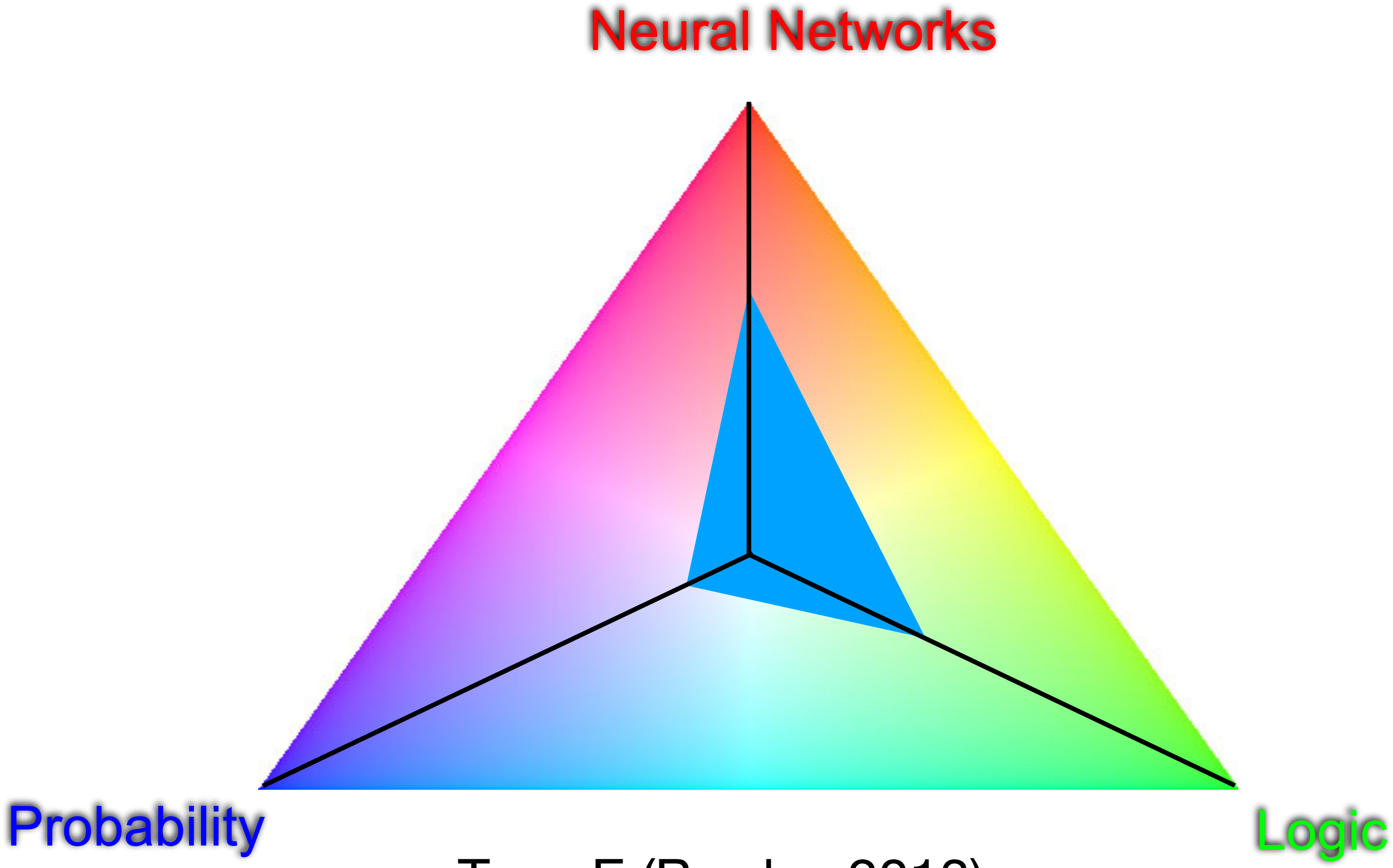


They sacrifice the logic and probability a lot by pushing everything inside the weights of the neural network.

Logic and probability are used only at training time. At inference time, only the neural net is used.

SBR (Diligenti et al, AI 2017)
LTN (Donatello et al, IJCAI 2017)
SL (Xu et al, ICML 2018)

Knowledge Graph Embeddings



TransE (Bordes 2013)
DistMult (Yang, 2014)
ComplEx (Trouillon, 2016)
NTN (Socher, 2013)

They use latent spaces, typical of neural computation to encode a relational structure of the data.

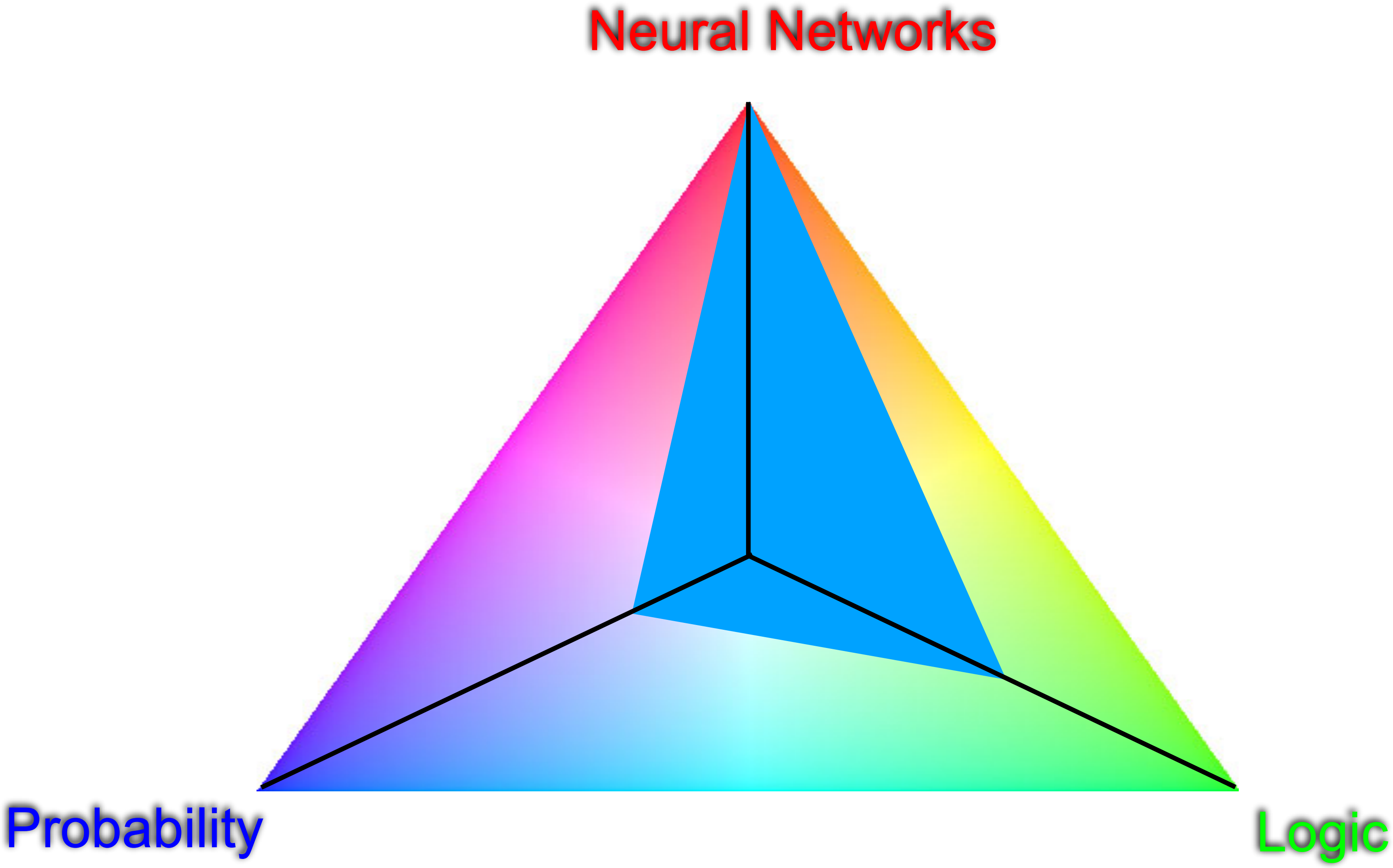
Neural networks cannot be recovered.

Logic is declined to encoding relations

Probabilistic modelling is strongly approximated (e.g. atom mean field)

Most scalable solutions.

Graph Neural Networks

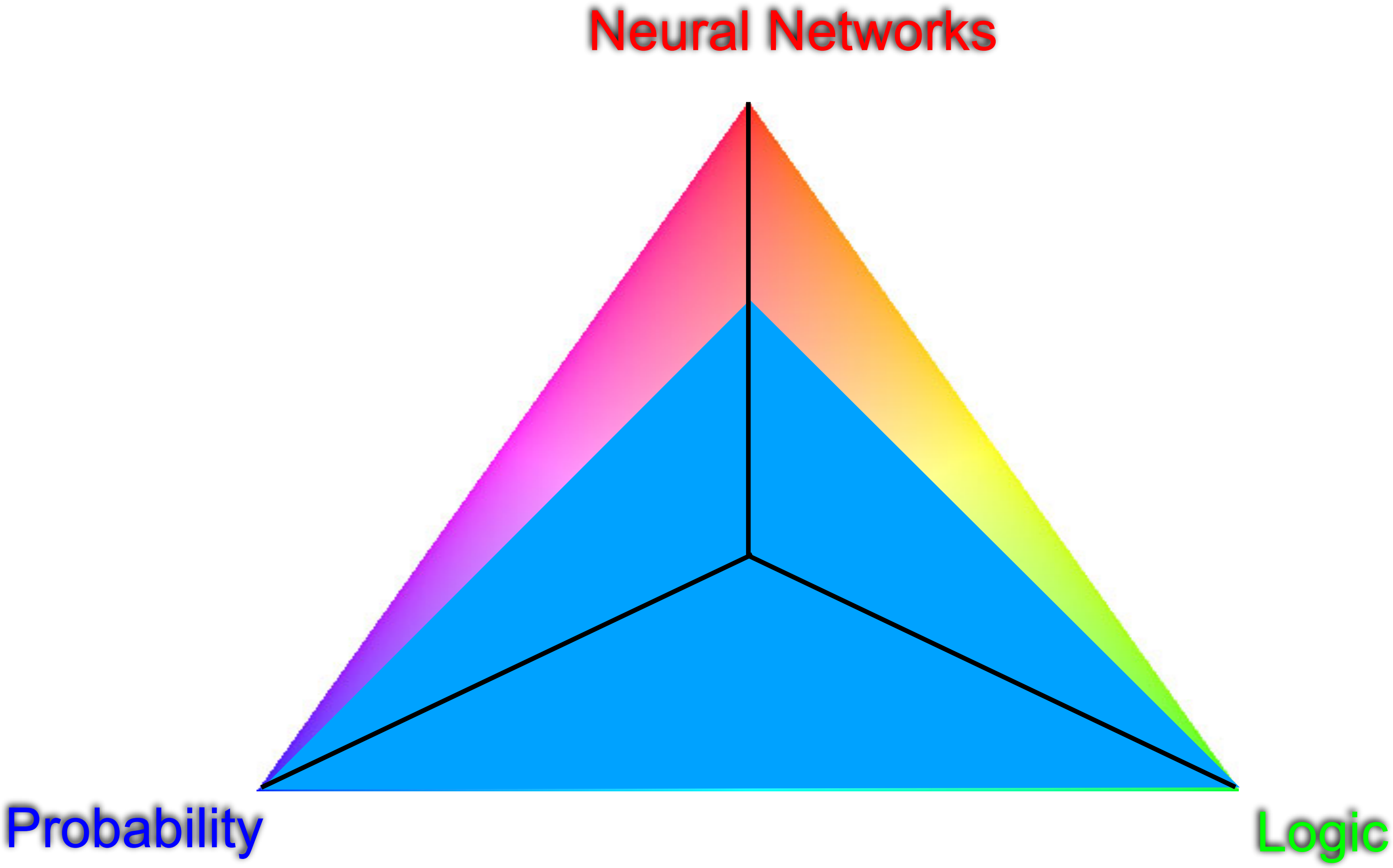


They extend neural network to provide some relational and multihop reasoning.

Logical semantics is not preserved.

R-GCN - Schlichtkrull et al, 2017

Probabilistic reparameterization



They extend StarAI with perception capabilities.

Subsymbols at the level of the constants only

- Not at the level of the atoms (like KGE)
- Not at the level of the rules (like GNNs)

One of the most promising direction for NeSy.

Main problem is scalability.

DeepProbLog (Manhaeve, 2018)
RNM (Marra, 2020)

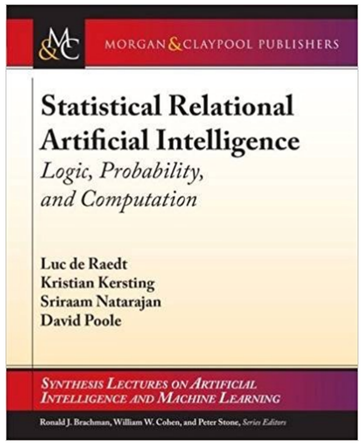
7. Logic vs Probability vs Neural Key Messages

- We have three paradigms in the NeSy spectrum: Logic, Probability and Neural Networks

- An integration of the three should have the original paradigms as special cases
  - Computationally complex

- The integration is usually achieved by sacrificing the base paradigms
  - More scalable

Conclusions



MORGAN & CLAYPOOL PUBLISHERS

Statistical Relational
Artificial Intelligence
Logic, Probability,
and Computation

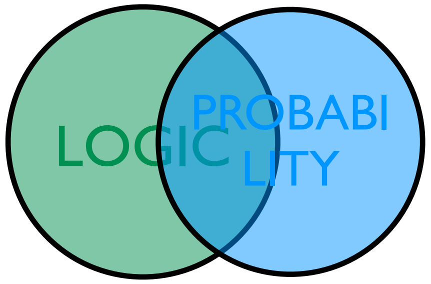
Luc de Raedt
Kristian Kersting
Sriraam Natarajan
David Poole

Synthesis Lectures on Artificial Intelligence and Machine Learning

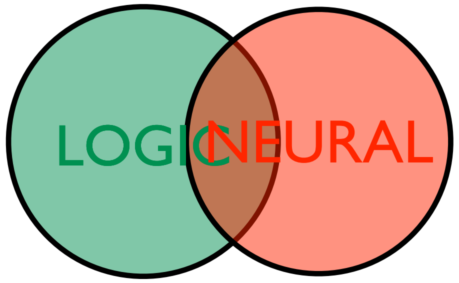
Ronald J. Brachman, William W. Cohen, and Peter Stone, *Series Editors*

Key Message

FROM



TO



StarAI and NeSy share similar problems and thus similar solutions apply

See also [De Raedt et al., IJCAI 20]

The Seven Dimensions

1. Proof vs Model based
2. Directed vs Undirected
3. Type of Logic
4. Symbols vs Subsymbols
5. Parameter vs Structure Learning
6. Semantics
7. Logic vs Probability vs Neural

Many questions to ask

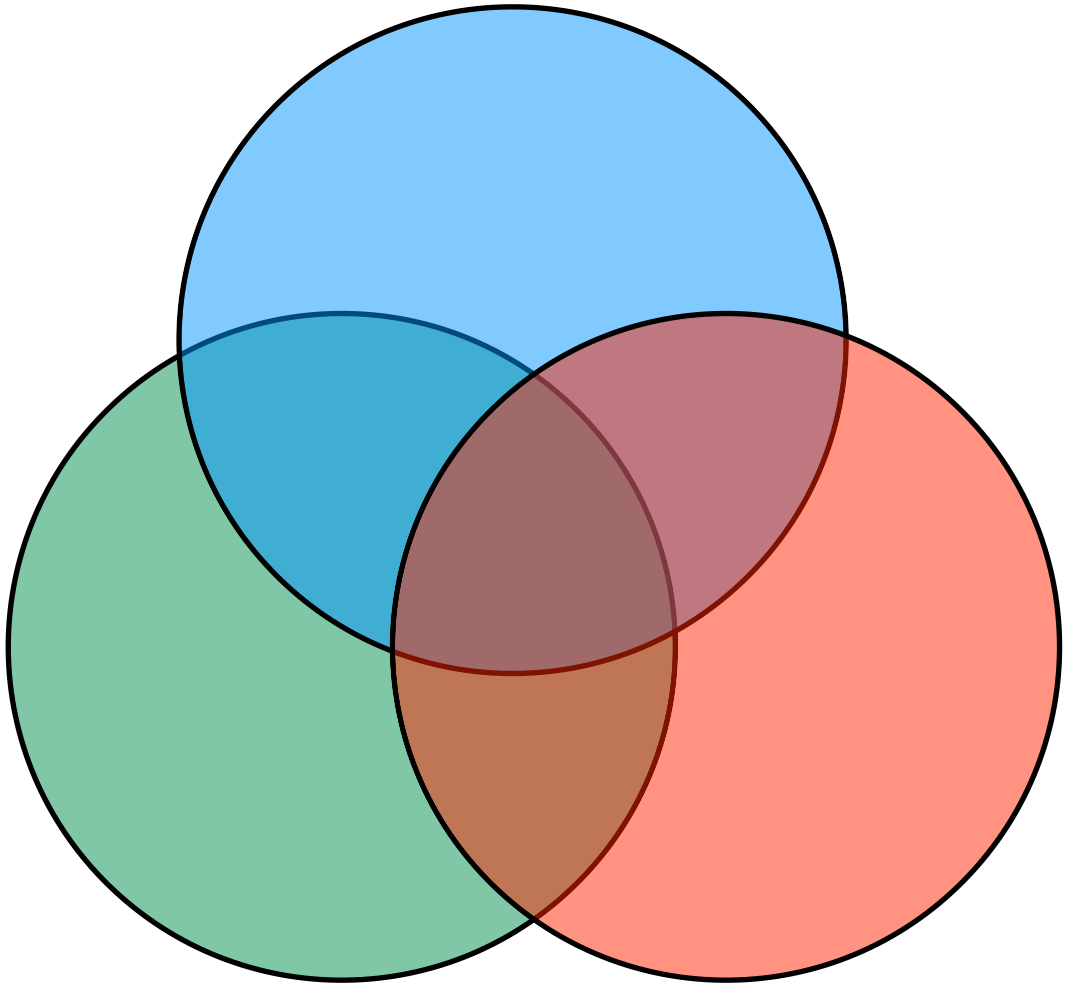
- What properties should integrated representations satisfy ?
    - Should one representation take over ? (As in most approaches to NeSy — push the logic inside and forget about it afterwards)
    - Should one build a pipeline or an interface between the integrated representations ?
    - Should one have the originals as a special case ?
        - (yes we believe you should be able to do all what you can do with the original representations)

Many questions to ask

- Which learning and reasoning techniques apply ?
  - Can you still reason logically / probabilistically ?
  - Can you still apply standard learning methods (like gradient descent) ?
  - Is everything explainable / trustworthy ?
- How to evaluate integrated representations ?
  - 1 + 1 = 3 ?
  - Can they do what the originals can do, and can they do more ?
  - Can they do something different ?

Challenges

- For NeSy,
  - scaling up
  - which models to use
  - real life applications
  - peculiarities of neural nets
  - logical inference can be expensive
- **This is an excellent area for starting researchers / PhDs**



THANKS

References

- Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C.Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symboliclearning and reasoning: A survey and interpretation.CoRR, abs/1711.03902, 2017.
- Matko Bošnjak, Tim Rocktäschel, Jason Naradowsky, and Sebastian Riedel. Programming with a differentiable forth interpreter. InICML,2017.
- William W. Cohen, Fan Yang, and Kathryn Mazaitis. Tensorlog: Deep learning meets probabilistic dbs.CoRR, abs/1707.05390, 2017.
- Andrew Cropper. Playgol: Learning programs through play. InIJCAI 2019, 2019.
- Andrew Cropper and Stephen H. Muggleton. Metagol system. https://github.com/metagol/metagol, 2016.
- Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. InIJCAI, 2011.
- Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning.FLAP, 6, 2019.
- Luc De Raedt, Sebastian Dumančić., Robin Manhaeve and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In IJCAI 2020.
- Luc De Raedt.Logical and relational learning. Springer, 2008.
- Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole.Statistical Relational Artificial Intelligence: Logic, Probability, andComputation. Morgan & Claypool Publishers, 2016.

References

- Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts.Machine Learning, 100, 2015.
- Luc De Raedt, Robin Manhaeve, Sebastijan Duman ˇci´c, Thomas Demeester, and Angelika Kimmig. Neuro-symbolic= neural+ logical+probabilistic. InNeSy @ IJCAI, 2019.
- Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. InEMNLP, 2016.
- Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference.Artif. Intell., 244, 2017.
- Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In IJCAI, 2017.
- Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. InICLR, 2019.
- Sebastijan Duman ˇci´c, Tias Guns, Wannes Meert, and Hendrik Blockeel. Learning relational representations with auto-encoding logic programs.InIJCAI, 2019.
- Kevin Ellis, Lucas Morales, Mathias Sablé-Meyer, Armando Solar-Lezama, and Josh Tenenbaum. Learning libraries of subroutines forneurally-guided bayesian program induction. InNeurIPS, 2018.
- Kevin Ellis, Maxwell I. Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. Write, execute, assess: Program synthesiswith a REPL.CoRR, abs/1906.04604, 2019.
- Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data.J. Artif. Intell. Res., 61, 2018.

References

- Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt.Inference and learning in probabilistic logic programs using weighted boolean formulas.Theory and Practice of Logic Programming, 15, 2015.
- Peter Flach.Simply Logical: Intelligent Reasoning by Example. John Wiley & Sons, Inc., 1994.
- Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. InIJCAI, 1999.
- Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice.Synthesis lectures on artificialintelligence and machine learning, 6, 2012.
- L. Getoor and B. Taskar, editors.An Introduction to Statistical Relational Learning. MIT Press, 2007.
- Francesco Giannini, Michelangelo Diligenti, Marco Gori, and Marco Maggini. On a convex logic fragment for learning and reasoning.IEEETFS, 27, 2018.CV Radhakrishnan et al.:Preprint submitted to Elsevier
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry.arXivpreprint arXiv:1704.01212, 2017.
- Goldman, O., Latcinnik, V., Naveh, U., Globerson, A., & Berant, J.. Weakly-supervised semantic parsing with abstract examples. ACL 2018
- Bernd Gutmann, Angelika Kimmig, Kristian Kersting, and Luc De Raedt. Parameter learning in probabilistic databases: A least squaresapproach. InECML&PKDD, 2008.
- Manfred Jaeger. Model-theoretic expressivity analysis. In Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors,Probabilistic Inductive Logic Programming - Theory and Applications, volume 4911 of LNCS. Springer, 2008.

References

- Ashwin Kalyan, Abhishek Mohta, Oleksandr Polozov, Dhruv Batra, Prateek Jain, and Sumit Gulwani. Neural-guided deductive search forreal-time program synthesis from examples. InICLR, 2018.
- Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. In L. Getoor and B. Taskar, editors,An introduction toStatistical Relational Learning. MIT Press, 2007.
- Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. InICML, 2005.
- Daphne Koller and Nir Friedman.Probabilistic Graphical Models - Principles and Techniques. MIT Press, 2009.
- Marco Lippi and Paolo Frasconi. Prediction of protein beta-residue contacts by markov logic networks with grounding-specific weights.Bioinform., 25, 2009.
- John W Lloyd.Foundations of logic programming. Springer Science & Business Media, 2012.
- Daniel Lowd and Pedro Domingos. Efficient weight learning for markov logic networks. InECML&PKDD, 2007.
- Robin Manhaeve, Sebastijan Duman ̌ci ́c, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logicprogramming. InNeurIPS, 2018.
- Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes,words, and sentences from natural supervision. In ICLR, 2019.
- Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines. In ECAI, 2020.
- Giuseppe Marra and Ondrej Kuželka. Neural markov logic networks. CoRR, abs/1905.13462, 2019.

References

- Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledgebases and natural language. InAAAI, 2020.
- Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. InUAI, 2017.
- Stephen Muggleton. Stochastic logic programs.Advances in inductive logic programming, 32, 1996.
- Maxwell I. Nye, Armando Solar-Lezama, Josh Tenenbaum, and Brenden M. Lake. Learning compositional rules via neural program synthesis.In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors,Advances in Neural InformationProcessing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual,2020.
- David Poole. The independent choice logic and beyond. InProbabilistic Inductive Logic Programming - Theory and Applications, volume4911 ofLNCS. Springer, 2008.
- Matthew Richardson and Pedro M. Domingos. Markov logic networks.Machine Learning, 62, 2006.
- Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. InNIPS, 2017.
- Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. InNAACL HLT, 2015.
- Stuart Russell. Unifying logic and probability.Communications of the ACM, 58, 2015.

References

- Xujie Si, Mukund Raghothaman, Kihong Heo, and Mayur Naik. Synthesizing datalog programs using numerical relaxation. InIJCAI, 2019.
- Lazar Valkov, Dipak Chaudhari, Akash Srivastava, Charles A. Sutton, and Swarat Chaudhuri. Houdini: Lifelong learning as program synthesis.InNeurIPS, 2018.
- Guy Van den Broeck, Dan Suciu, et al. Query processing on probabilistic data: A survey.Foundations and Trends® in Databases, 7, 2017.
- Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators.CoRR, abs/2002.06100, 2020.
- Wenya Wang and Sinno Jialin Pan. Integrating deep learning with logic fusion for information extraction.CoRR, abs/1912.03041, 2019.
- Wang, P., Wu, Q., Shen, C., Hengel, A. V. D., & Dick, A. . Explicit knowledge-based reasoning for visual question answering. IJCAI 2017
- Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. Nlprolog: Reasoning with weak unification forquestion answering in natural language. InACL, 2019.
- Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolicknowledge. InICML, 2018.
- Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. InNIPS, 2017.
- Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. InProceedings of theTwenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI, pages 1755–1762,

References

- Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoningfrom vision and language understanding. InNeurIPS, 2018.
- Lotfi A Zadeh. Fuzzy logic and approximate reasoning.Synthese, 30(3-4):407–428, 1975.
- Pedro Zuidberg Dos Martires, Vincent Derkinderen, Robin Manhaeve, Wannes Meert, Angelika Kimmig, and Luc De Raedt. Transformingprobabilistic programs into algebraic circuits for inference and learning. InProgram Transformations for ML Workshop at NeurIPS, 2019.
- Gustav Šourek, Vojtech Aschenbrenner, Filip Zelezný, Steven Schockaert, and Ondrej Kuželka. Lifted relational neural networks: Efficientlearning of latent relational structures.J. Artif. Intell. Res., 62, 2018