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

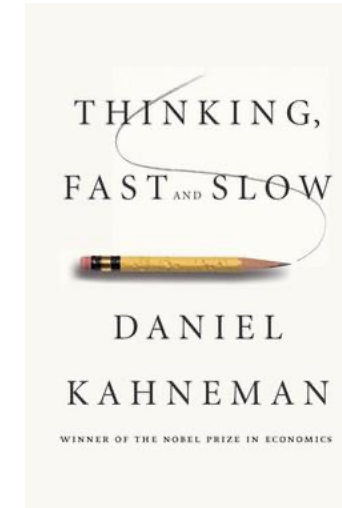
DTAI MACHINE LEARNING · ÖREBRO UNIVERSITET · KU LEUVEN · LEUVEN.AI INSTITUTE · EOS THE EXCELLENCE OF SCIENCE · WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM · TAILOR · erc

You will 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.

AUTO

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.

AUTO

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

Who can go first ?

A. The red car

B. The blue van

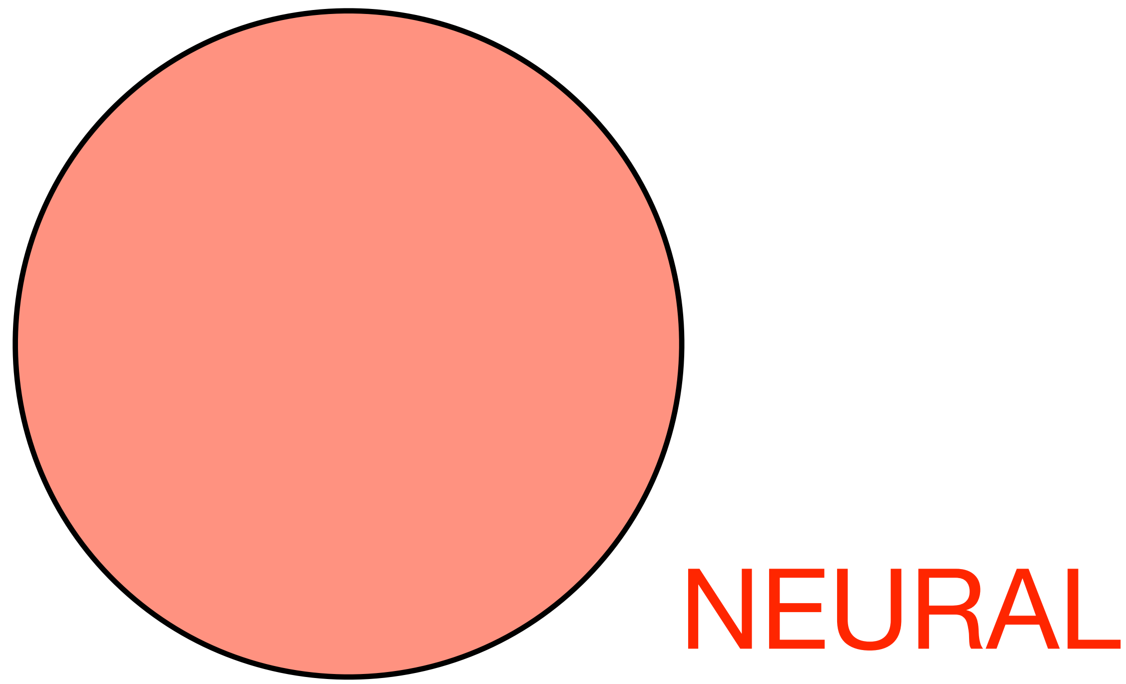
C. The white car

Sub-symbolic perception

Reasoning

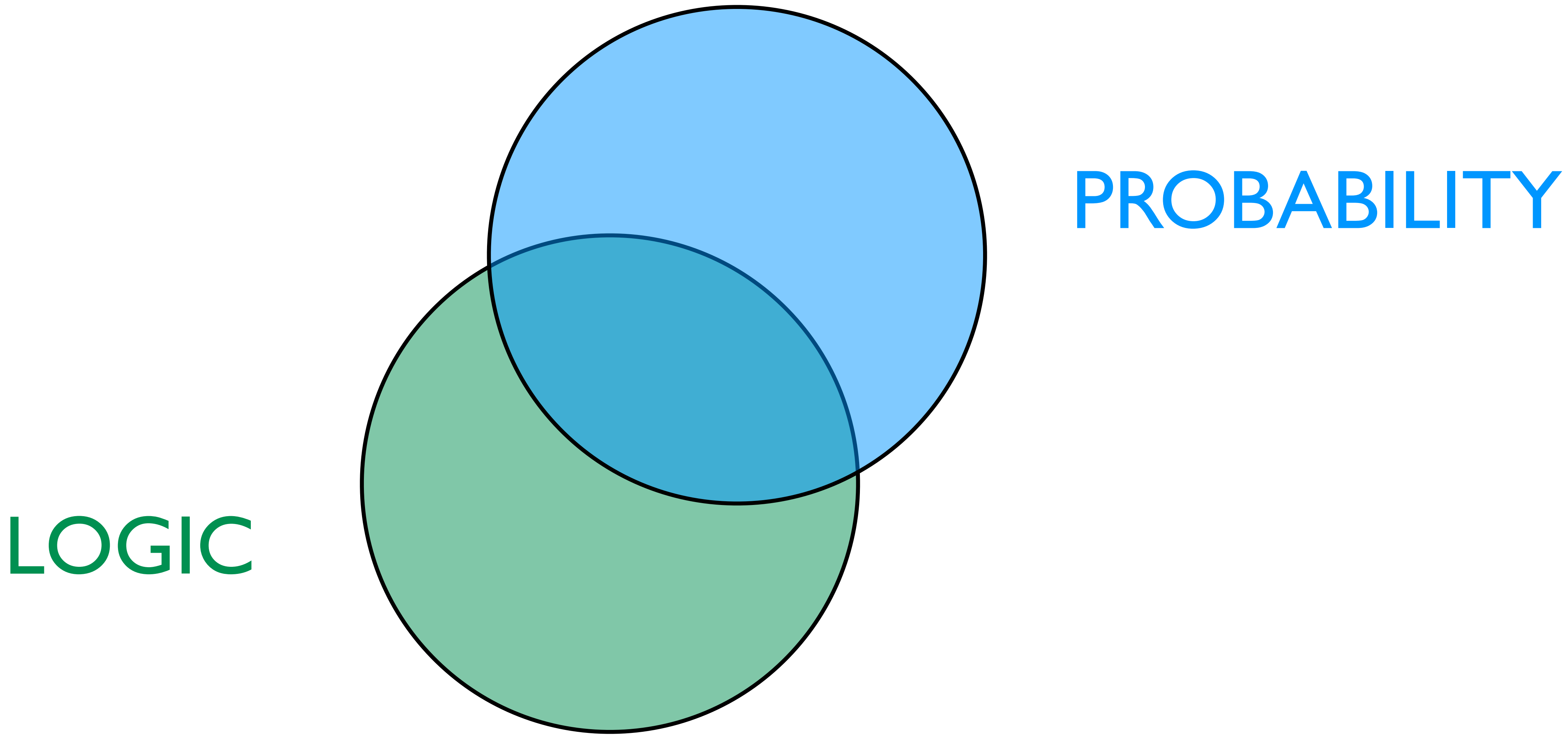
Thinking fast

MAIN PARADIGM in AI
Focus on Learning



Thinking slow = reasoning

TWO MAIN PARADIGMS in AI



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

Applications

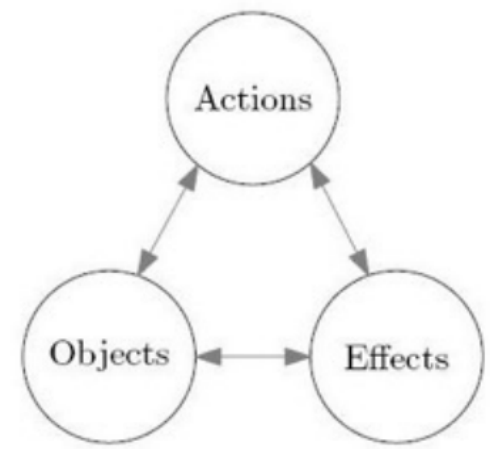
Relational Affordances

- **Object Affordance:**
  What can one do with
  particular object?

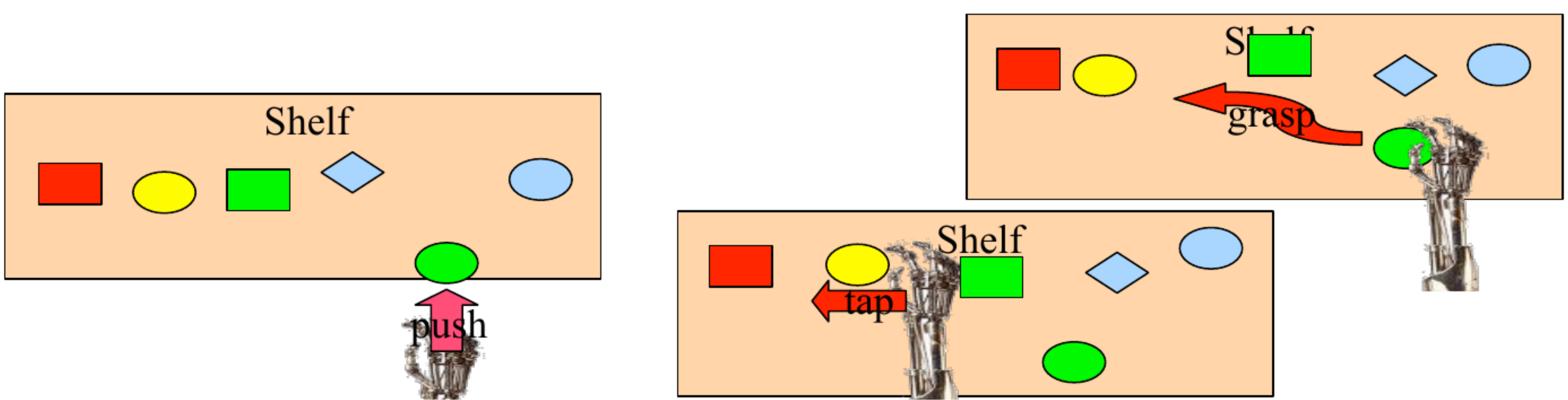
- **Relational Affordance:**
  in a particular context?

  with multiple objects and
  relations among them

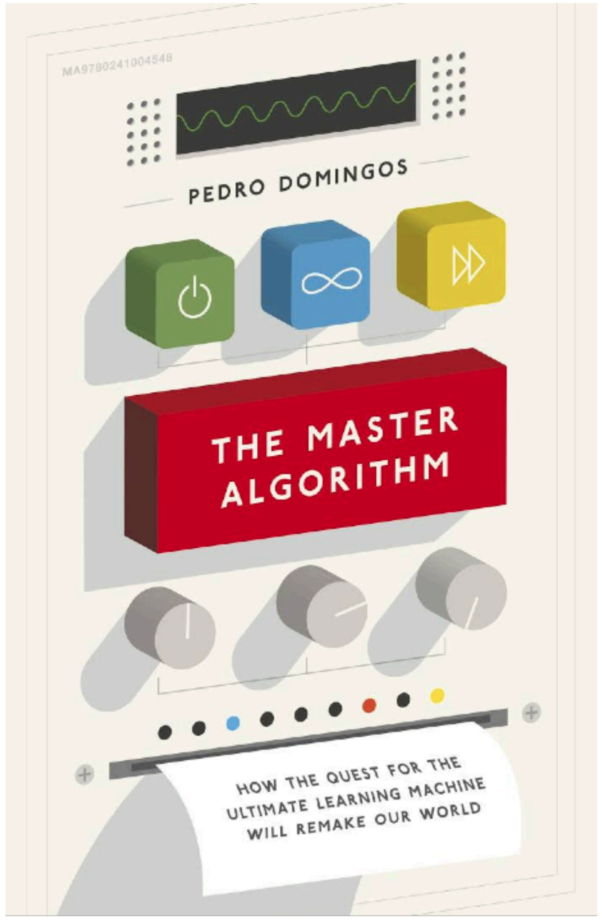
- Use of statistical relational learning, **probabilistic programming for learning, reasoning and planning !**

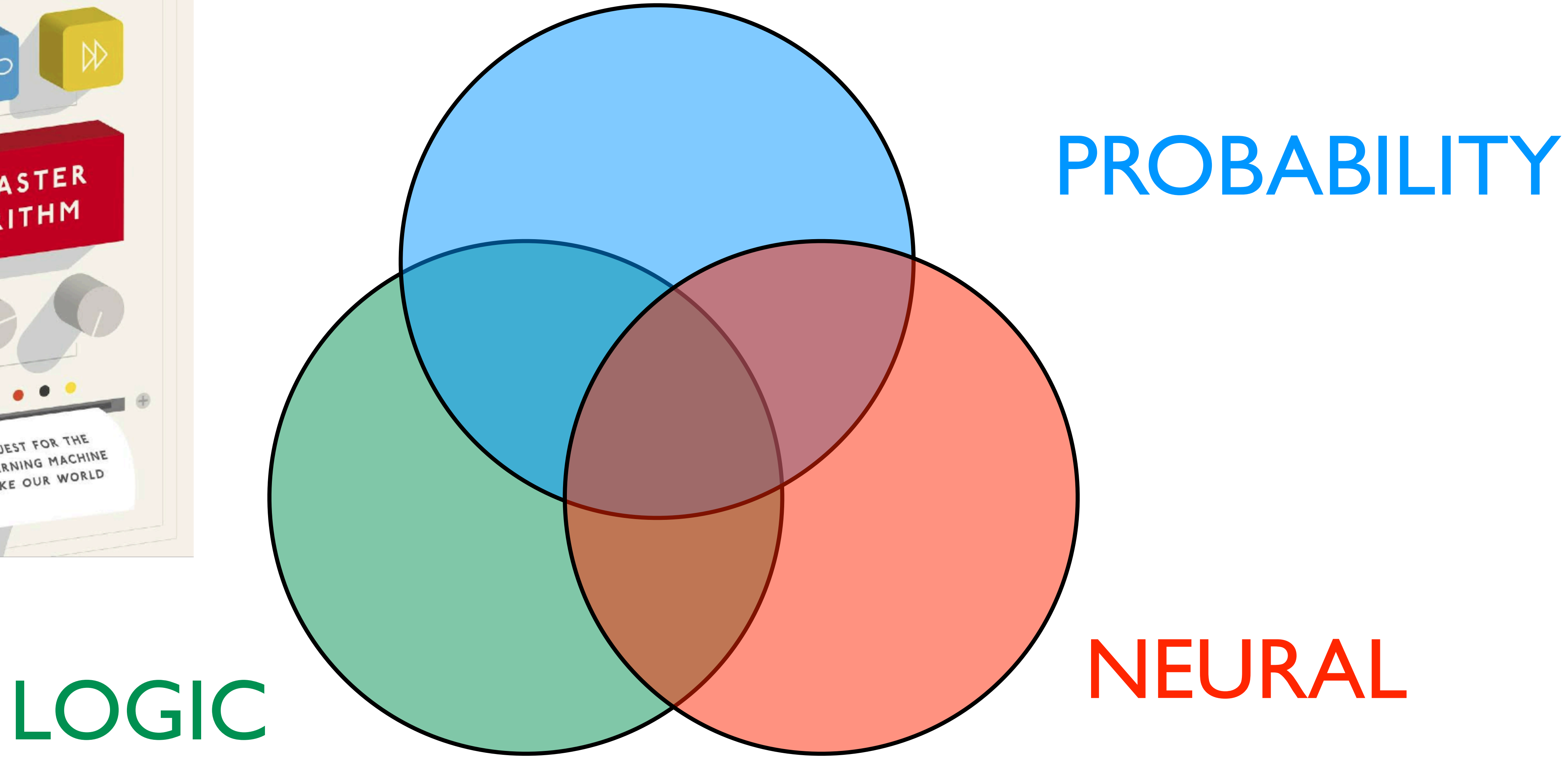


Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection



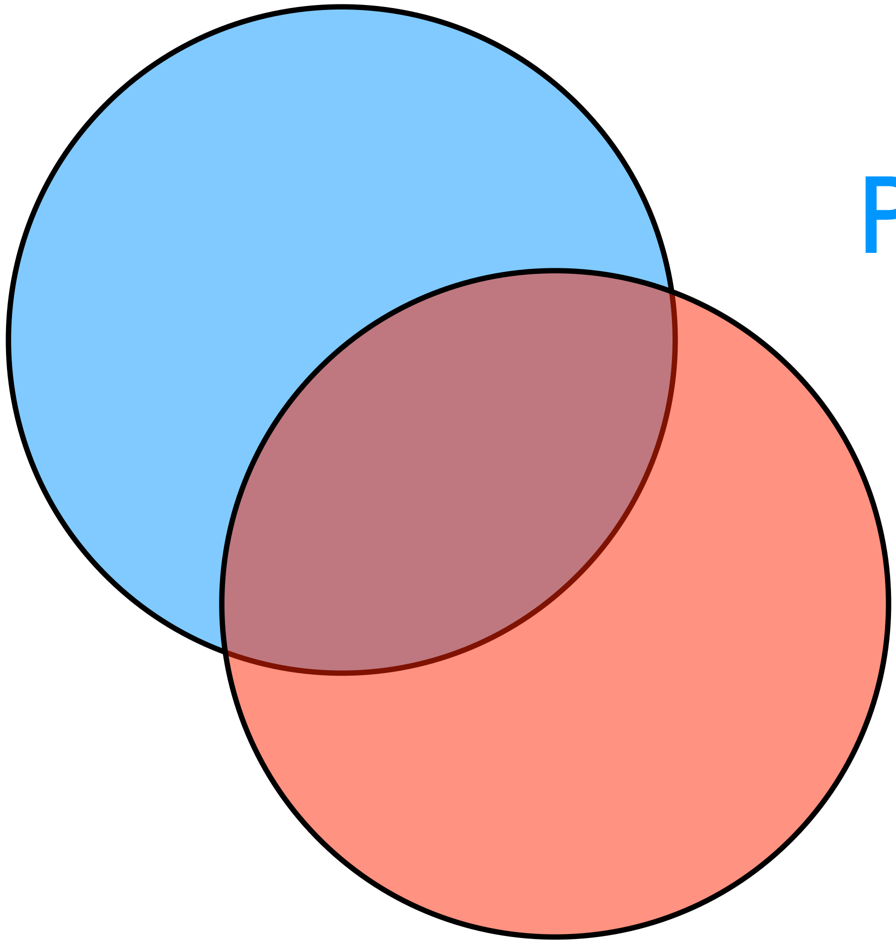
Learning





How to integrate these three paradigms in AI ?

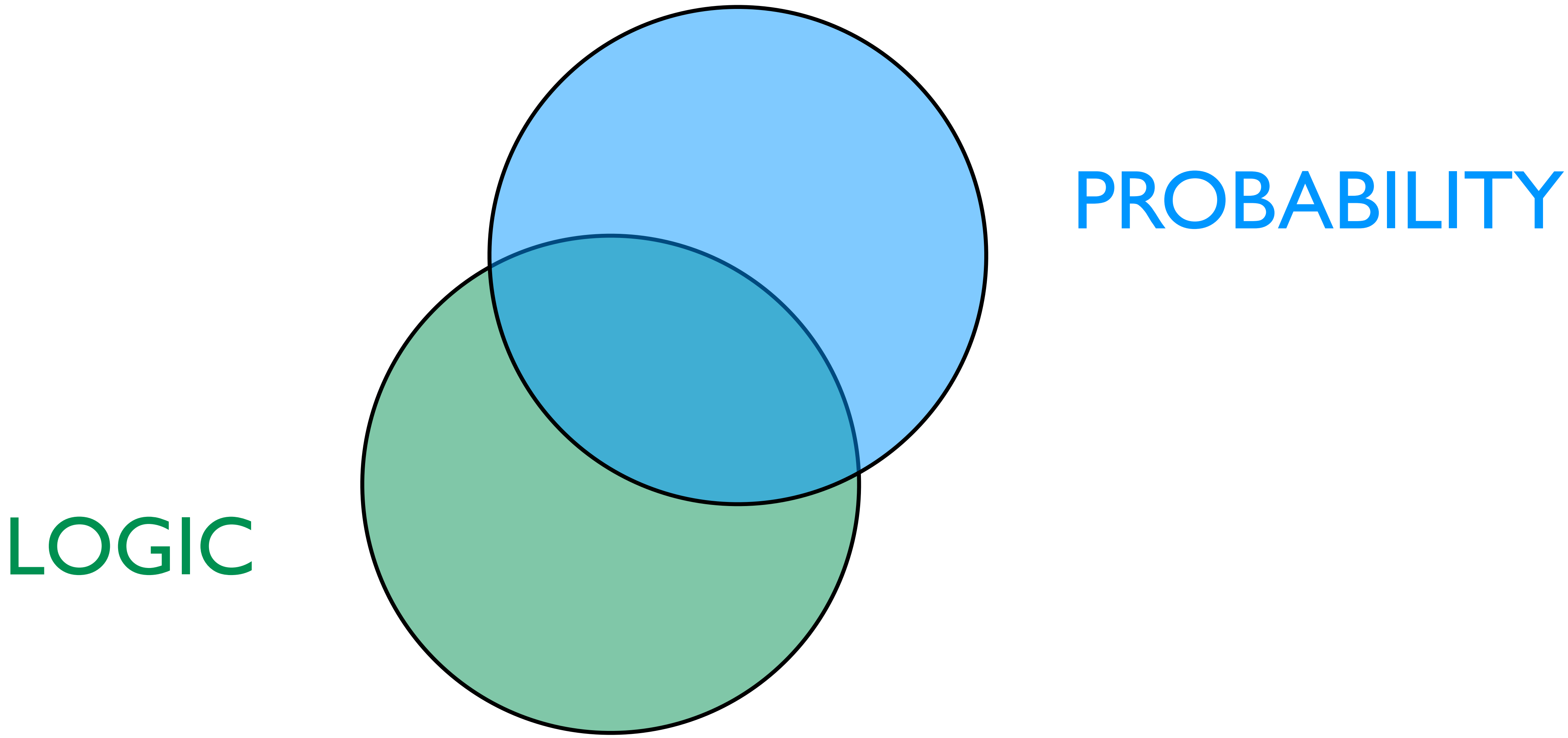
A lot of ML



Well studied from a LEARNING perspective in Deep Learning

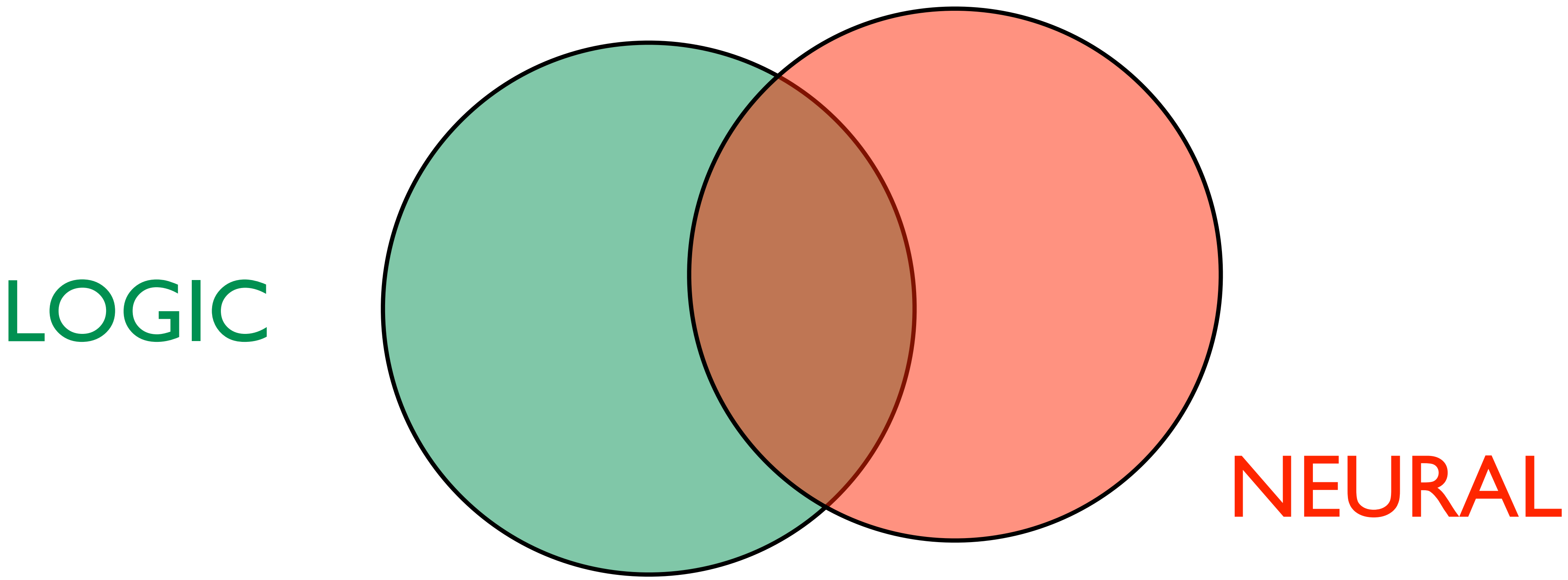
Thinking slow = reasoning

TWO MAIN PARADIGMS in AI

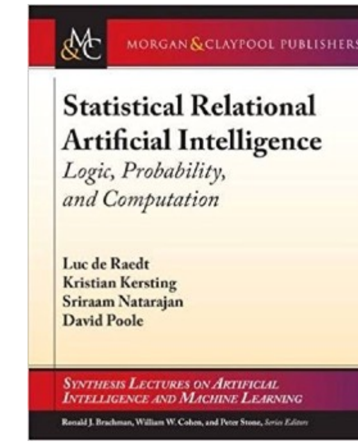


Their integration has been well studied in
Probabilistic (Logic) Programming and Statistical Relational AI (StarAI)

State of the Art

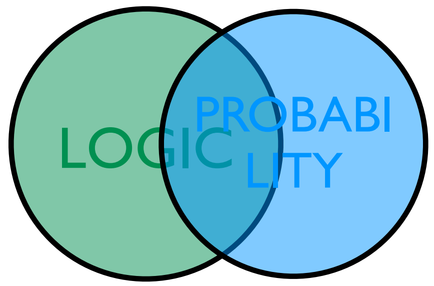


Being studied from a LEARNING perspective in Neuro Symbolic Computation

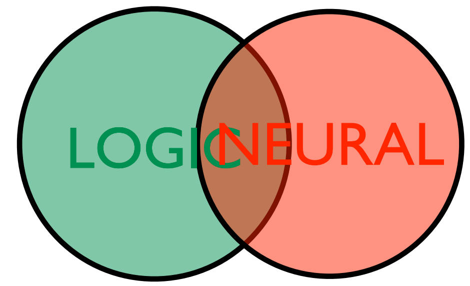


Key Message

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, and long version on arXiv

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 (e.g. space of programs)

Addition

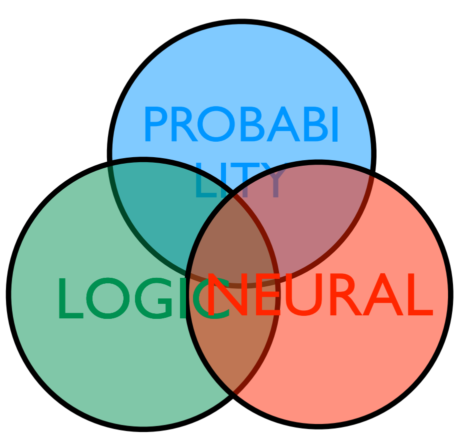
Learn to add the sum of lists of MNIST images

+ = ?

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 ?



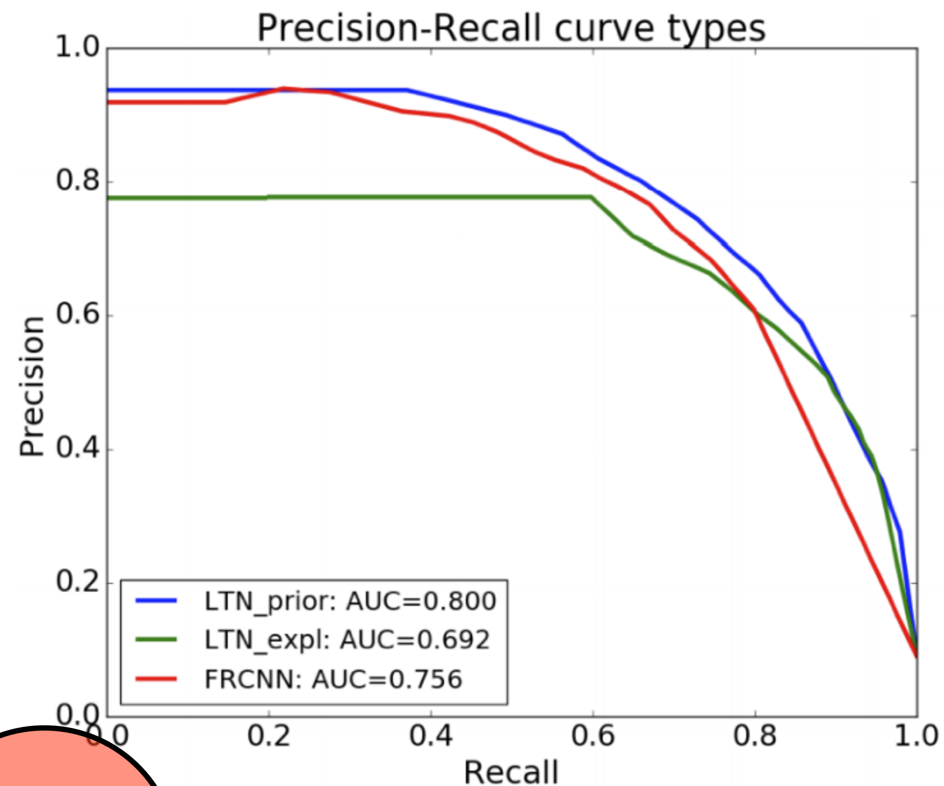
DeepProbLog, Manhaeve et al, NeurIPS 2018

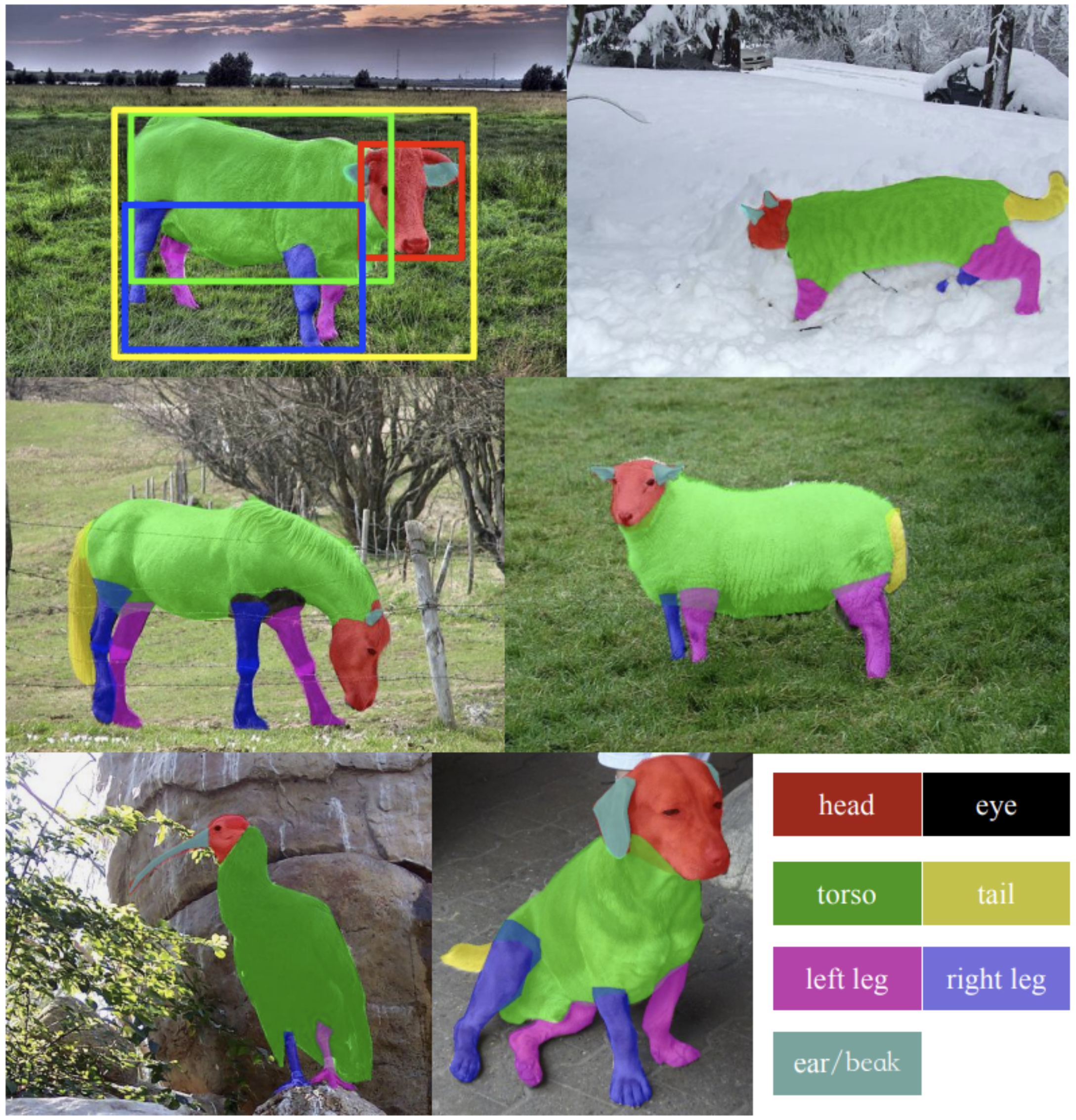
Semantic Image Interpretation

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

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

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





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?

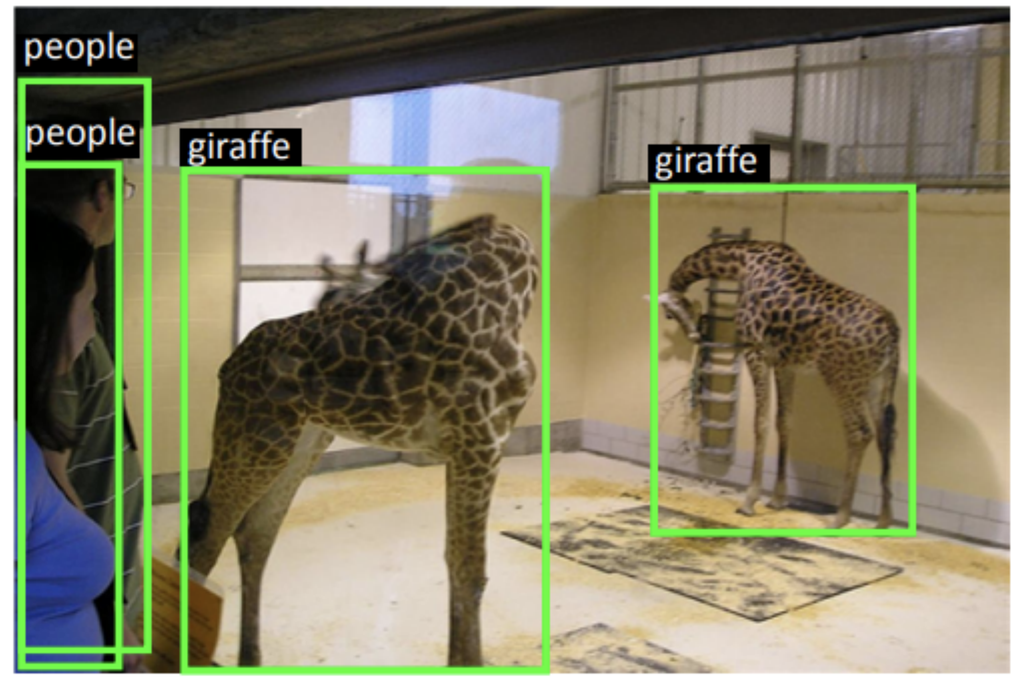
Figure 1: Human reasoning is interpretable and disentangled: we first draw abstract knowledge of the scene via visual perception and then perform logic reasoning on it. This enables compositional, accurate, and generalizable reasoning in rich visual contexts.

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

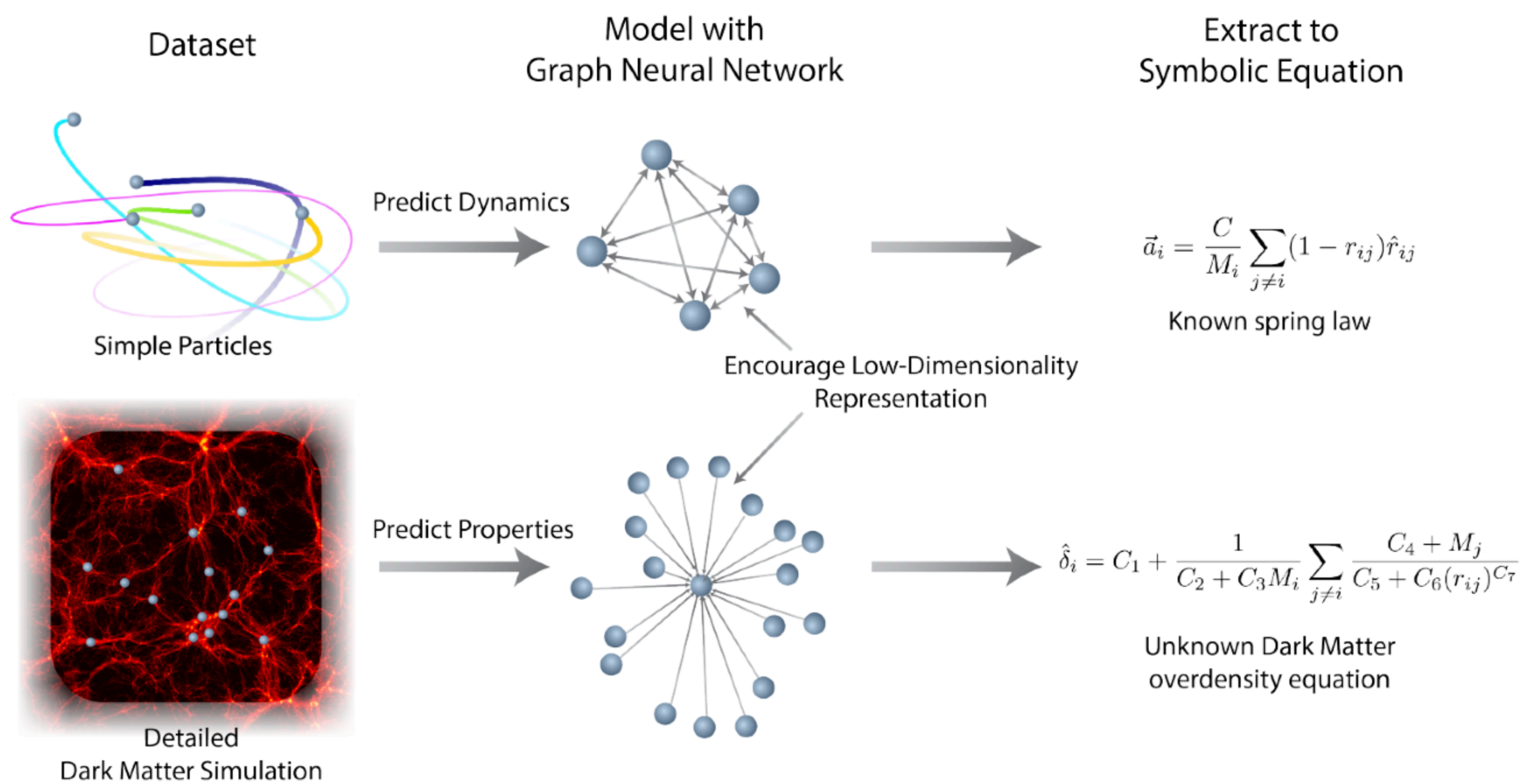
Visual Question: How many giraffes in the image?
Answer: Two. Reason: Two giraffes are detected.

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

(New) Scientific Discovery



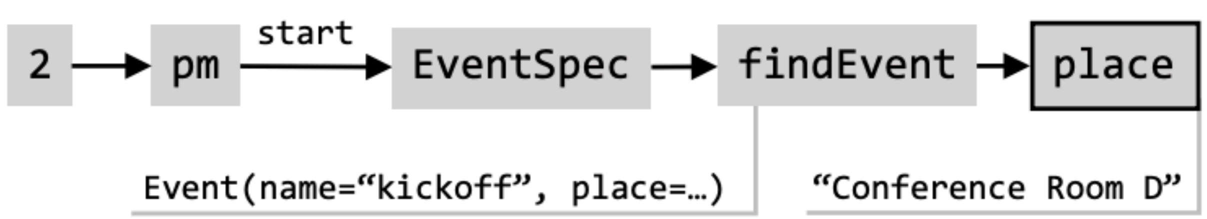
Cranmer, et al. NeurIPS 2020

(New) Dialog Systems

User: *Where is my meeting at 2 this afternoon?*

```
place(findEvent(EventSpec(start=pm(2))))
```

(1)



Agent: *It's in Conference Room D.*

Dialogues represented as symbolic programs (e.g. dataflow graphs)

Andreas, Jacob, et al. ACL, 2020

(New) Game Playing

NukkAi
@nukkailab1

🤖 : 6136
👥 : 5238

NooK won The Nukkai Challenge!

The NeSy NooK system
defeats eight
world bridge champions
in Paris (2022)

Talk in context of TAILOR network
by Veronique Ventos
Jan 30, 15.30

https://challenge.nukk.ai/

Both StarAI and NeSy

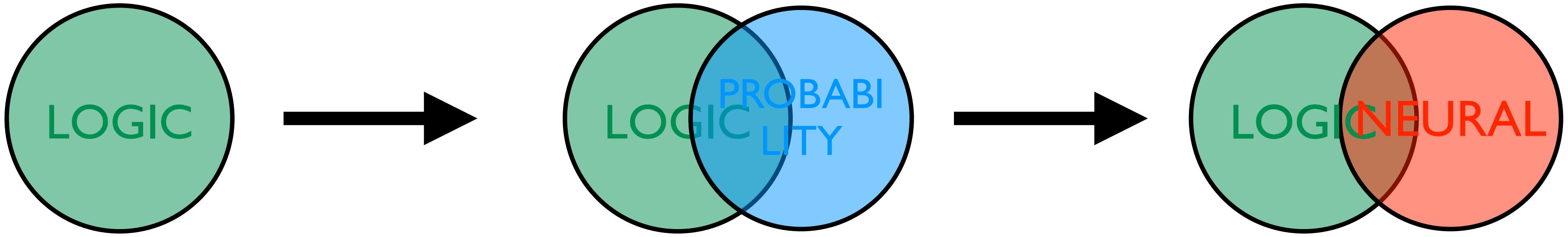
- Structured environments
  - objects, and
  - relationships amongst them
- and possibly
  - using background knowledge
- cope with uncertainty and/or perception
- learn from data and reason with knowledge

Power of Logic of Programs

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.

facts :
burglary = true

earthquake.
hears_alarm_john.

alarm :– earthquake.

alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.

calls_john :– alarm, hears_alarm_john.

LOGIC

erc

Logic Programs

as in the programming language Prolog

Propositional logic program

burglary.
hears_alarm_mary.

earthquake.
hears_alarm_john.

alarm :– earthquake.

alarm :– burglary.

rule:
calls_mary =true IF alarm = true AND hears_alarm_mary = true

calls_mary :– alarm, hears_alarm_mary.

calls_john :– alarm, hears_alarm_john.

LOGIC

Logic Programs

as in the programming language Prolog

Propositional logic program

burglary.
hears_alarm_mary.

earthquake.
hears_alarm_john.

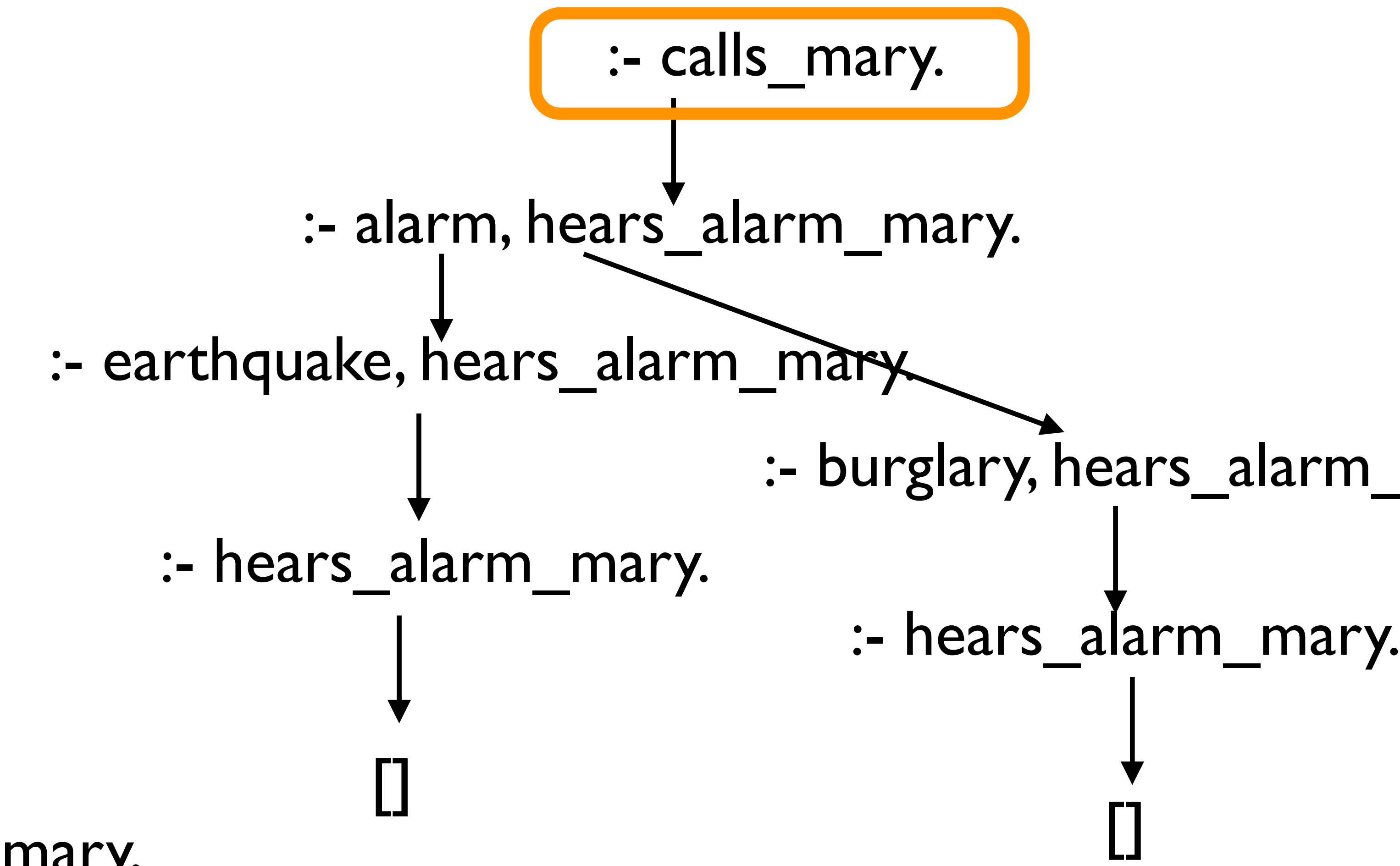
alarm :– earthquake.

alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.

calls_john :– alarm, hears_alarm_john.

Two proofs (by refutation)



A proof-theoretic view
backward chaining

Logic as constraints

as in SAT solvers

Propositional logic

IF AND

calls(mary) ← hears_alarm(mary) ∧ alarm

calls(john) ← hears_alarm(john) ∧ alarm

OR

alarm ← earthquake v burglary

Model / Possible World

{ 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

BOTH for model and proof-based appraoch

LOGIC

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)

erc

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).
```

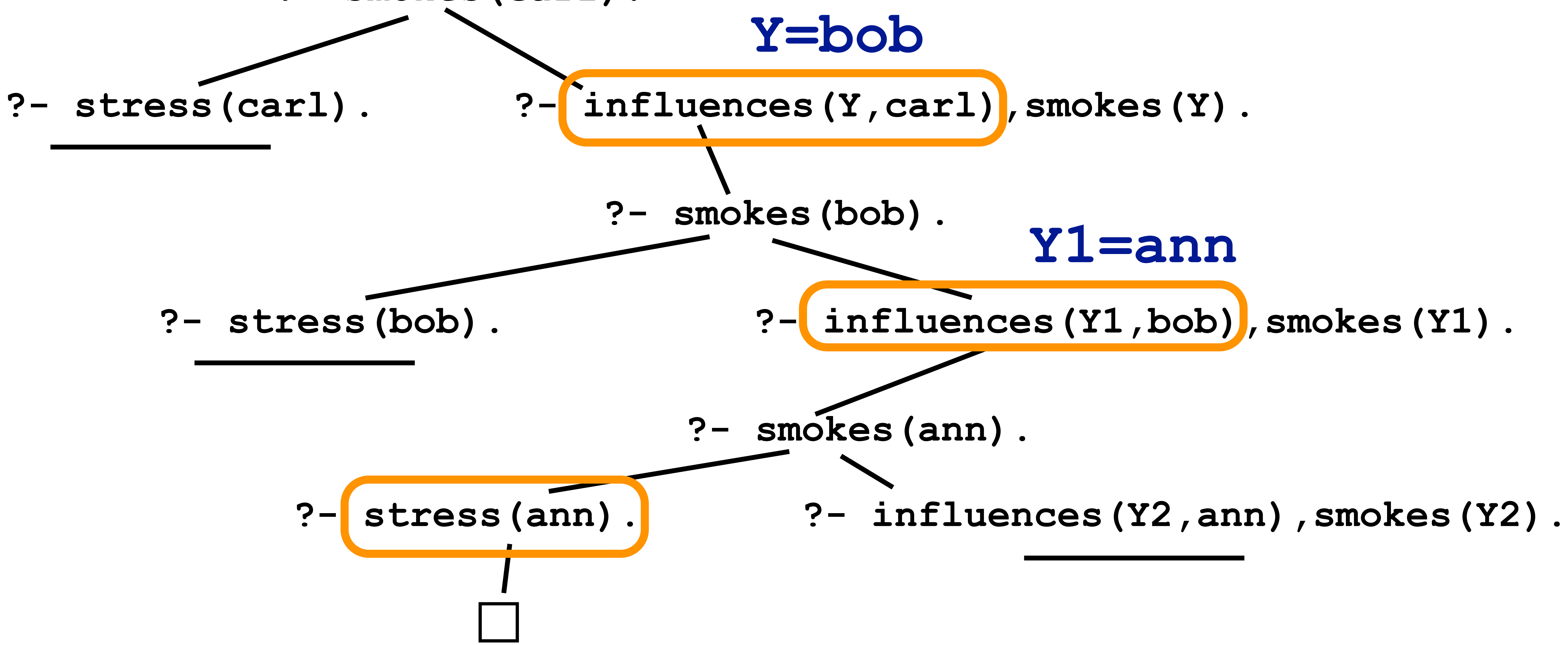
Clark's completion's as a grounding is incorrect for Prolog when there are cycles

but it is too hard to explain why here

```
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)
```

Logical Reasoning Proofs

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

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



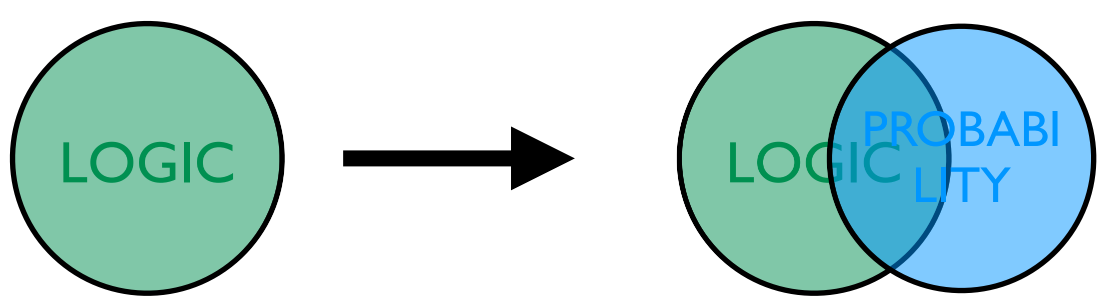
facts used in successful derivation:

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

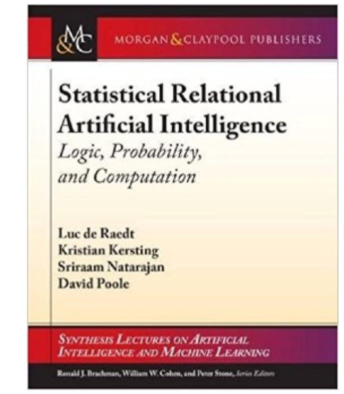
1. Proof vs Model based the logic dimension

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

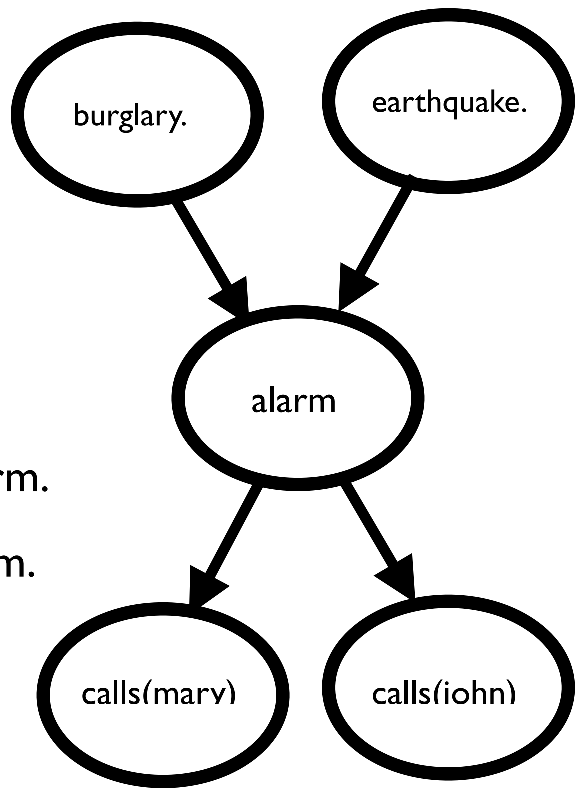
1. Proof vs Model based
2. Directed vs Undirected

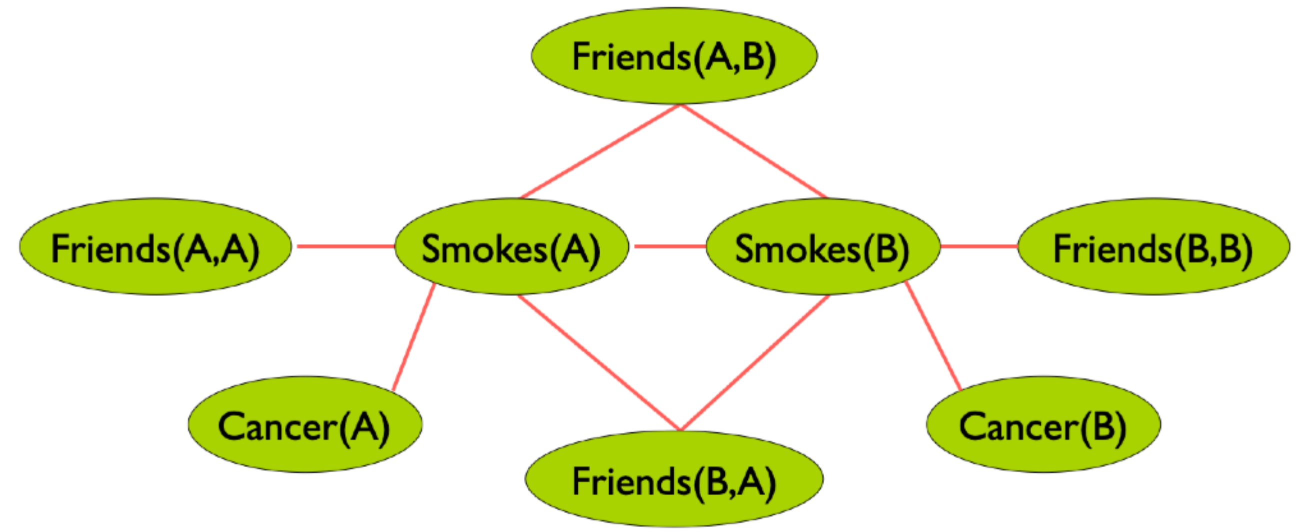


2. Directed vs Undirected the PGM / StarAI dimension



0.1 :: burglary.
0.05 :: earthquake.
alarm :– earthquake.
alarm :– burglary.
0.7::calls(mary) :– alarm.
0.6::calls(john) :– alarm.





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

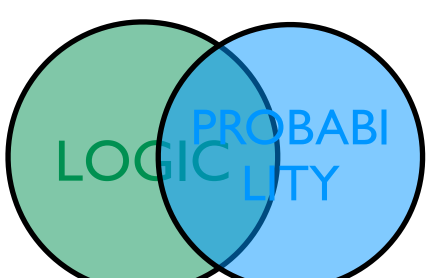
$1.1 \quad \forall x,y\ Friends(x,y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

Markov Logic

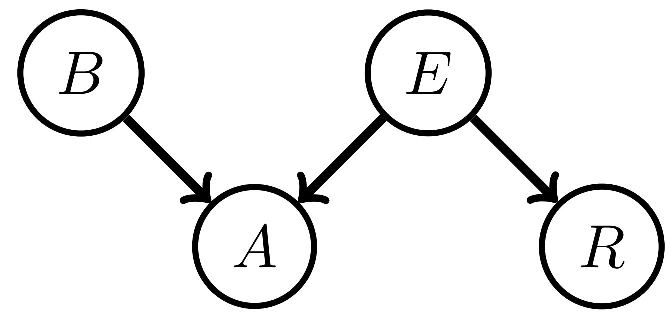
undirected
Markov Net
model theoretic



key representatives

erc

Bayesian Net



$\mathbf{P}(A|B,E)$

alarm (= true)	Burglar	Earthquake
0.9999	true	true
0.99	true	false
0.99	false	true
0.0001	false	true

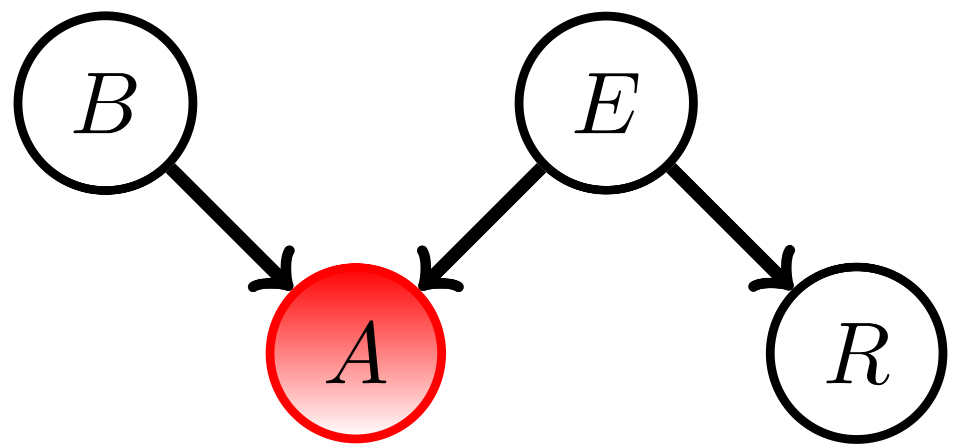
$\mathbf{P}(R|E)$

radio	Earthquake
1	true
0	false

The remaining tables are $P(b) = 0.01$ and $P(e) = 0.000001$. The tables and graphical structure fully specify the joint distribution $\mathbf{P}(A, R, E, B)$.

Queries

Initial evidence: The alarm is sounding



$$P(b|a) = \frac{P(b,a)}{P(a)} = \frac{\sum_{e,r} P(b,e,a,e)}{\sum_{b,e,r} P(b,e,a,r)}$$

$$= \frac{\sum_{e,r} P(r|b,e)P(b)P(e)P(r|e)}{\sum_{b,e,r} P(a|b,e)P(b)P(e)P(r|e)} \approx 0.99$$

Logic Programs

as in the programming language Prolog

Propositional logic program

burglary.
hears_alarm(mary).

earthquake.
hears_alarm(john).

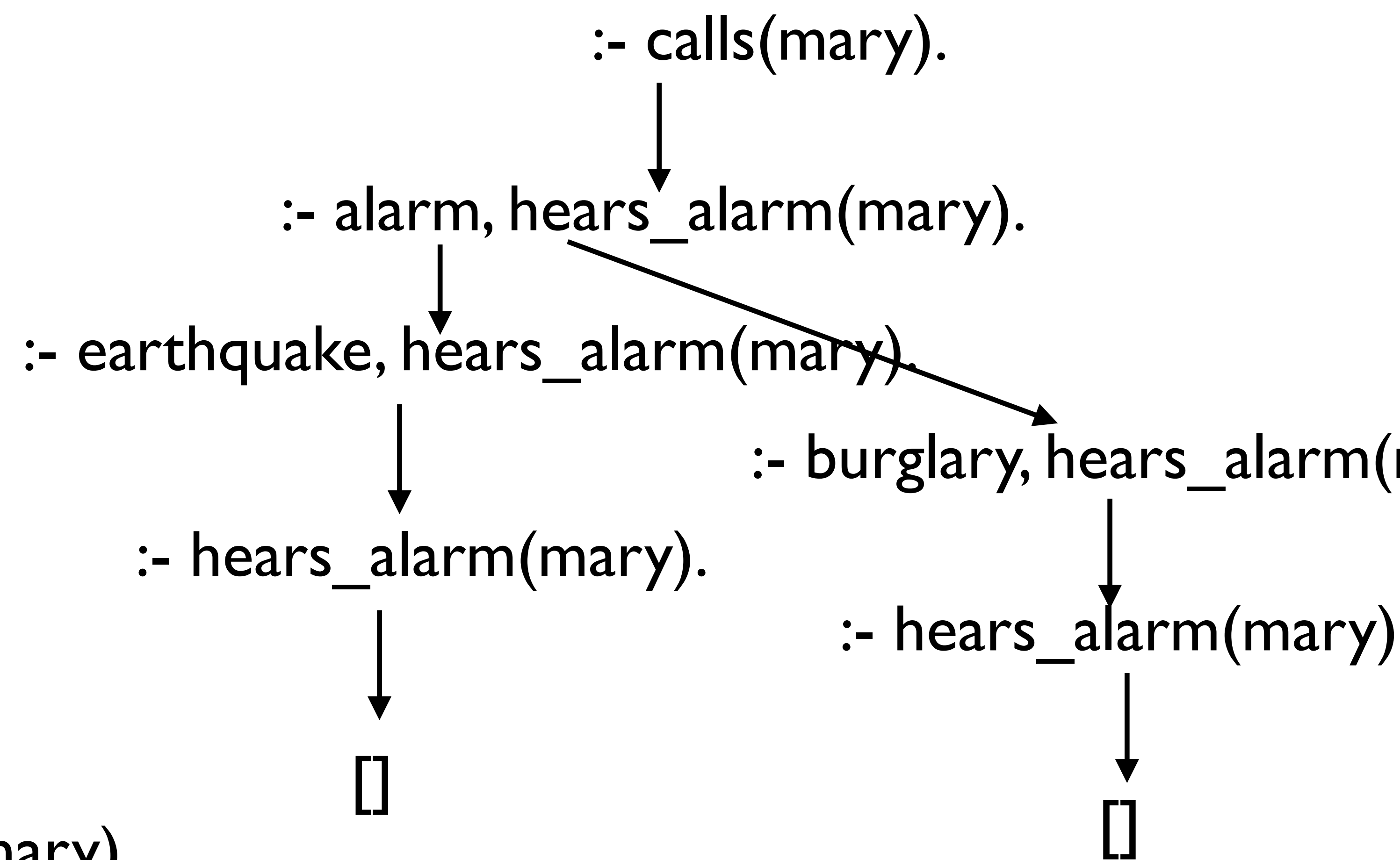
alarm :– earthquake.

alarm :– burglary.

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

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

Two proofs (by refutation)



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).

Probabilistic facts

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

alarm :– earthquake.
alarm :– burglary.

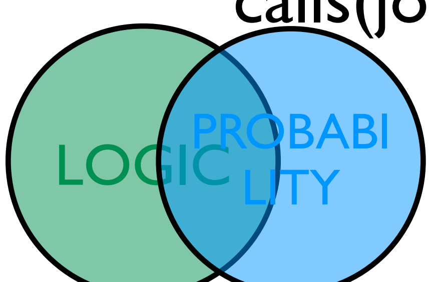
calls(mary) :– alarm, hears_alarm(mary).
calls(john) :– alarm, hears_alarm(john).

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).

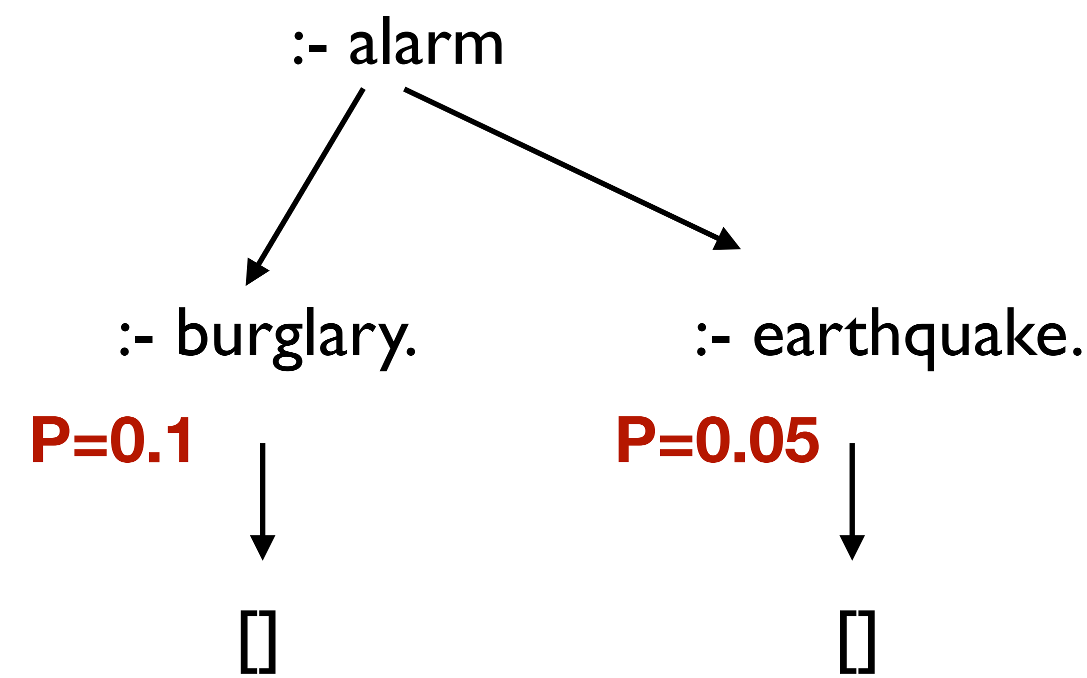
alarm :– earthquake.

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).

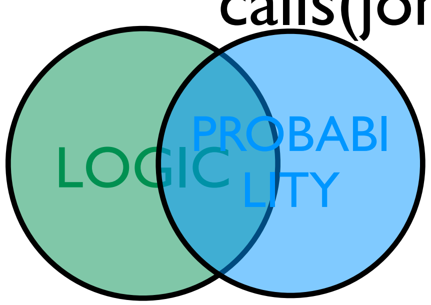
0.05 ::earthquake.
0.6 ::hears_alarm(john).

alarm :– earthquake.

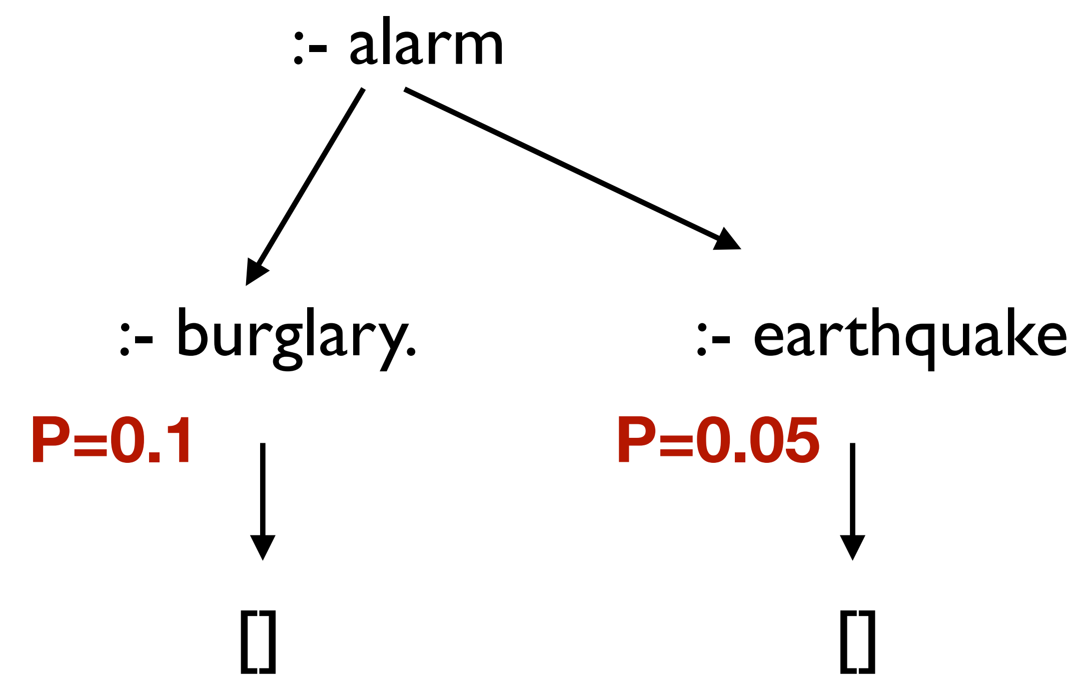
alarm :– burglary.

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

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



Disjoint sum problem



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

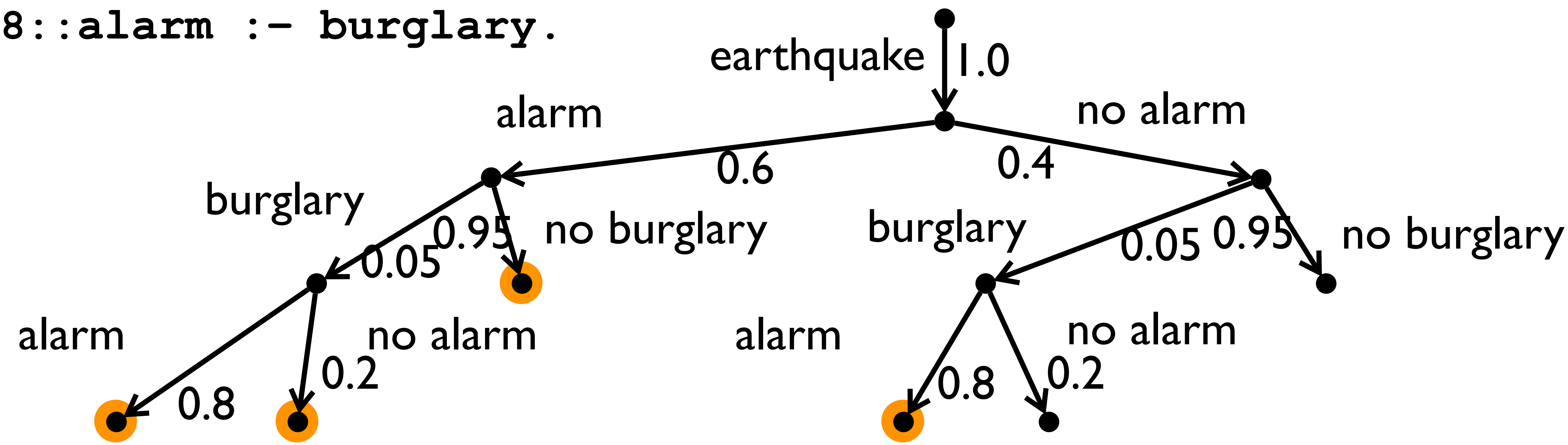
P(alarm) = P(burg OR earth)
= P(burg) + P(earth) - P(burg AND earth)
=/= P(burg) + P(earth)

Probabilistic Logic Program Semantics

```
earthquake.
0.05::burglary.

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

[Vennekens et al, ICLP 04]

probabilistic causal laws



$$P(\text{alarm})=0.6\times0.05\times0.8+0.6\times0.05\times0.2+0.6\times0.95+0.4\times0.05\times0.8$$

Probabilistic Logic Program Semantics

Propositional logic program

0.1 :: burglary.

0.05 :: earthquake.

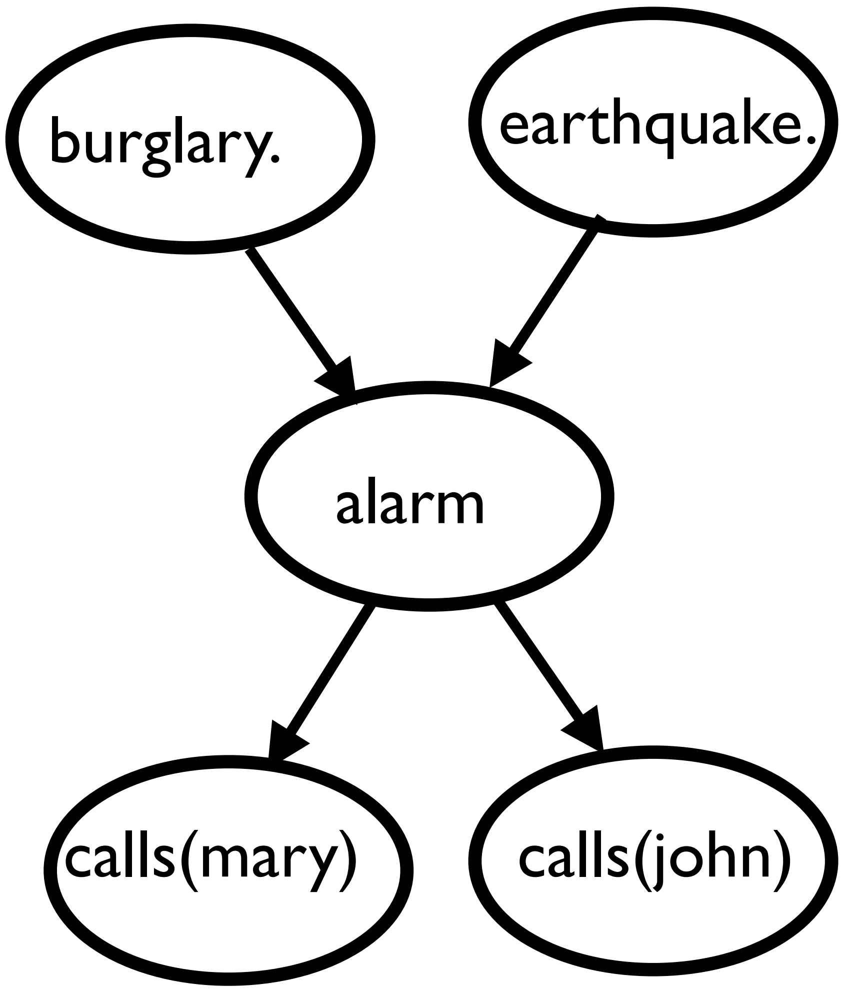
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.

Bayesian Network



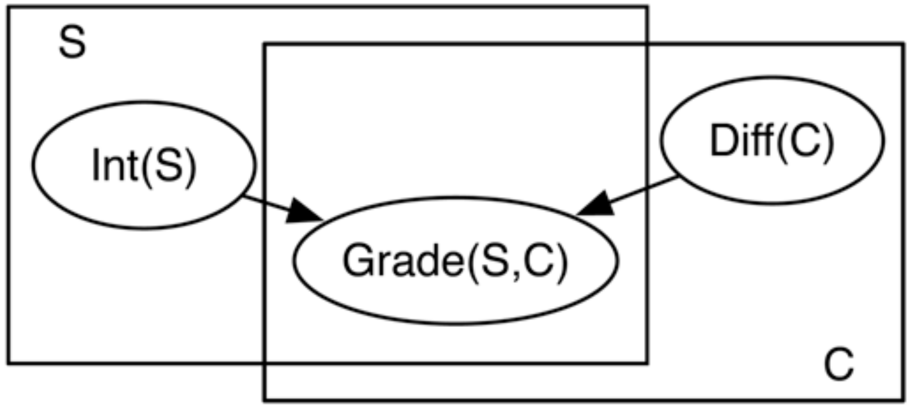
Bayesian net encoded as Probabilistic Logic Program
PLPs correspond to directed graphical models

LOGIC PROBABILITY

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

erc

Flexible and Compact Relational Model for Predicting Grades



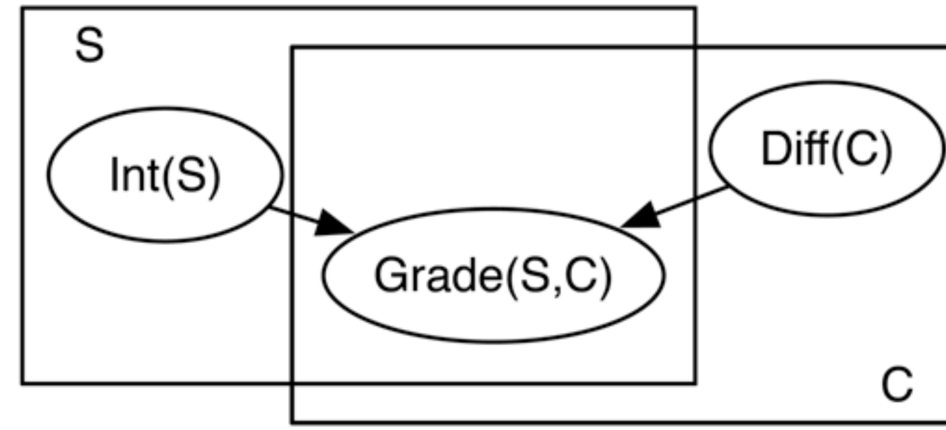
“Program” Abstraction:

- 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



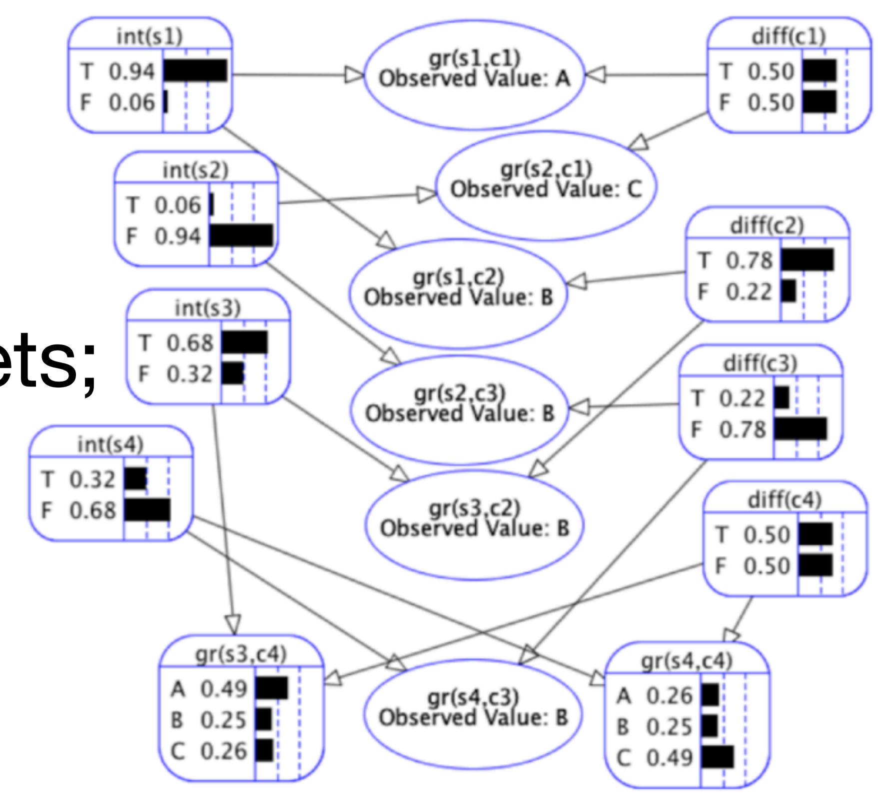
Shows relational structure

- grounded model: replace variables by constants

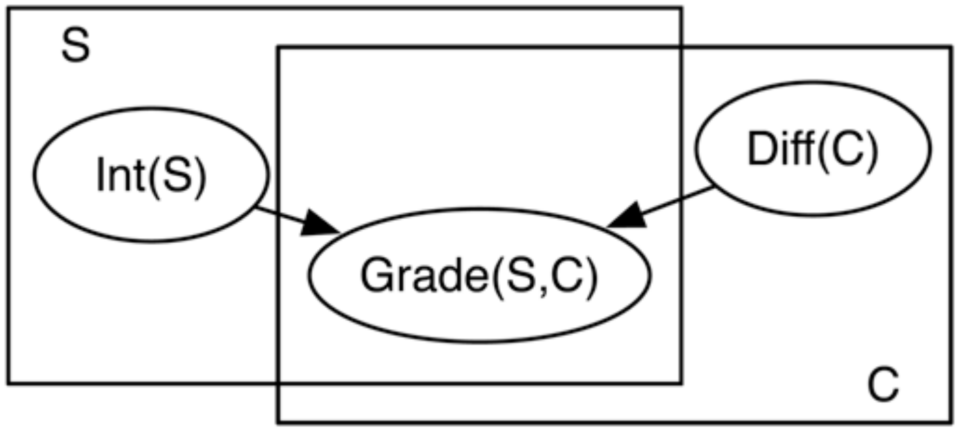
Works for any number of students / classes (for 1000 students and 100 classes, you get 101100 random variables); still only few parameters

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

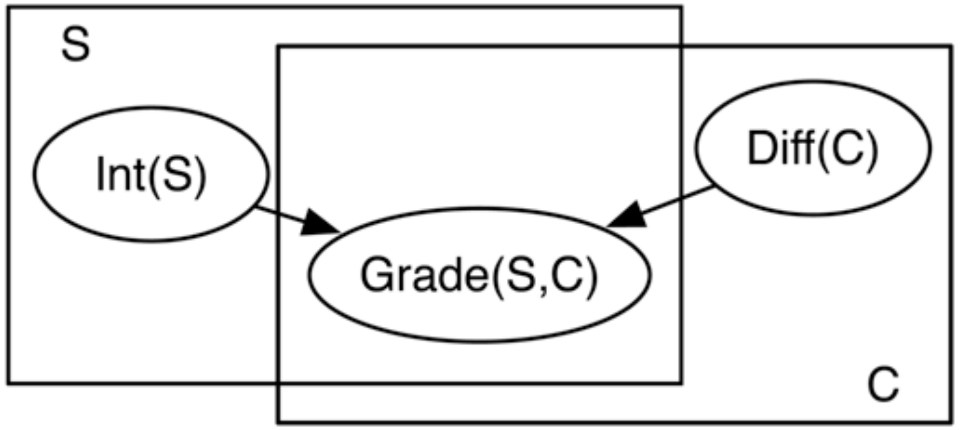
Works for any number of students / classes (for 1000 students and 100 classes, you get 101100 random variables); still only few parameters

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

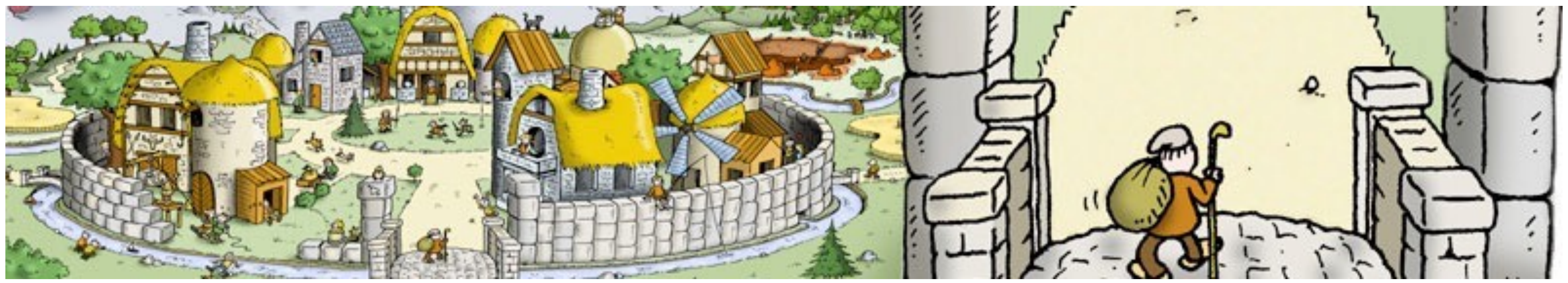



```
0.4 :: int(S) :- student(S).
0.5 :: diff(C):- course(C).

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



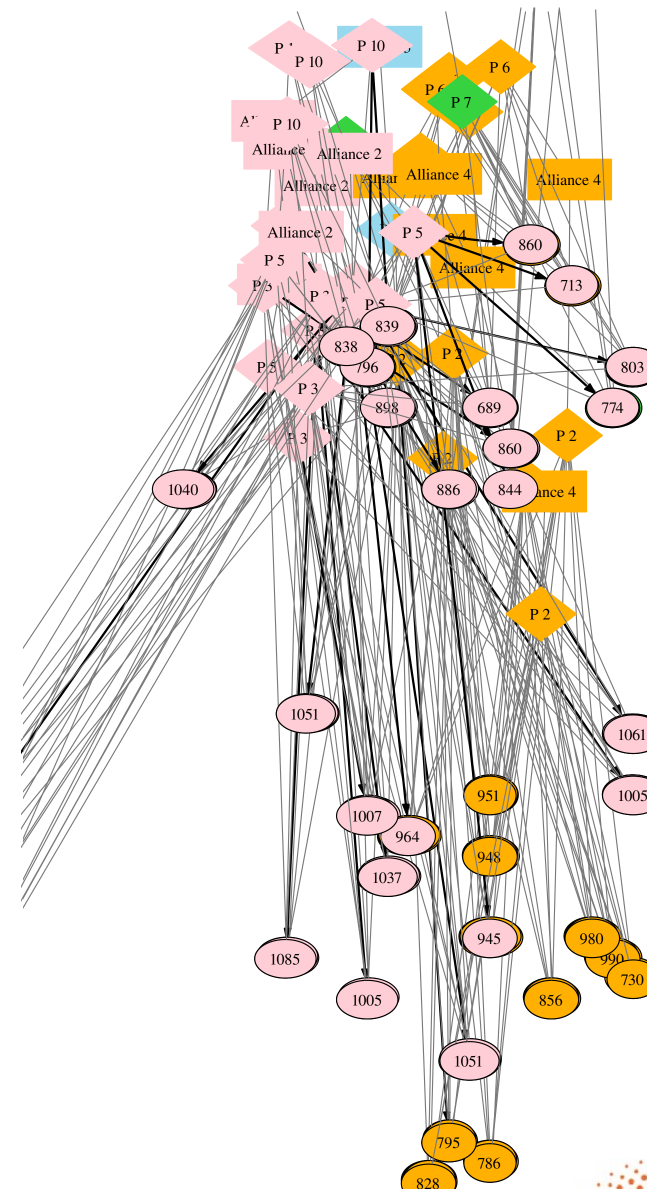
Travian: A massively multiplayer real-time strategy game

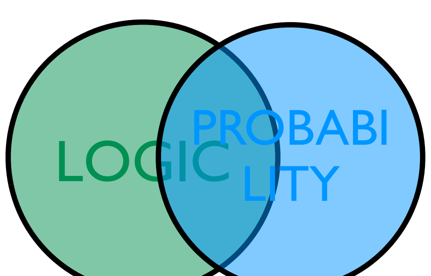
Can we build a model

of this world ?

Can we use it for playing

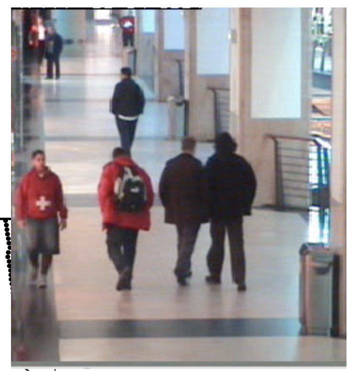
better ?

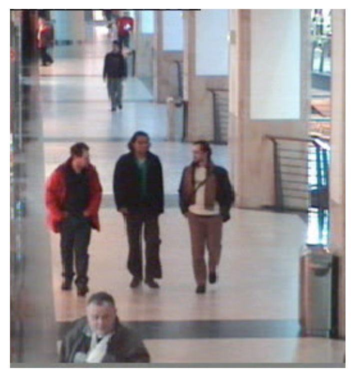


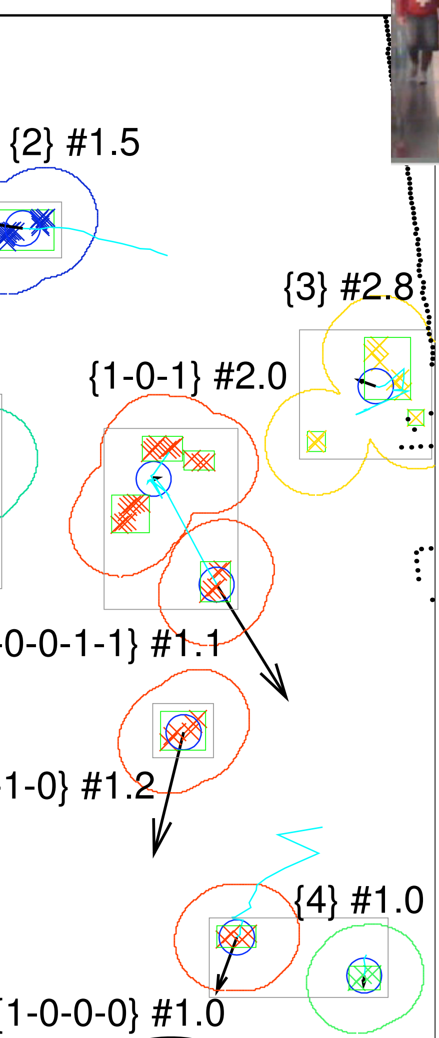


[Thon et al, MLJ 11]

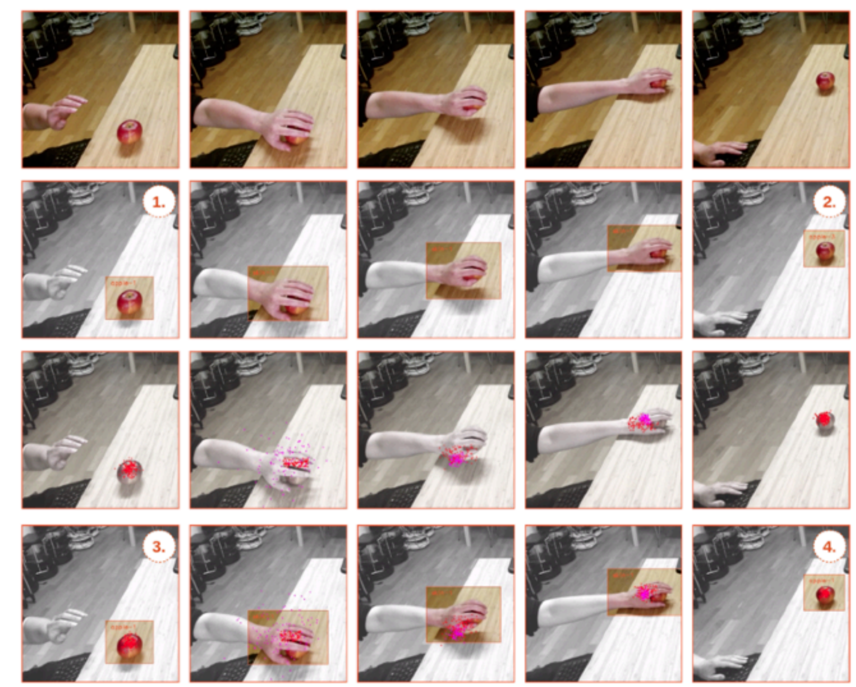
Activity analysis and tracking
video analysis





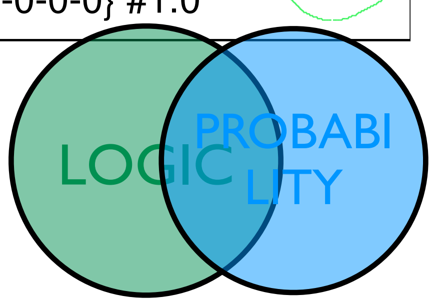


- Track people or objects over time? Even if temporarily hidden?
- Recognize activities?
- Infer object properties?



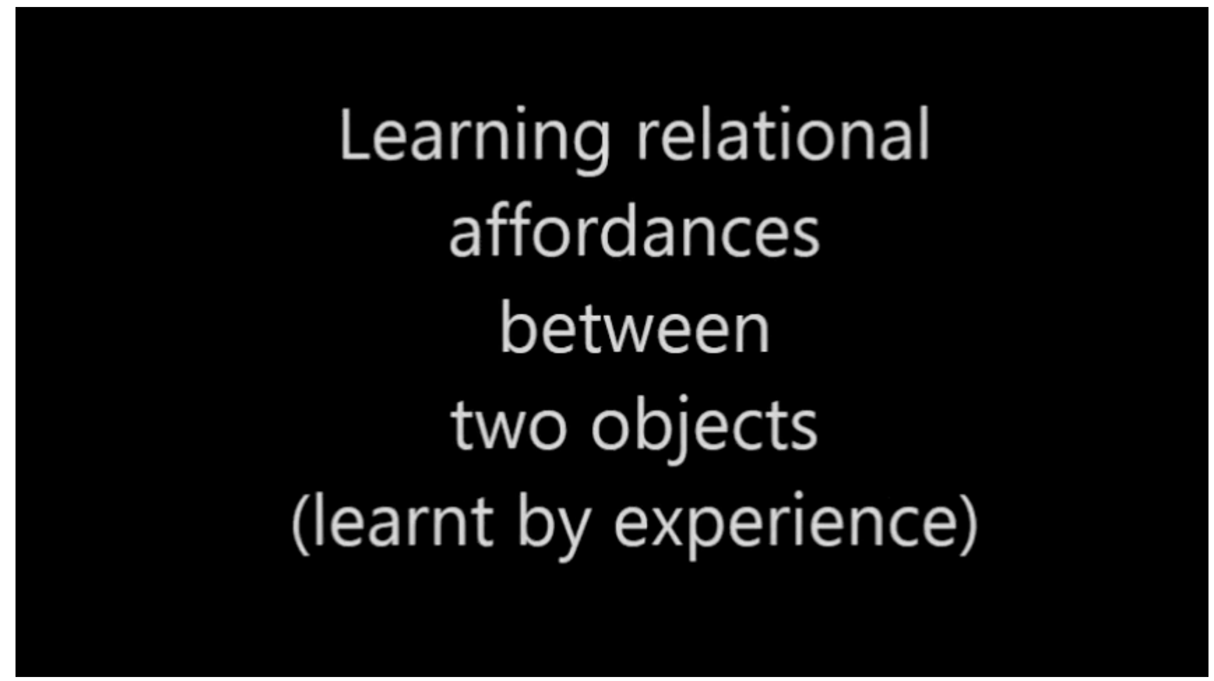
[Skarlatidis et al, TPLP 14; Nitti et al, IROS 13, ICRA 14, MLJ 16]

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



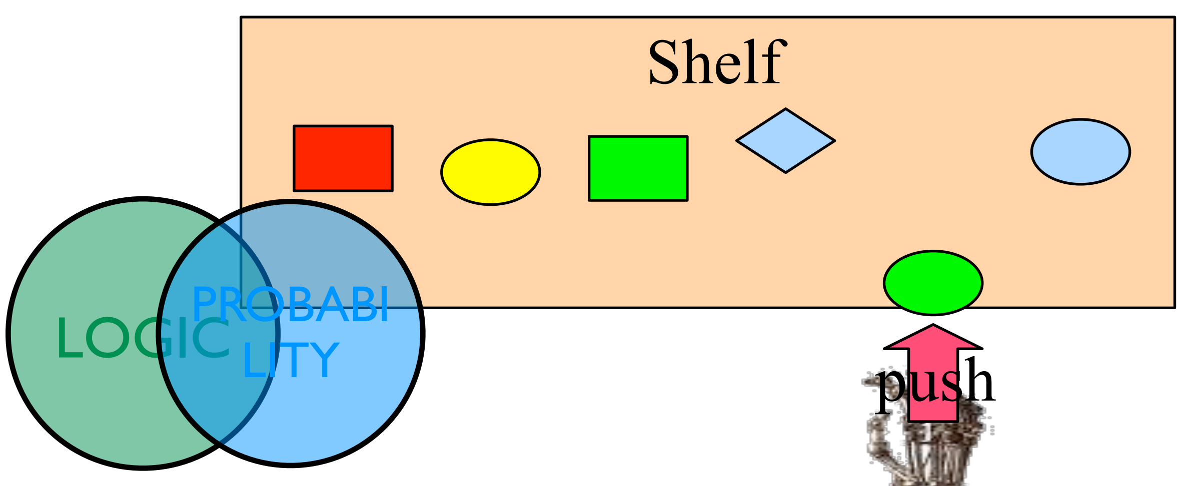
erc

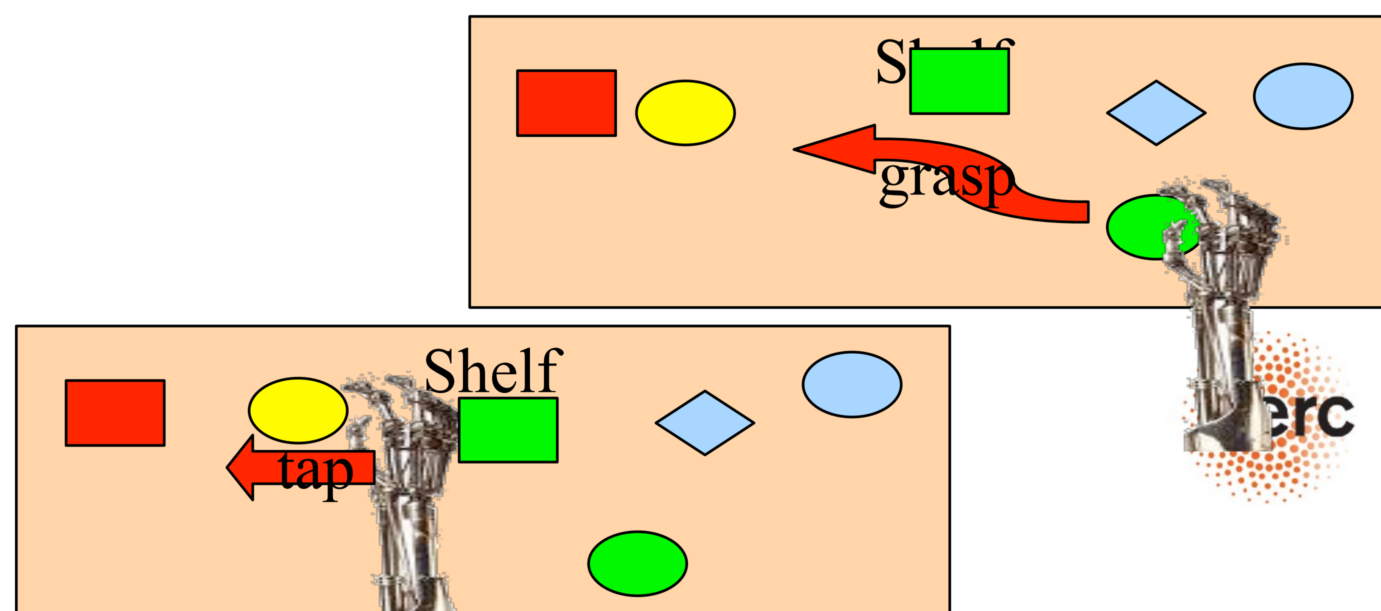
Learning relational affordances



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

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





Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

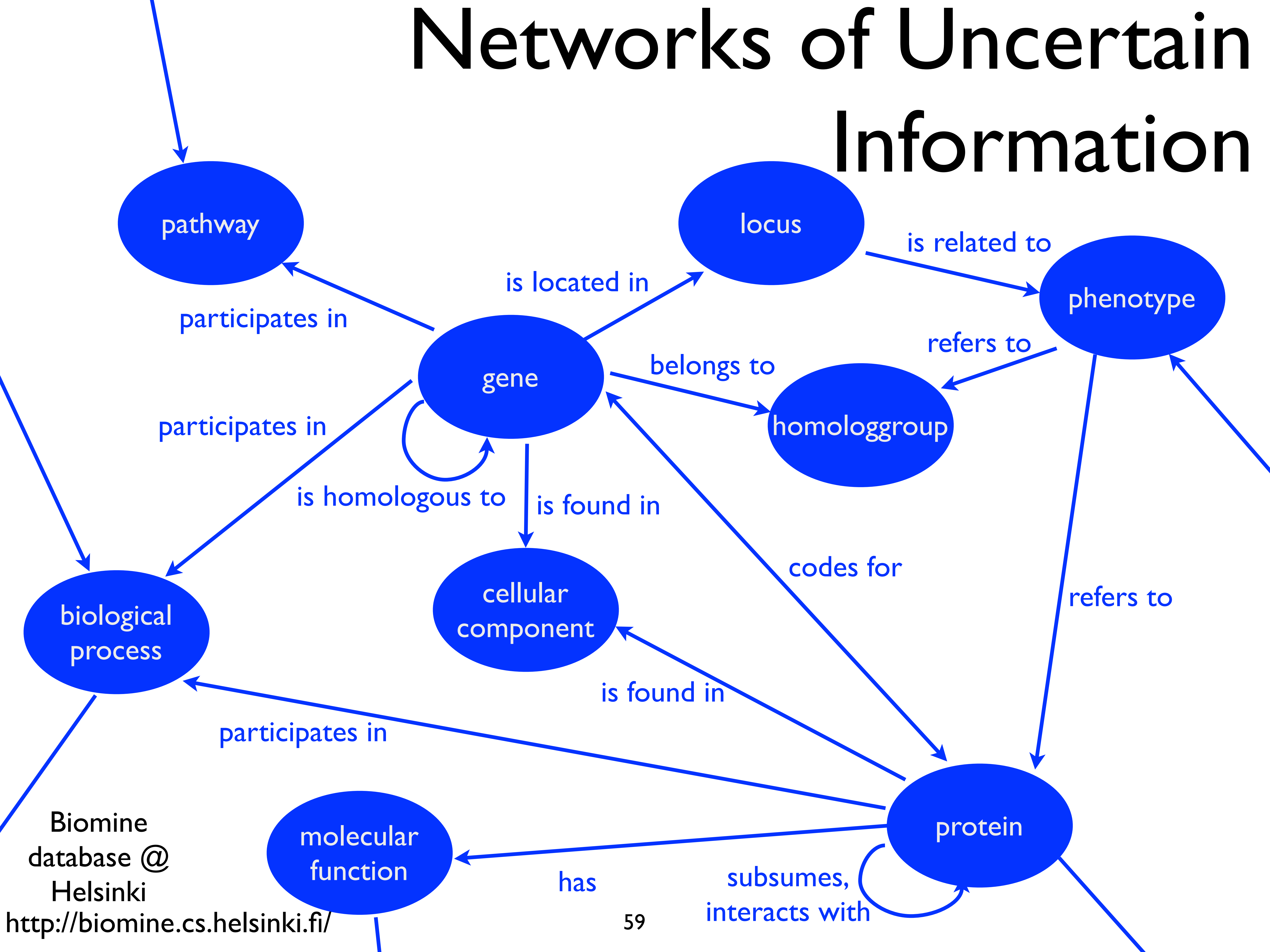
random variable with Gaussian distribution

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
```

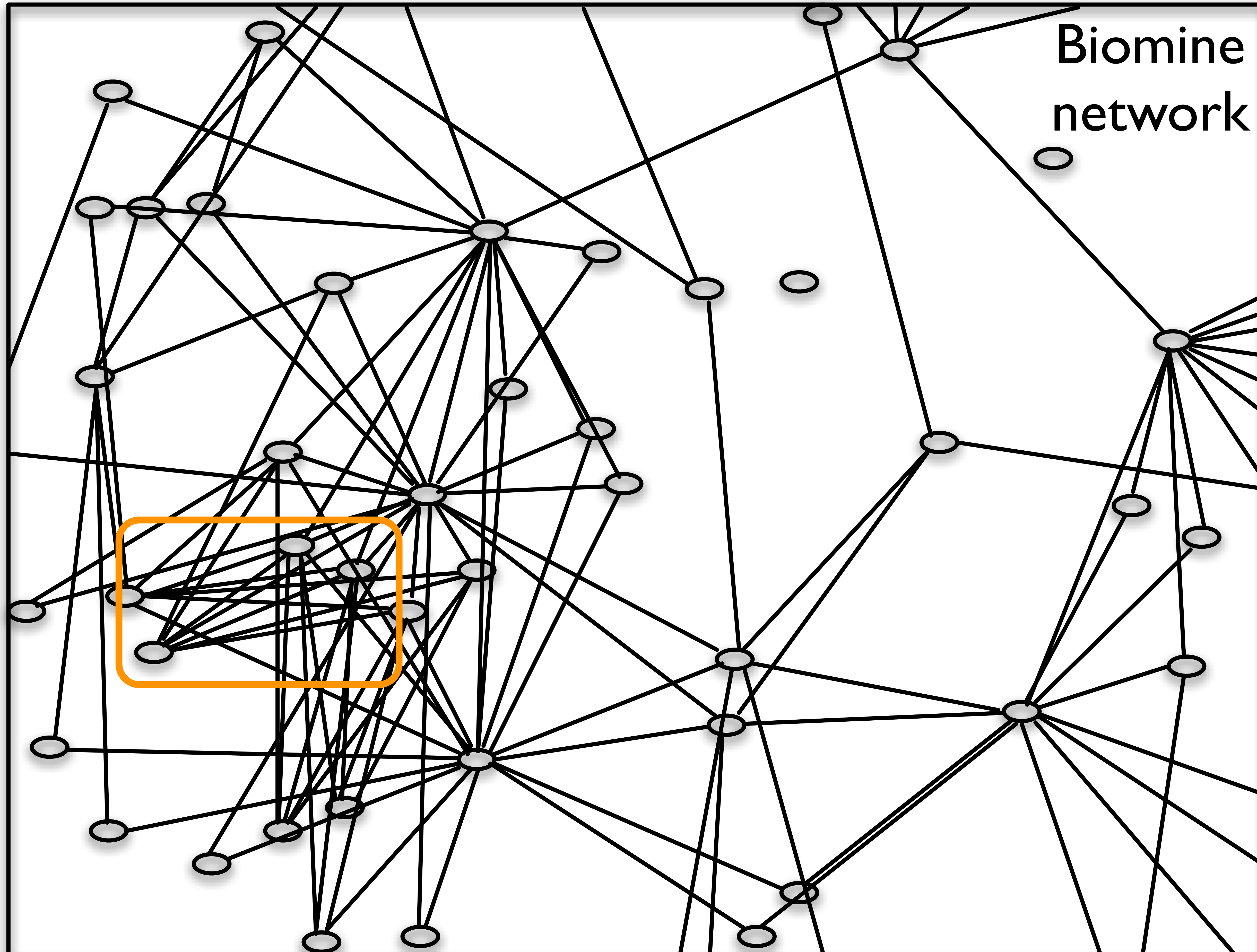
comparing values of random variables

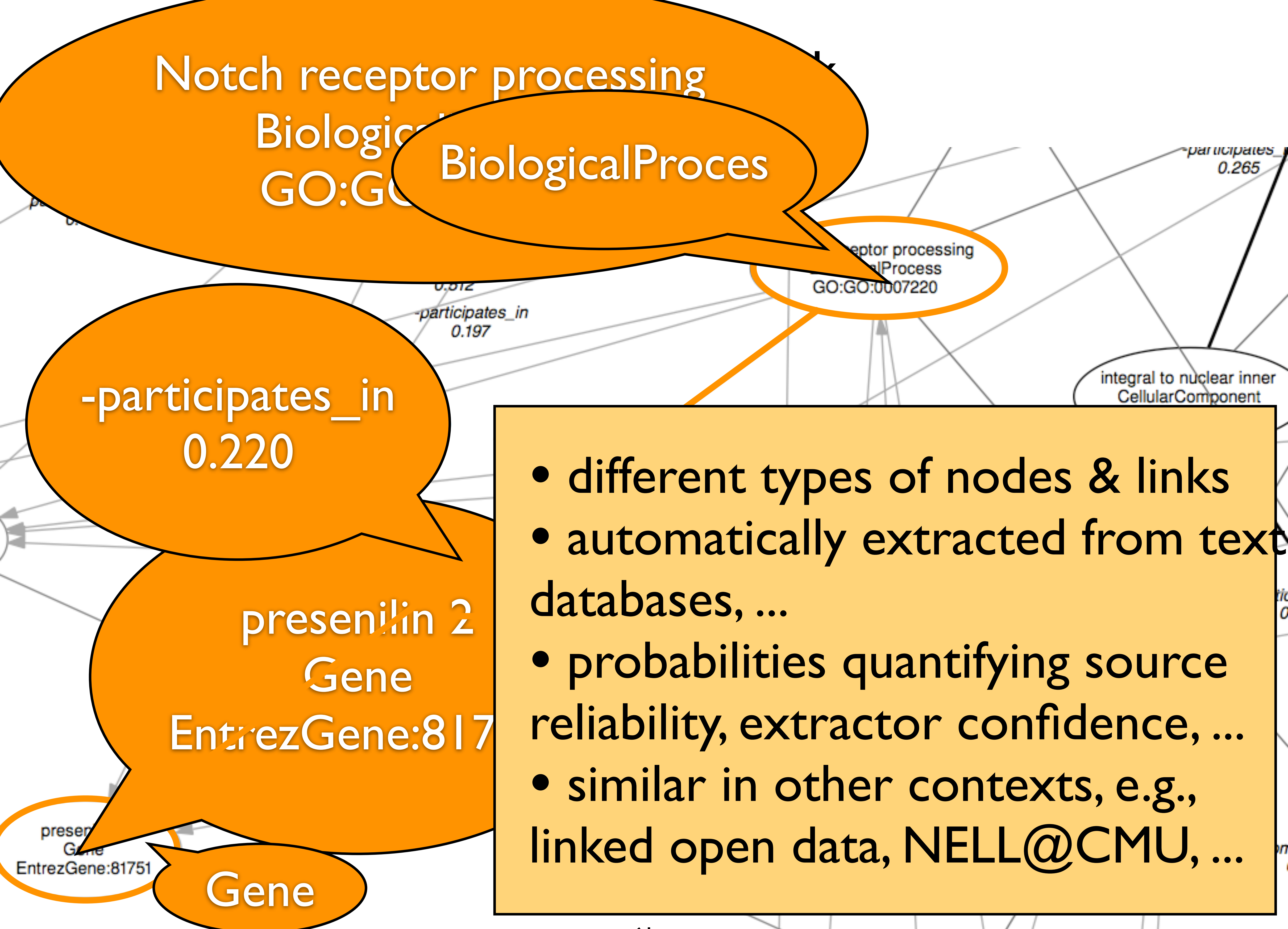
[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Networks of Uncertain Information



Biomine database @ Helsinki
http://biomine.cs.helsinki.fi/


Biomine
network



- different types of nodes & links
- automatically extracted from text, databases, ...
- probabilities quantifying source reliability, extractor confidence, ...
- similar in other contexts, e.g., linked open data, NELL@CMU, ...

Biology

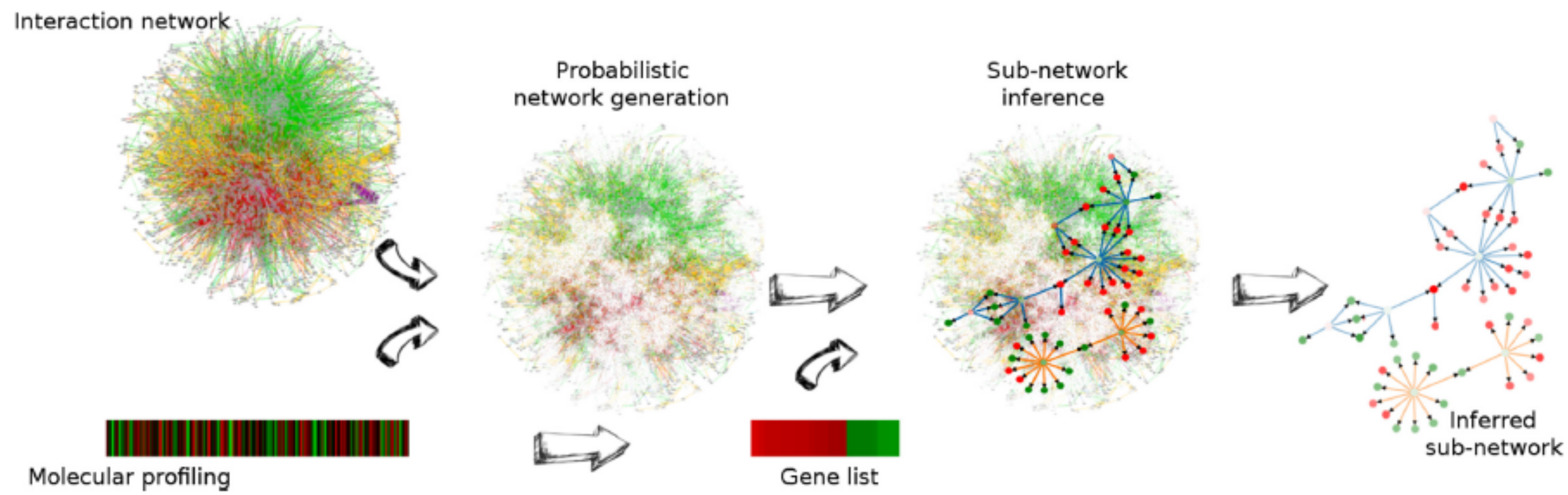


Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for 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.

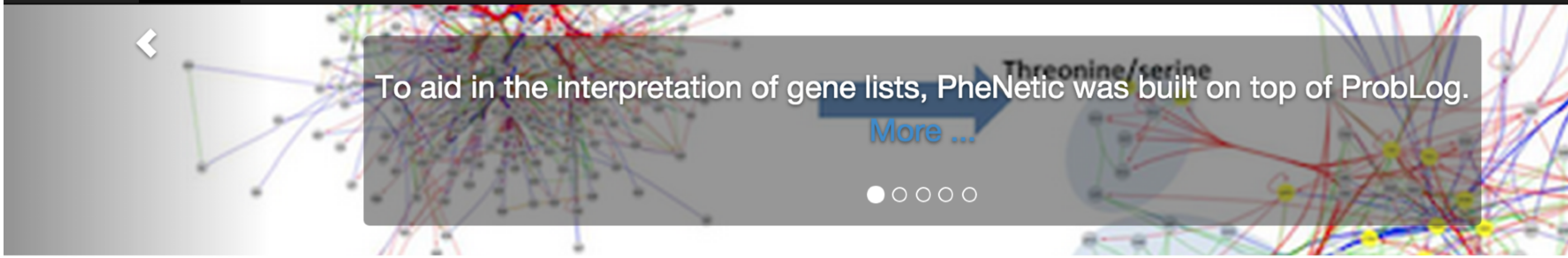
- Causes: Mutations
  - All related to similar phenotype
- Effects: Differentially expressed genes
- 27 000 cause effect pairs

- 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] https://dtai.cs.kuleuven.be/problog/

ProbLog | Home | Download | Publications | Help | People | Interactive



Introduction.

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

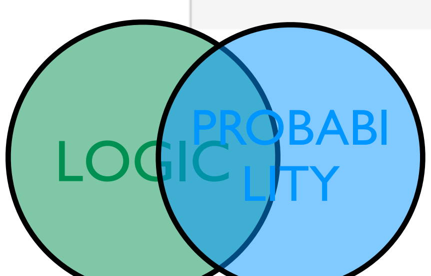
ProbLog is a tool that allows you to intuitively build programs that do not only encode **complex interactions** between a large sets of **heterogenous components** bu **uncertainties** that are present in real-life situations.

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).
```



Probabilistic Programming Languages outside LP

- IBAL [Pfeffer 01]
- Figaro [Pfeffer 09]
- Church [Goodman et al 08]
- BLOG [Milch et al 05]
- Stan & Edward & Anglican
- and many more appearing recently such

Church

probabilistic functional programming

[Goodman et al, UAI 08]

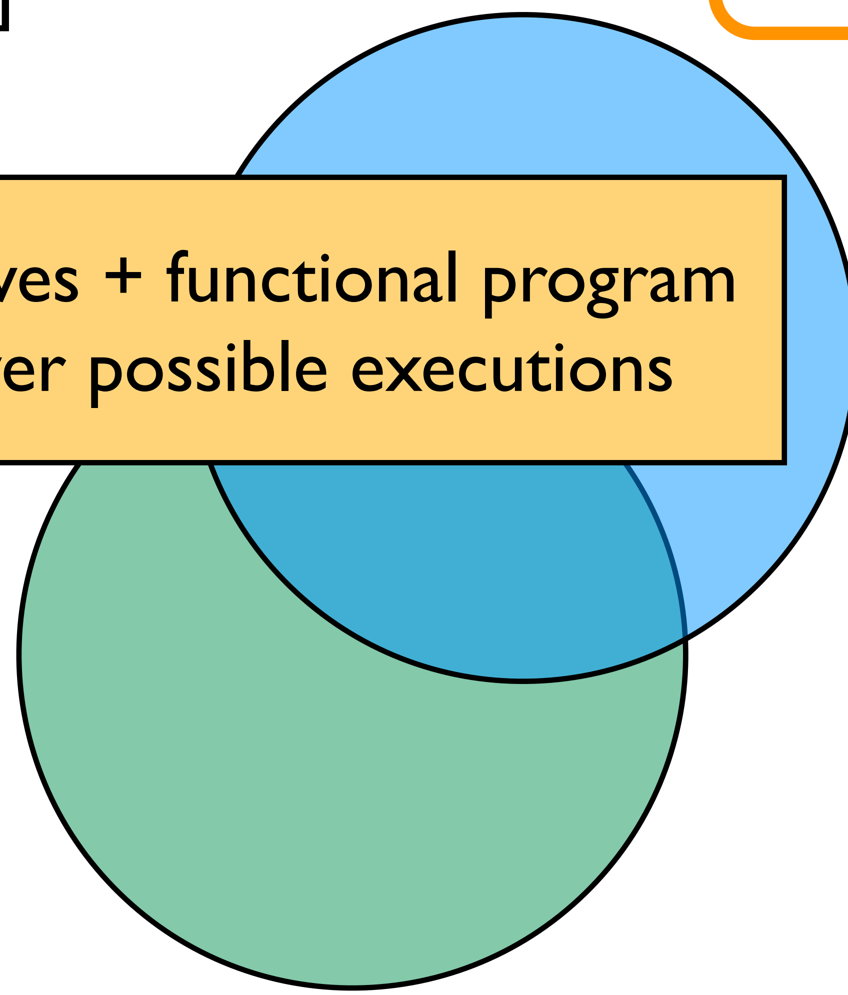
several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```

Dealing with uncertainty

random primitives

probabilistic primitives + functional program
→ distribution over possible executions



Reasoning with relational data

functional programming

Learning

one execution

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```


http://probmods.org

Church vs ProbLog

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))
```

Church result: (1 2) with 0.4×0.4
(1 7) with 0.4×0.6
(6 2) with 0.6×0.4
(6 7) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,2],_)).
```

ProbLog result: (1 2) with 0.4×0.4
(1 7) with 0.4×0.6
(6 2) with 0.6×0.4
(6 7) with 0.6×0.6

results for [1,1]?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

Church result: (1 1) with 0.4×0.4
(1 6) with 0.4×0.6
(6 1) with 0.6×0.4
(6 6) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,1],_)).
```

ProbLog result: (1 1) with 0.4
(1 6) with 0.0
(6 1) with 0.0
(6 6) with 0.6

stochastic memoization

Solution

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

```
0.4::p5(N,N,ID);0.6::p5(N,M,ID) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M,L),
    lp5(L,K).

query(lp5([1,1],_)).
```

identifier distinguishes calls

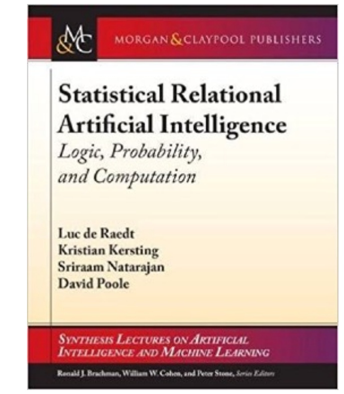
Stochastic Memoization

```
(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))

(map randplus5 '(1 1))
```

remember first value & reuse for all later calls

ProbLog always memoizes
PRISM never memoizes
Church allows fine-grained choice

2. Directed vs Undirected
the PGM / StarAI dimension



0.1 :: burglary.

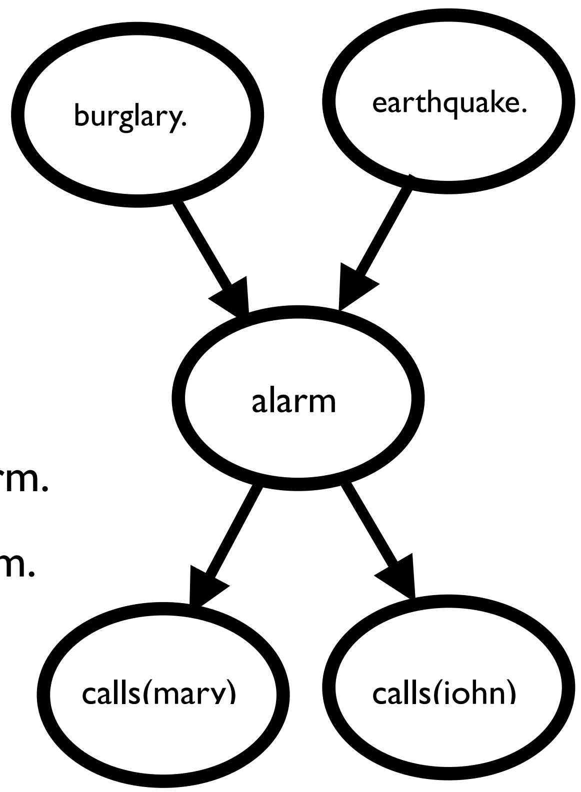
0.05 :: earthquake.

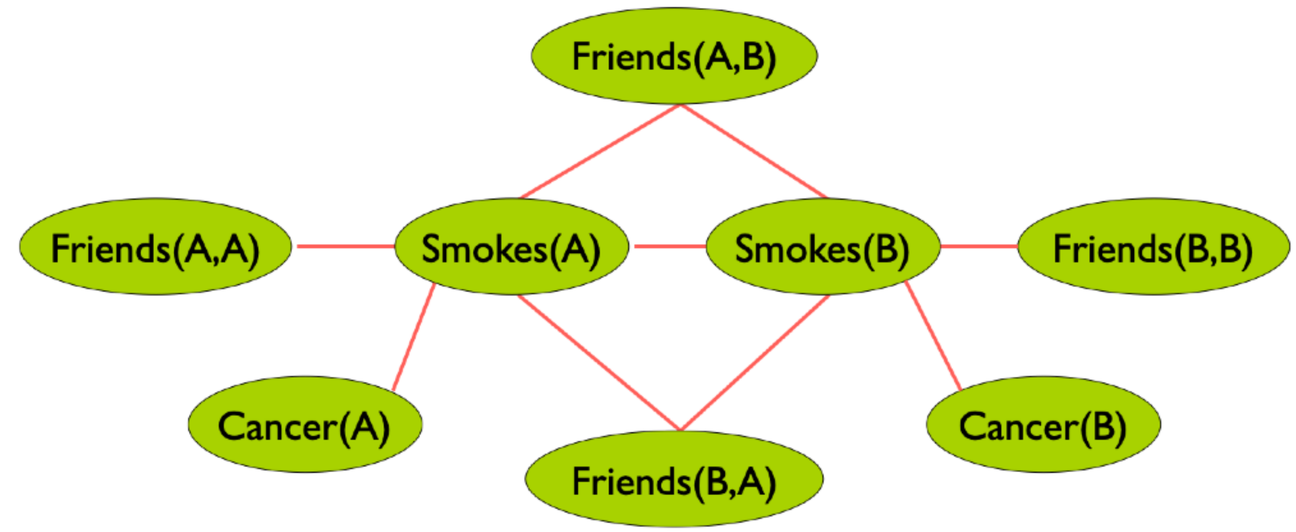
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.





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

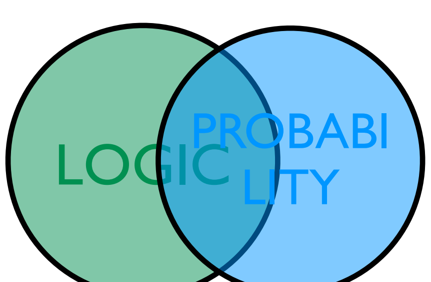
$$1.1 \quad \forall x,y\ Friends(x,y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

Markov Logic

undirected
Markov Net
model theoretic



key representatives

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**

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

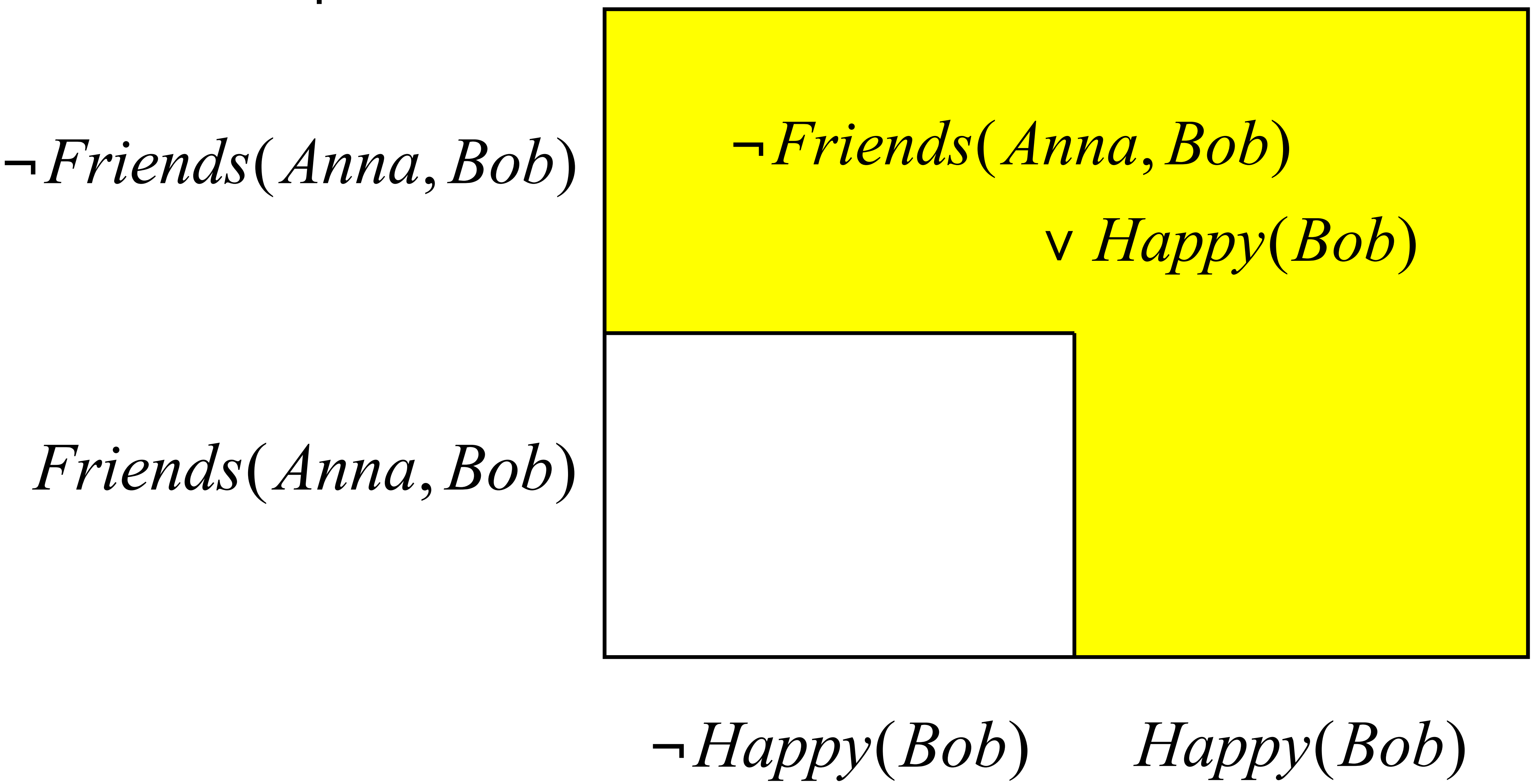
slides by Pedro Domingos

A possible worlds view

Logical formulas such as

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

exclude possible worlds



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:

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

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

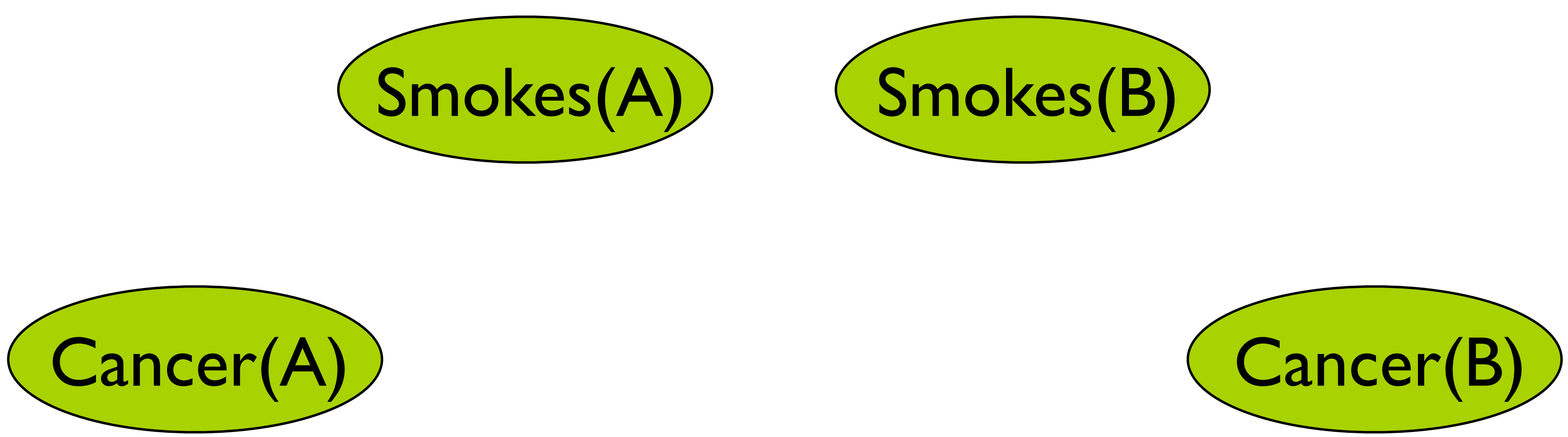
	$\neg Happy(Bob)$	$Happy(Bob)$
$\neg Friends(Anna, Bob)$	1	1
$Friends(Anna, Bob)$	0.75	1

This can also be viewed as building a graphical model

Markov Logic

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

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

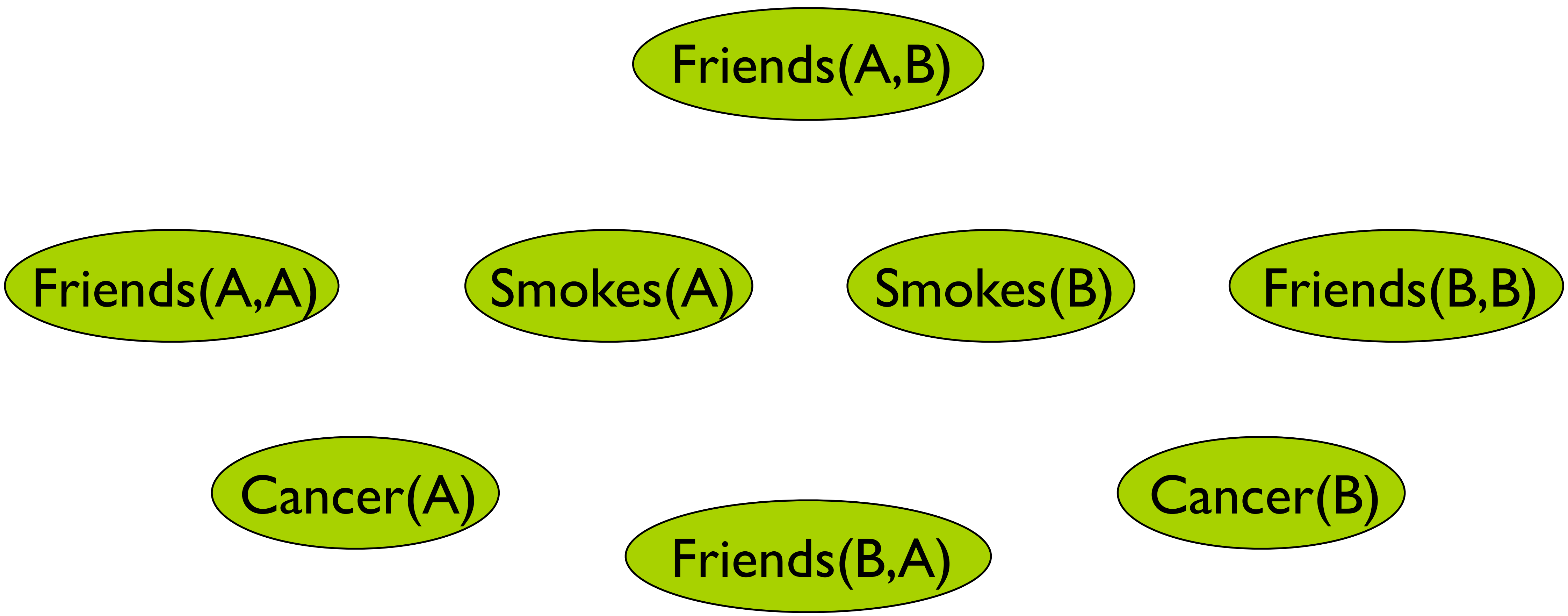


slides by Pedro Domingos

Markov Logic

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

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

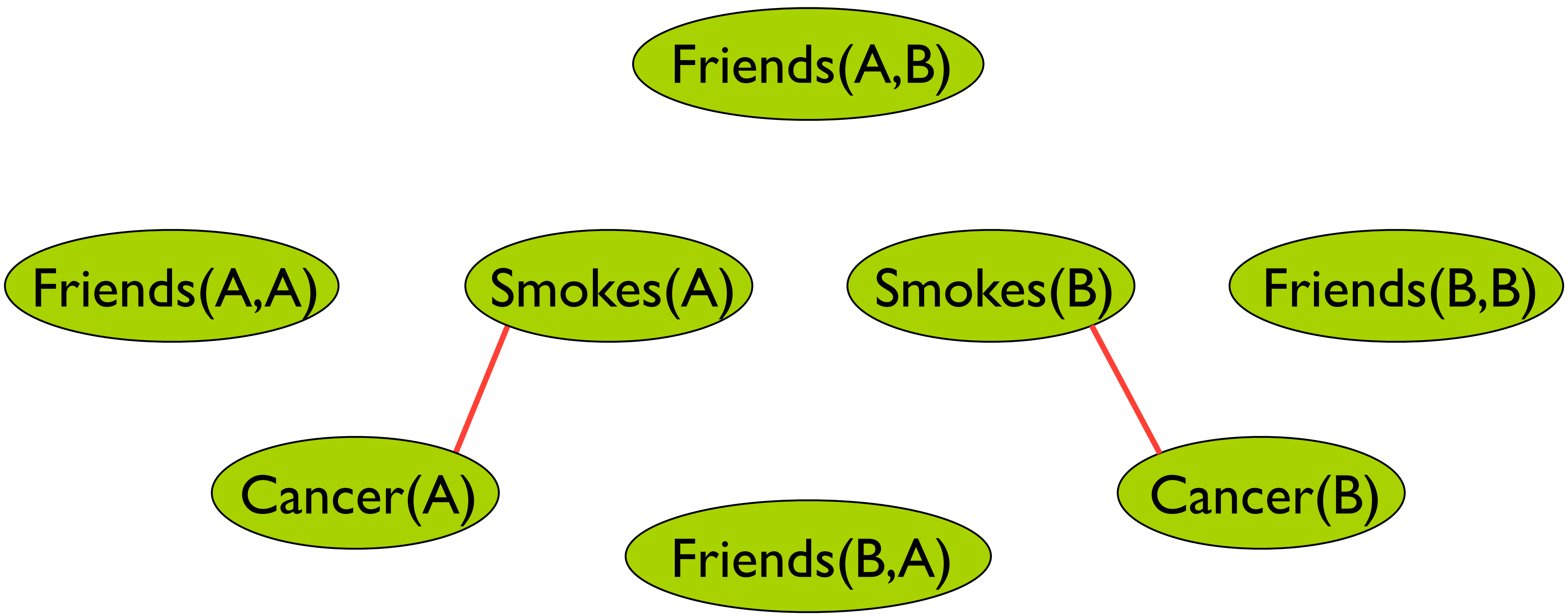


slides by Pedro Domingos

Markov Logic

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

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

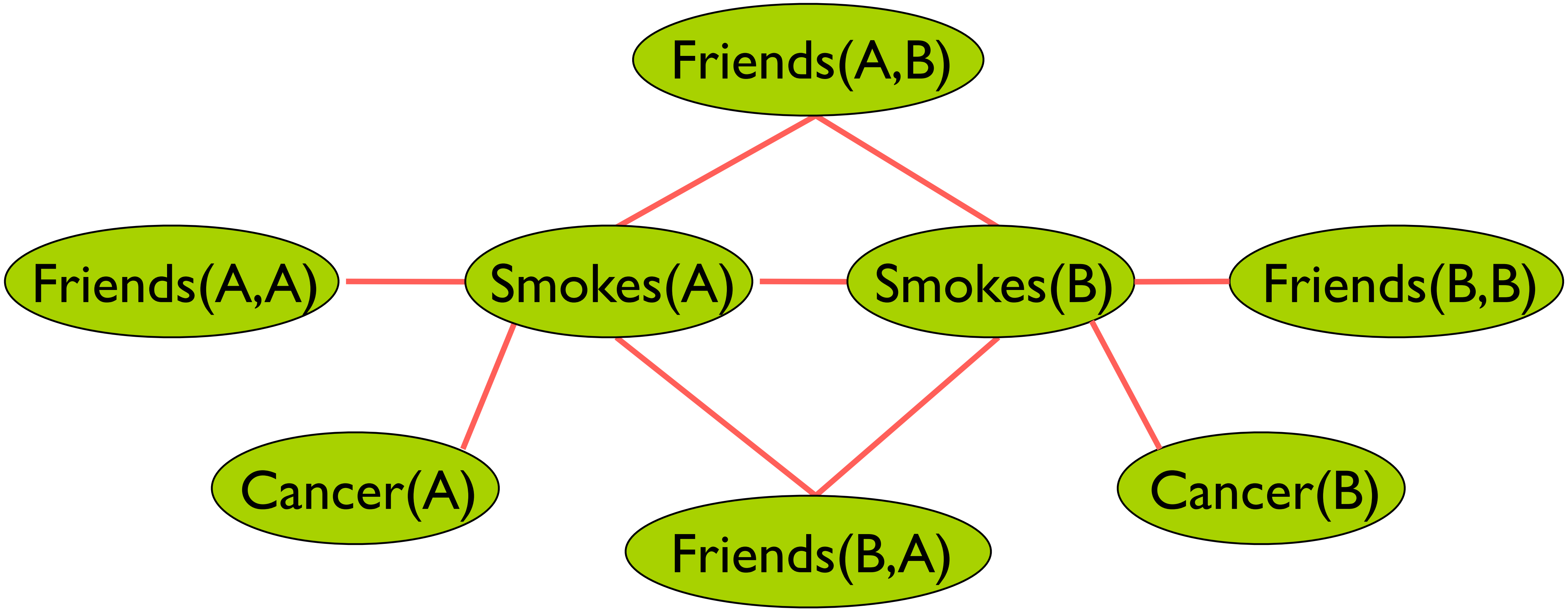


slides by Pedro Domingos

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

Markov Logic

- A Markov Logic Network (MLN) is a set of pairs (F, w) where
  - F is a formula in first-order logic
  - w is a real number
- An MLN defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w
- Probability of a world

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x) \right)$$

w_i: Weight of formula *i*

$n_i(x)$: No. of true groundings of formula *i* in *x*

Possible Worlds

A vocabulary

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)
0	0	0	0
⋮	⋮	⋮	⋮
1	0	1	0
⋮	⋮	⋮	⋮
1	1	1	1

Possible worlds
Logical interpretations

Slides adapted from Guy Van den Broeck

Possible Worlds

A logical theory

$$\forall x,y,\ \mathrm{Smokes}(x) \land \mathrm{Friends}(x,y) \Rightarrow \mathrm{Smokes}(y)$$

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	1
⋮	⋮	⋮	⋮	⋮
1	0	1	0	0
⋮	⋮	⋮	⋮	⋮
1	1	1	1	1

Interpretations that satisfy the theory
Models

Slides adapted from Guy Van den Broeck

First-Order Model Counting

A logical theory

$\forall x,y,\ Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y)$

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	1
⋮	⋮	⋮	⋮	⋮
1	0	1	0	0
⋮	⋮	⋮	⋮	⋮
1	1	1	1	1

Σ First-order model count ~#SAT

Slides Guy Van den Broeck

Markov Logic

- A Markov Logic Network (MLN) is a set of pairs (F, w) where
  - F is a formula in first-order logic
  - w is a real number
- An MLN defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w
- Probability of a world

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x) \right)$$

Weight of formula *i* (w_i)

No. of true groundings of formula *i* in *x* ($n_i(x)$)

Parameter Learning

$$\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w\left[n_i(x)\right]$$

No. of times clause *i* is true in data

Expected no. times clause *i* is true according to MLN

Has been used for generative learning (Pseudolikelihood);
Many variations (also discriminative);
applications in networks, NLP, bioinformatics, …

Markov Logic

A Markov Logic theory

1.5 ∀x,y, Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	$\frac{1}{Z} exp(1.5 * 2)$
⋮	⋮	⋮	⋮	
1	0	1	0	$\frac{1}{Z} exp(1.5 * 1)$
⋮	⋮	⋮	⋮	
1	1	1	1	$\frac{1}{Z} exp(1.5 * 2)$

counting only substitutions for which X =/= Y
X=Alice, Y=Bob
X=Bob, Y=Alice

Slides adapted from Guy Van den Broeck

Markov Logic

A Markov Logic theory

1.5 $\forall x,y$, Smokes(x) $\wedge$ Friends(x,y) $\Rightarrow$ Smokes(y)

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	$\frac{1}{Z} exp(1.5 * 2)$
⋮	⋮	⋮	⋮	
1	0	1	0	$\frac{1}{Z} exp(1.5 * 1)$
⋮	⋮	⋮	⋮	
1	1	1	1	$\frac{1}{Z} exp(1.5 * 2)$

Σ

Z
partition function

Slides adapted from Guy Van den Broeck

Weighted First-Order Model Counting

A logical theory and a weight function for predicates

Smokes → 1
¬Smokes → 2
Friends → 4
¬Friends → 1

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory	weight
0	0	0	0	1	$2 \cdot 2 \cdot 1 \cdot 1$
⋮	⋮	⋮	⋮	⋮	⋮
1	0	1	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	1	$1 \cdot 1 \cdot 4 \cdot 4$

$\sum$ Weighted first-order model count

Related to ProbLog Inference !

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:

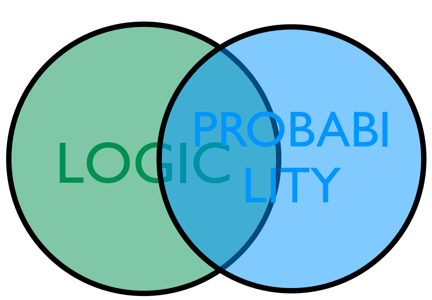
Alchemy Lite

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.

Why StarAI ?

- Reasoning (Probability + Logic) AND Learning
- SRL : Expressive Probabilistic Graphical Models
  - First order logic results supports entities + relationships + background knowledge — abstraction of multiple entities
  - Recursion (e.g. smokers cannot be represented by a plate model)
- PP : Power of a universal Turing machine = a prog. language
  - you can program in it and have builtin expressive prob. models
  - PP can learn -> so bring learning to programming languages
- ProbLog fits both paradigms

Inference



Inference / Reasoning

- Most of the work in PP and StarAI is on inference
  - It is hard (complexity wise)
  - Many inference methods
    - exact, approximate, sampling and lifted …
- Inference is the key to learning

Two Steps

- **Logical inference** -
  - about a ground logical theory
    - proofs or model theoretic …
  - *Result: Weighted Model Counting problem*
- **Probabilistic propositional inference** —
  - Knowledge Compilation
  - Backtracking search — DPLL, VE, RC based
- **Advanced** — lifted inference

ProbLog Inference

Answering a query in a ProbLog program happens in four steps

1. Grounding the program w.r.t. the query
2. **Rewrite the ground logic program into a propositional logic formula**
3. Compile the formula into an arithmetic circuit
4. Evaluate the arithmetic circuit

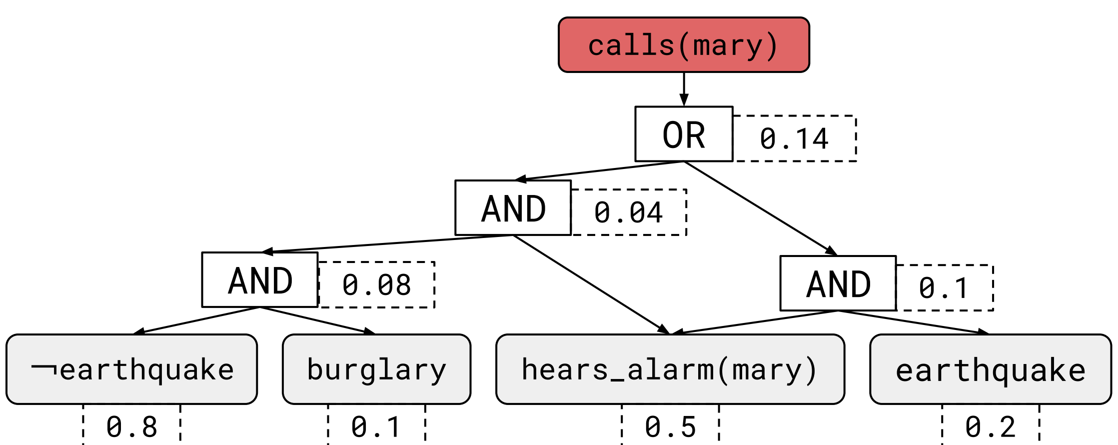
```
0.1 :: burglary.
0.5 :: hears_alarm(mary).

0.2 :: earthquake.
0.4 :: hears_alarm(john).

alarm :– earthquake.

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

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

$$\text{calls(mary)} \leftrightarrow \text{hears\_alarm(mary)} \wedge (\text{burglary} \vee \text{earthquake})$$

ProbLog Inference

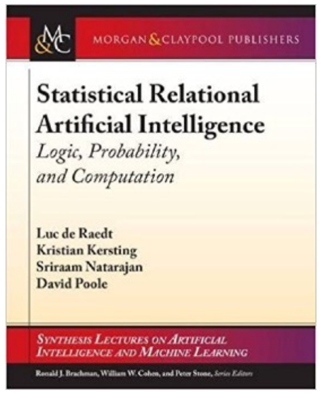
Answering a query in a ProbLog program happens in four steps

1. Grounding the program w.r.t. the query
2. Rewrite the ground logic program into a propositional logic formula
3. **Compile the formula into an arithmetic circuit (knowledge compilation)**
4. Evaluate the arithmetic circuit



$$\text{calls(mary)} \leftrightarrow \text{hears\_alarm(mary)} \land (\text{burglary} \lor \text{earthquake})$$

2. Directed vs Undirected the PGM / StarAI dimension



0.1 :: burglary.

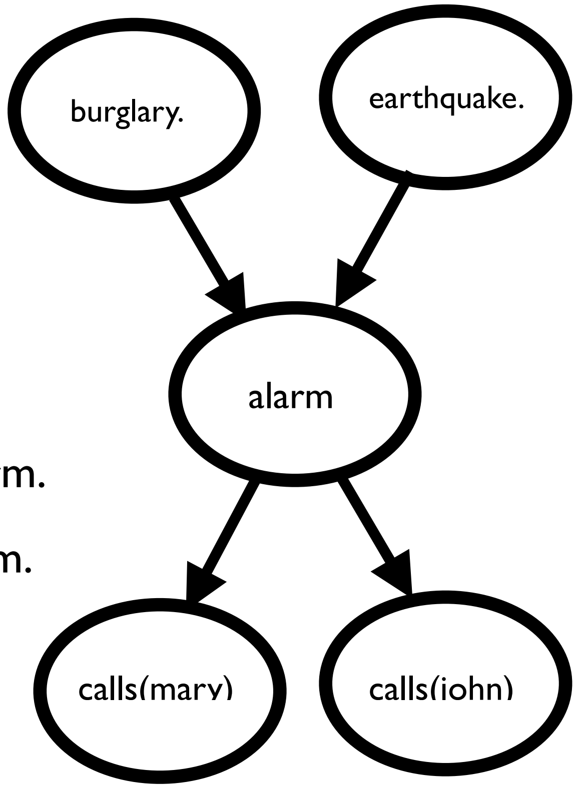
0.05 :: earthquake.

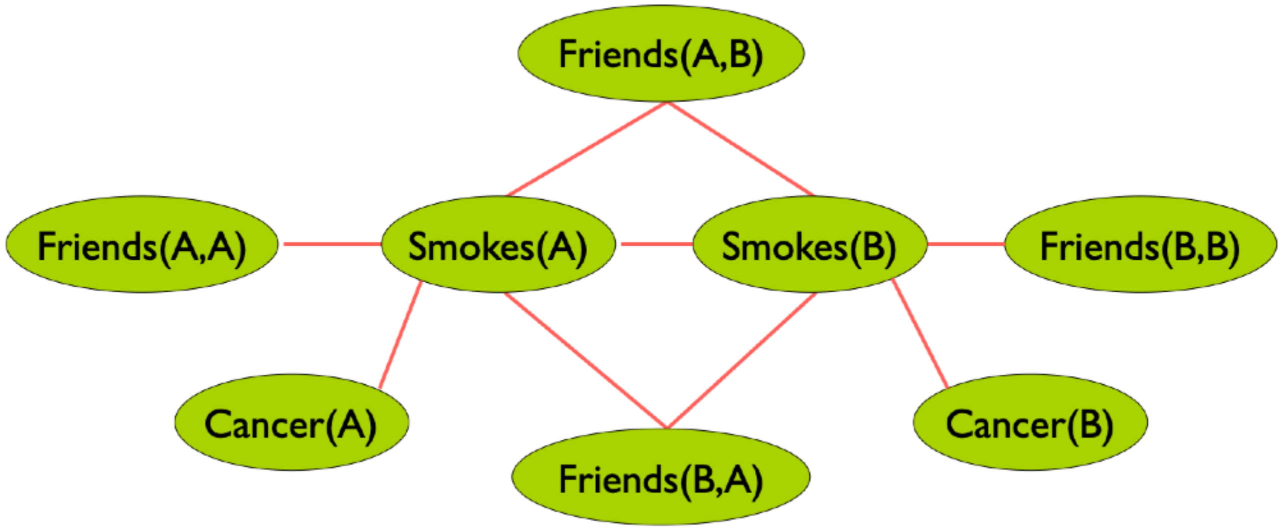
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.





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

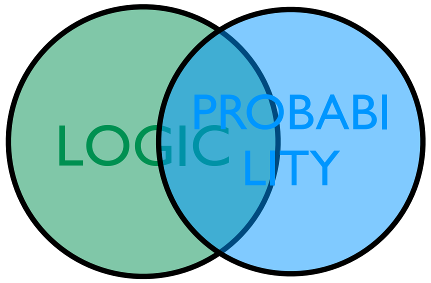
$$1.1 \quad \forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

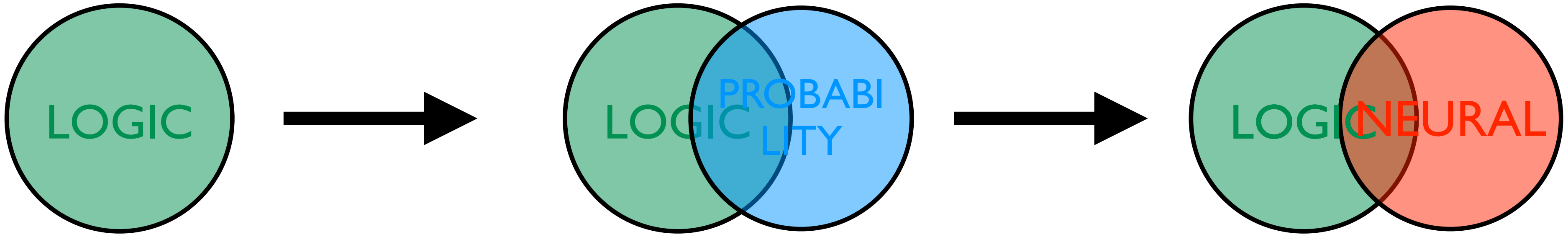
Markov Logic

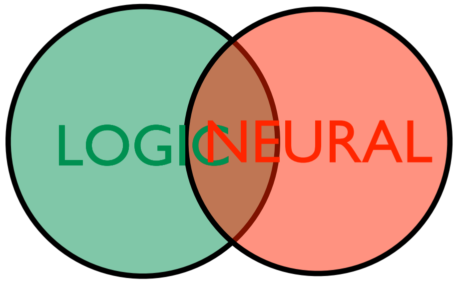
undirected
Markov Net
model theoretic



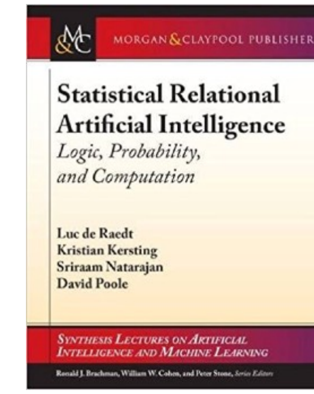
key representatives

1. Proof vs Model based
2. Directed vs Undirected





2. Directed vs Undirected the NeSy dimension



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

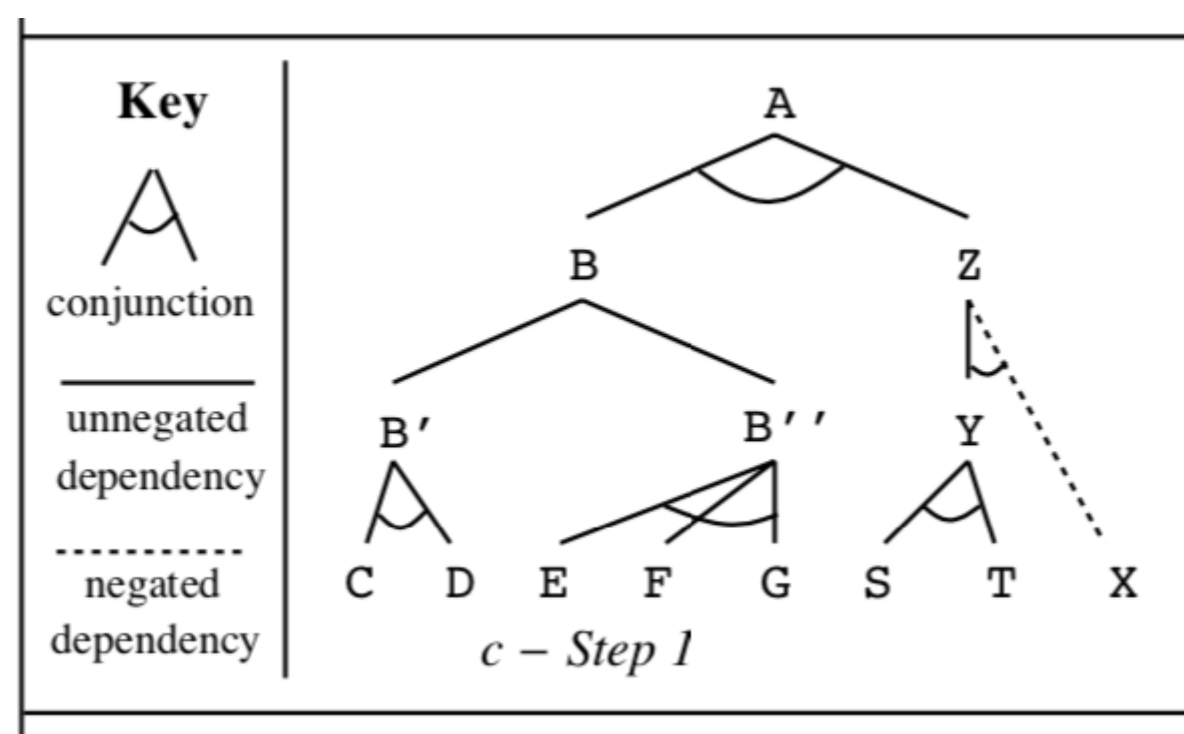
- KBANN (Towell and Shavlik AIJ 94)
- Turn a (propositional) Prolog program into a neural network and learn

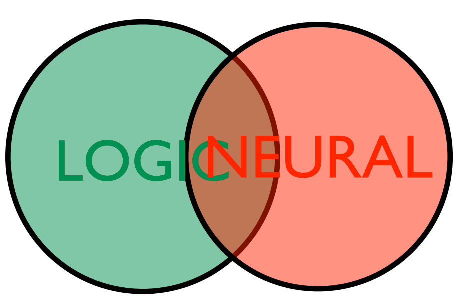
```
A :- B, Z.
B :- C, D.
B :- E, F, G.
Z :- Y, not X.
Y :- S, T.
```

REWRITE →

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

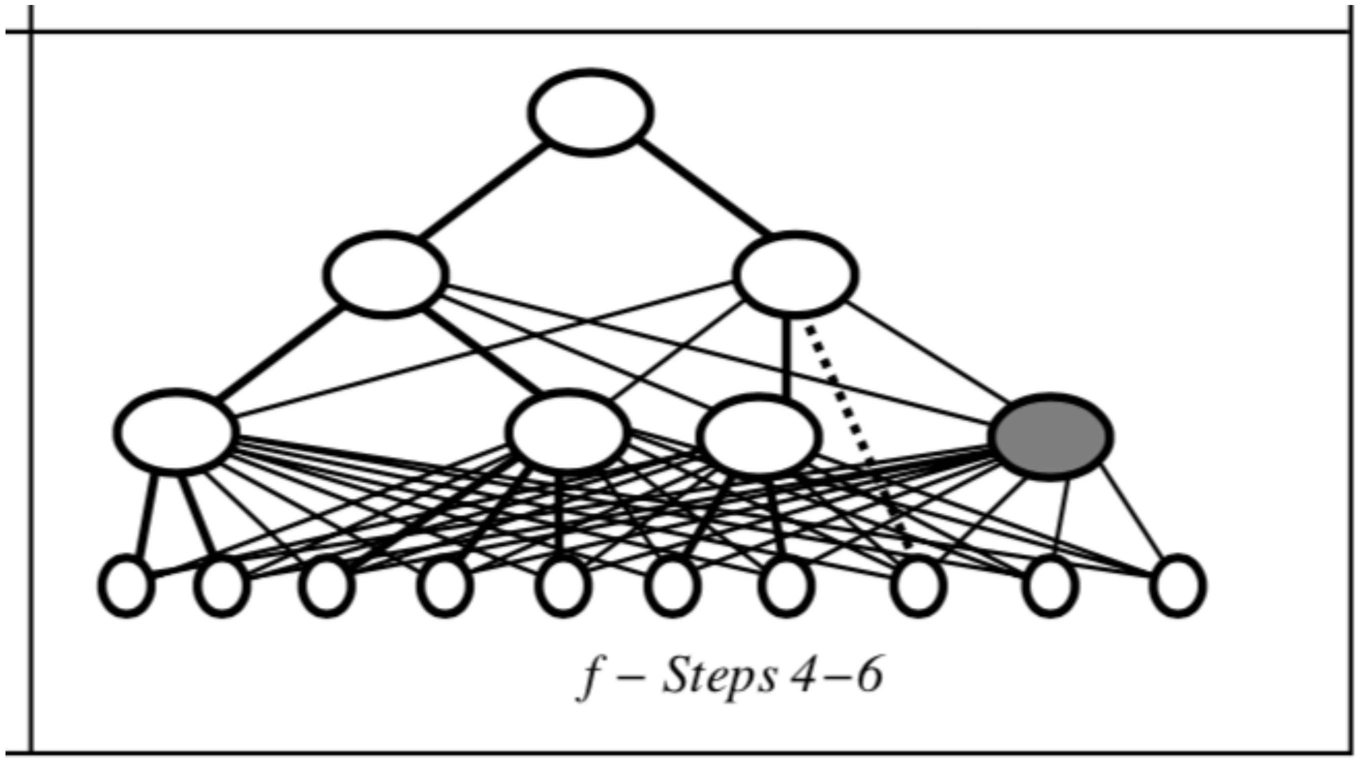
→

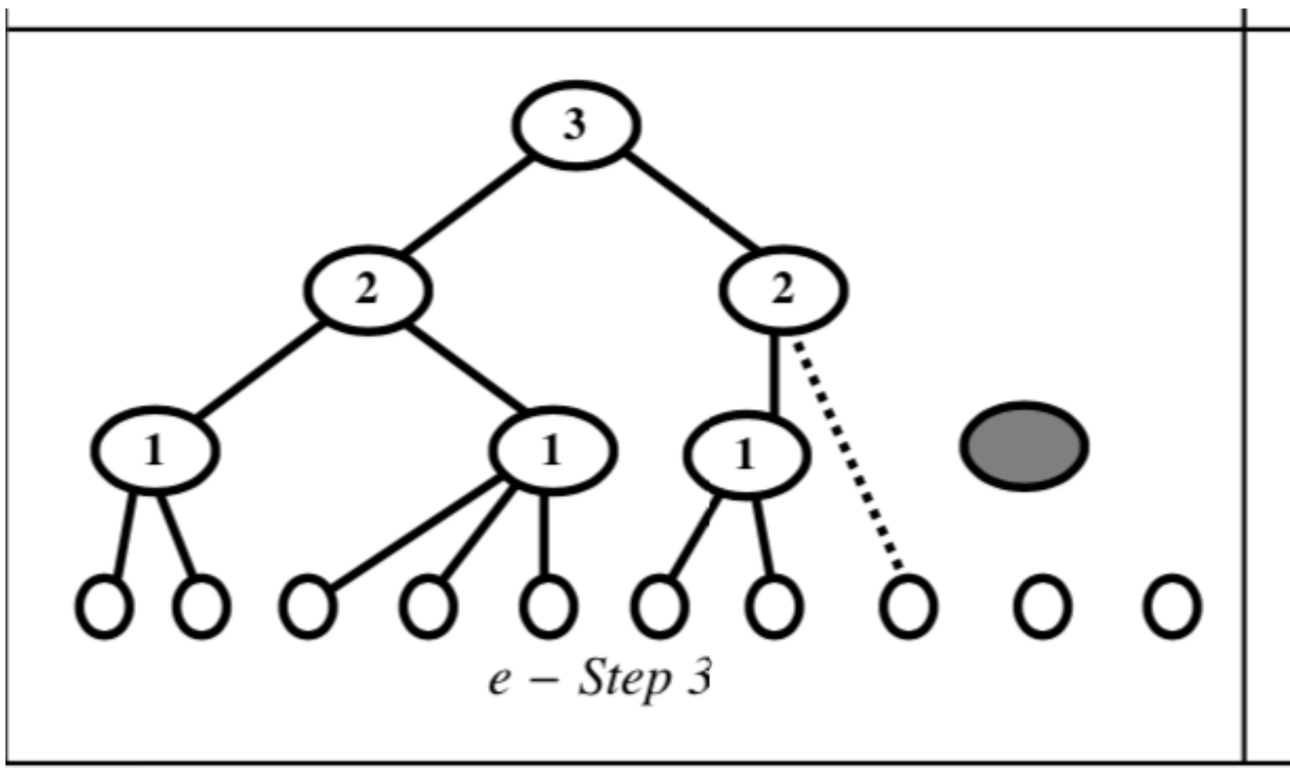




Logic as a neural program

directed StarAI approach and logic programs





ADD LINKS — ALSO SPURIOUS ONES

HIDDEN UNIT

and then learn

(Details of activation & loss functions not mentioned)

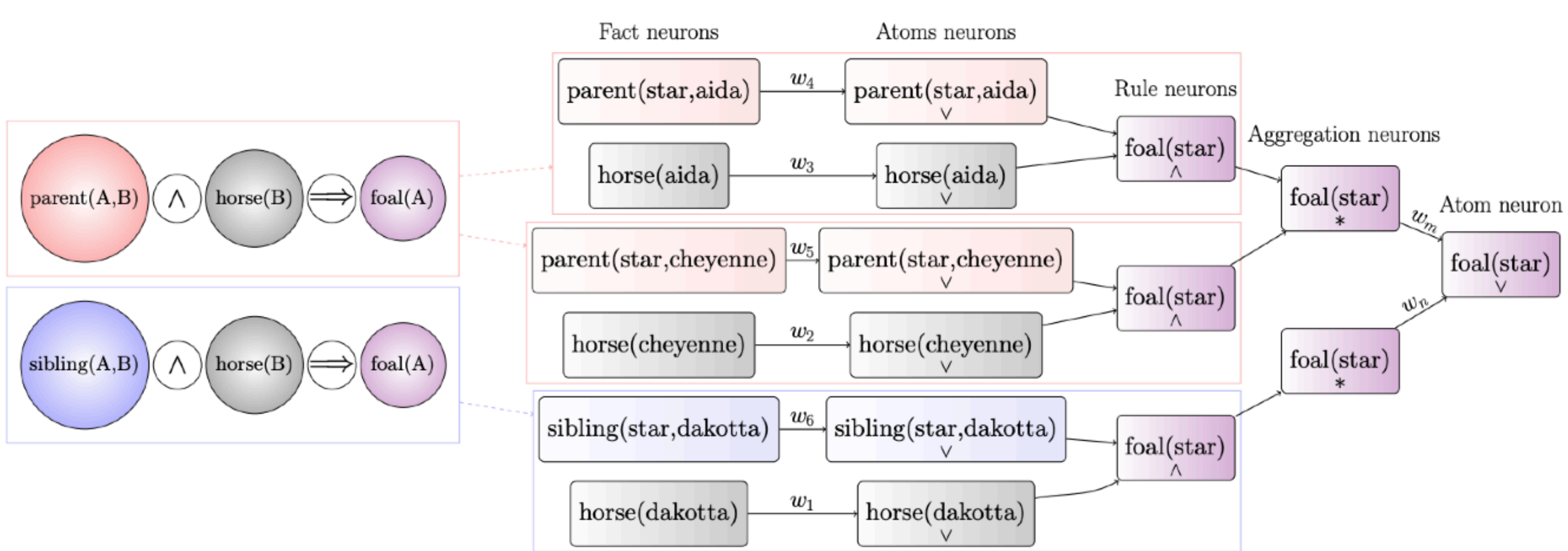
LOGIC NEURAL

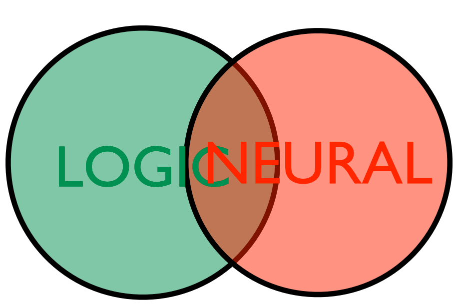
erc

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)

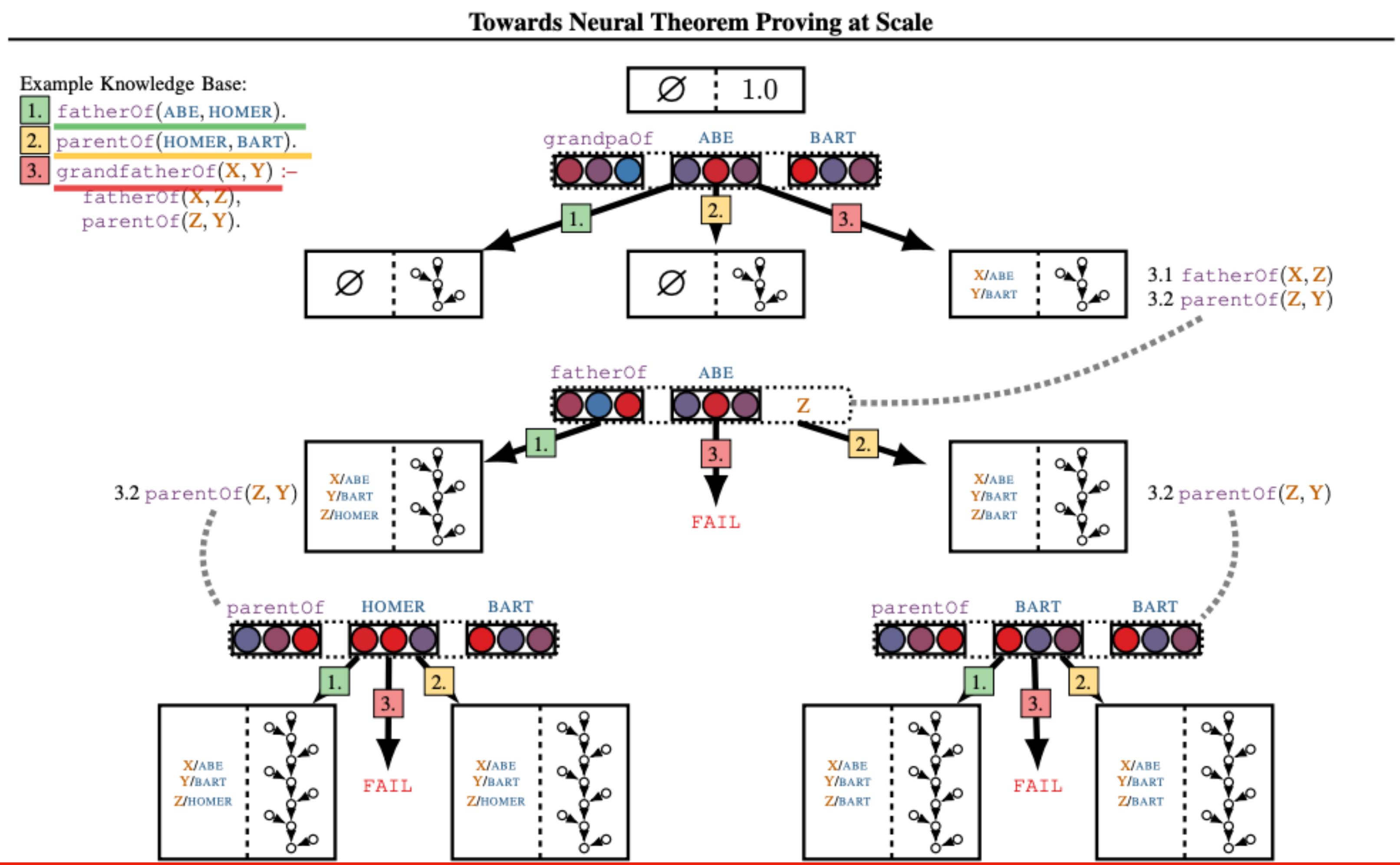




[Sourek, Kuzelka, et al JAIR]

Neural Theorem Prover

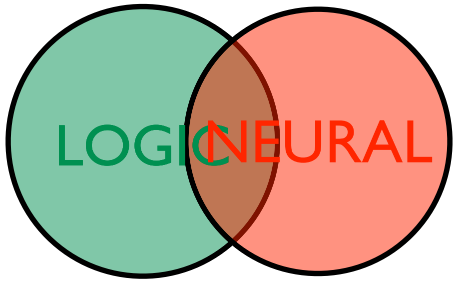
directed StarAI approach and logic programs



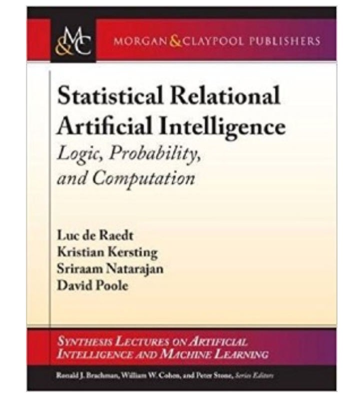
the logic is encoded in the network
how to reason logically ?

LOGIC NEURAL

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



2. Directed vs Undirected the NeSy dimension



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

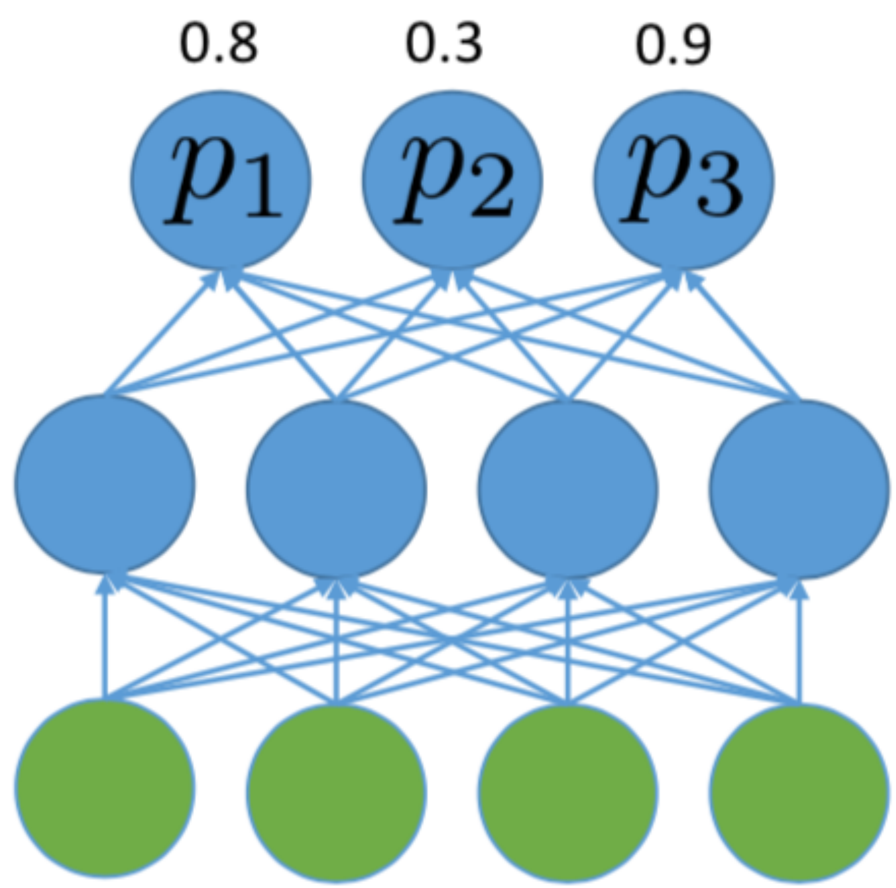
Also, many NeSy systems are doing
knowledge based model construction KBMC
where logic is used as a template

Just like in StarAI

Logic as constraints

undirected StarAI approach and (soft) constraints

multi-class classification



This constraint should be satisfied

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

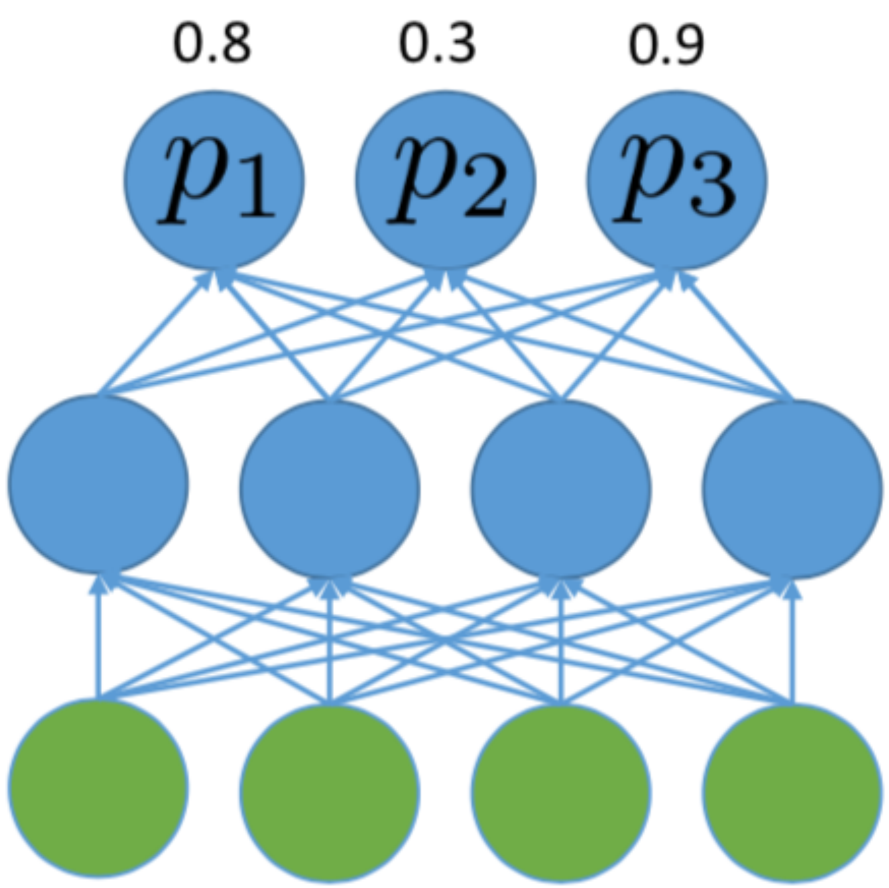
from Xu et al., ICML 2018

LOGIC NEURAL

Logic as constraints

undirected StarAI approach and (soft) constraints

multi-class classification

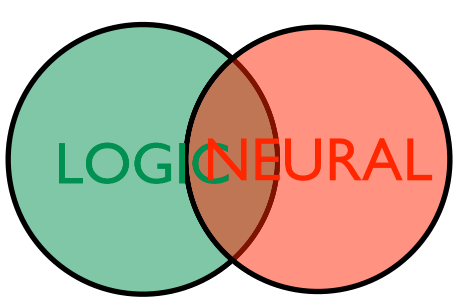


Probability that constraint is satisfied

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

basis for SEMANTIC LOSS

(weighted model counting)

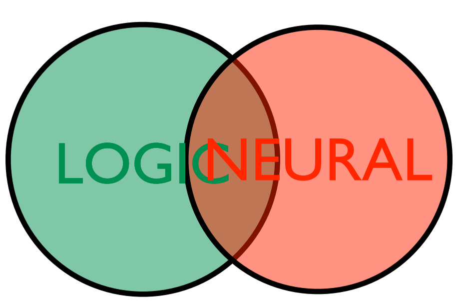


Logic as a regularizer

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



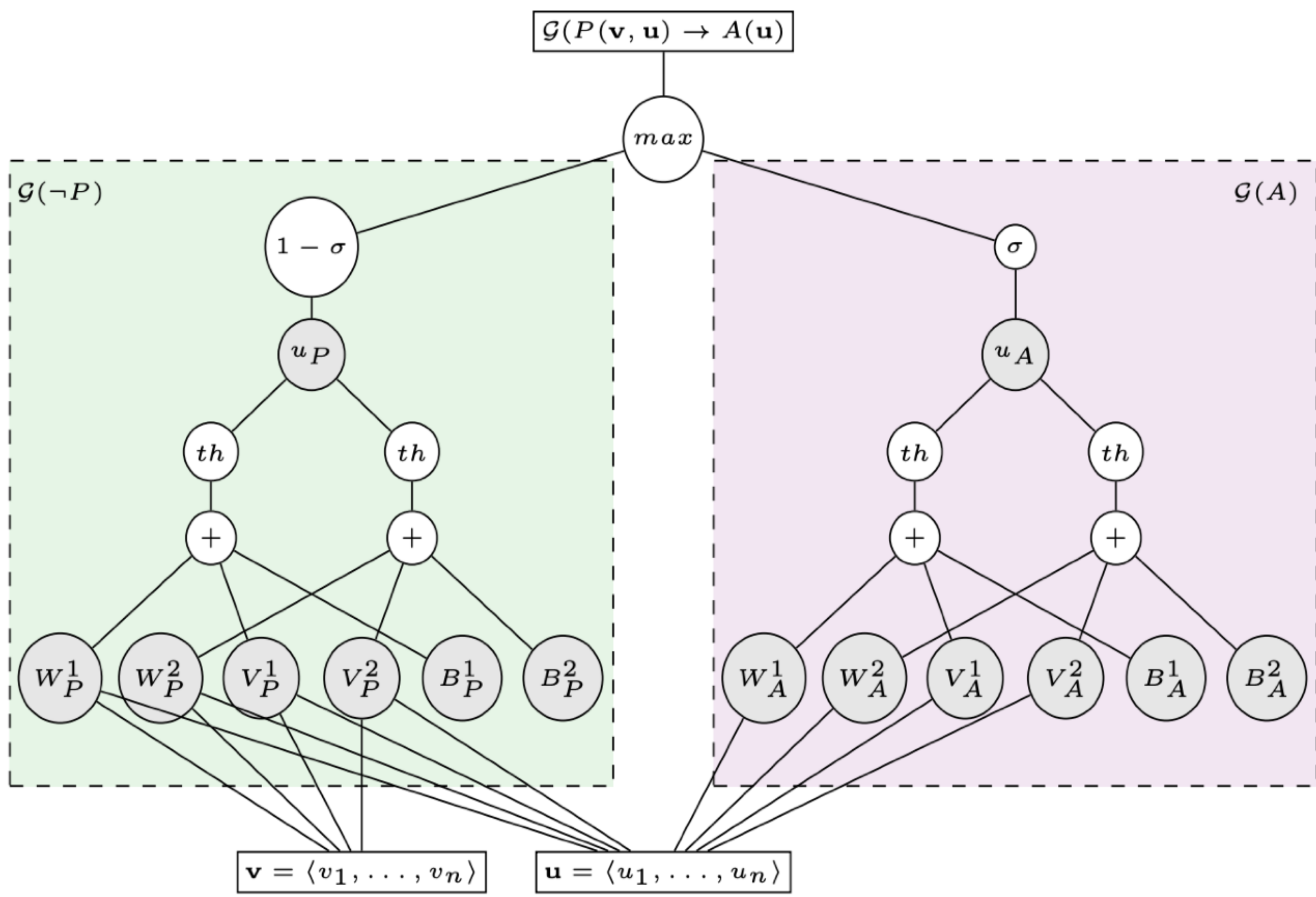
Logic as a regularizer

- Semantic Loss can be used with any logical constraint theory
- Examples with semi-supervised learning, where the constraint enforces that each example should have a class
- very nice properties :
  - differentiable, also monotonicity
  - if $\alpha \models \beta$ then $SLoss(\alpha) \geq SLoss(\beta)$

Logic Tensor Networks

undirected StarAI approach and (soft) constraints

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



LOGIC NEURAL

Serafini & Garcez

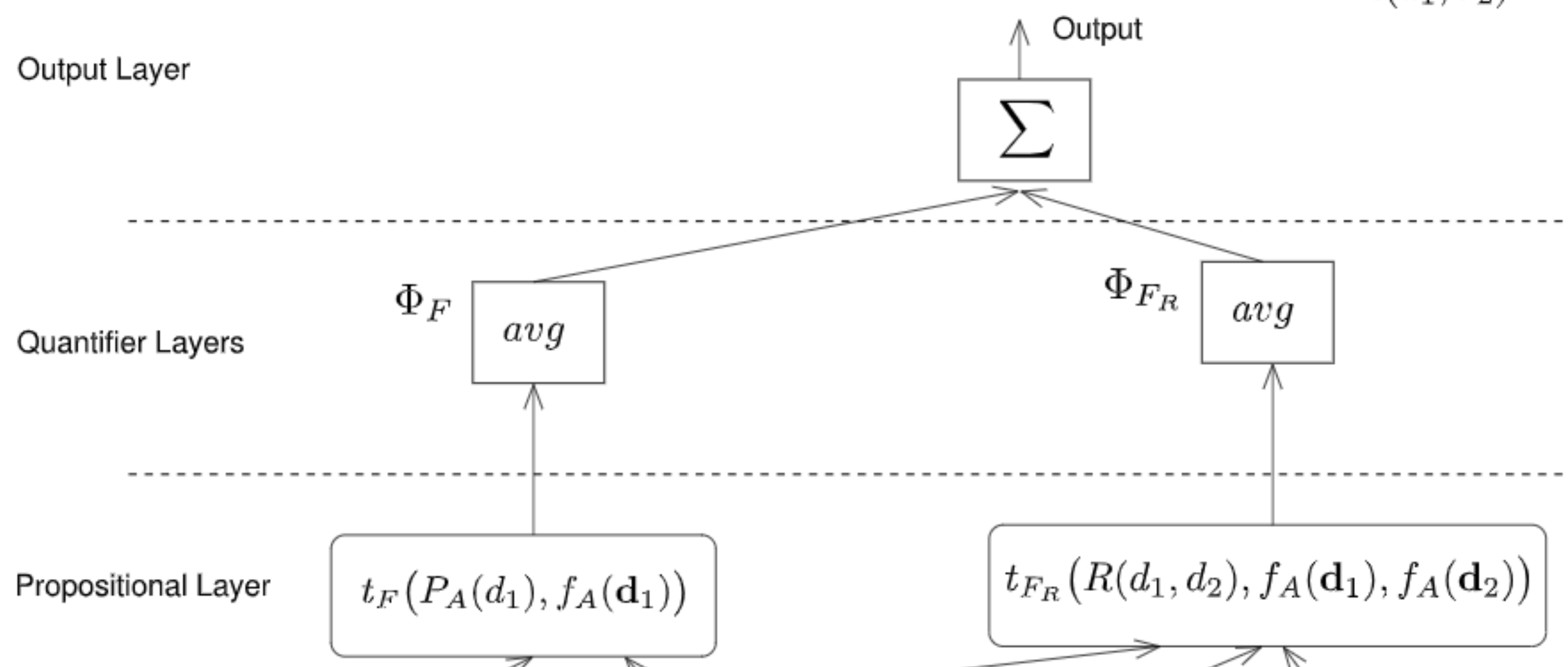
Semantic Based Regularization

undirected StarAI approach and (soft) constraints

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

Evidence Predicate Groundings

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

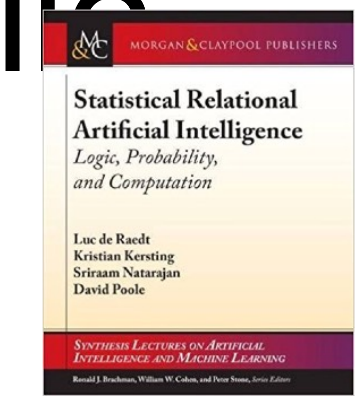


the logic is encoded in the network
how to reason logically ?

LOGIC NEURAL

Diligenti et al. AIJ

Two types of Neural Symbolic Systems



Just like in StarAI

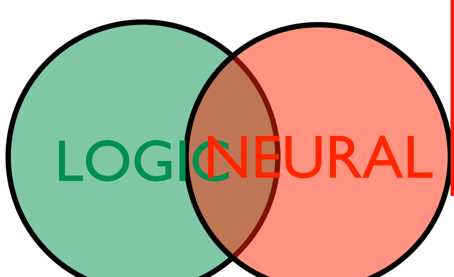
Logic as a kind of ***neural program***

directed StarAI approach and logic programs

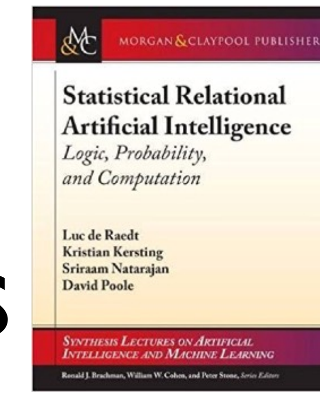
Logic as the ***regularizer***
(reminiscent of Markov Logic Networks)

undirected StarAI approach and (soft) constraints

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

Just like in StarAI

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

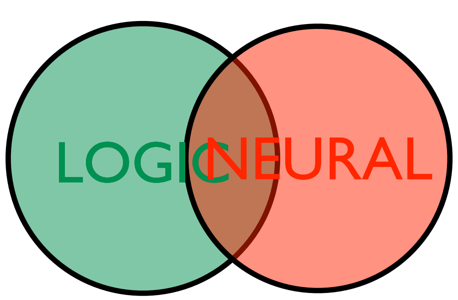
directed StarAI approach and logic programs

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

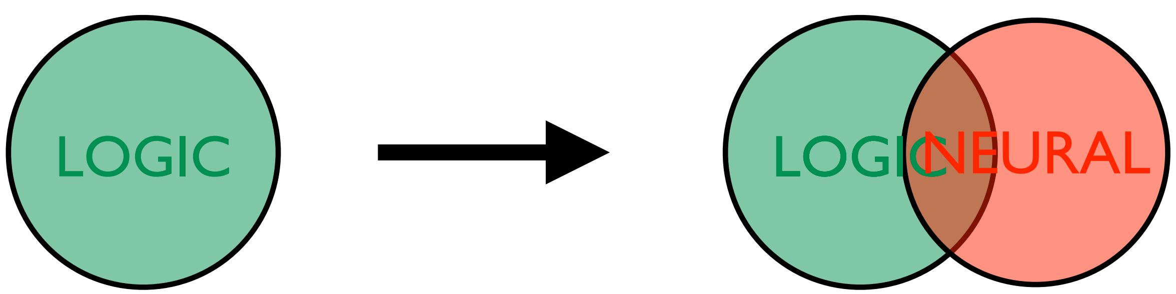
undirected StarAI approach and (soft) constraints

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

Just like in StarAI



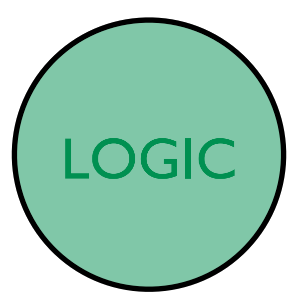
3. Types of Logic



3. Types of Logic
Key Messages

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

3. Types of Logic



Various flavours of logic

alarm :– earthquake.

alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.

calls_john :– alarm, hears_alarm_john.

Propositional logic

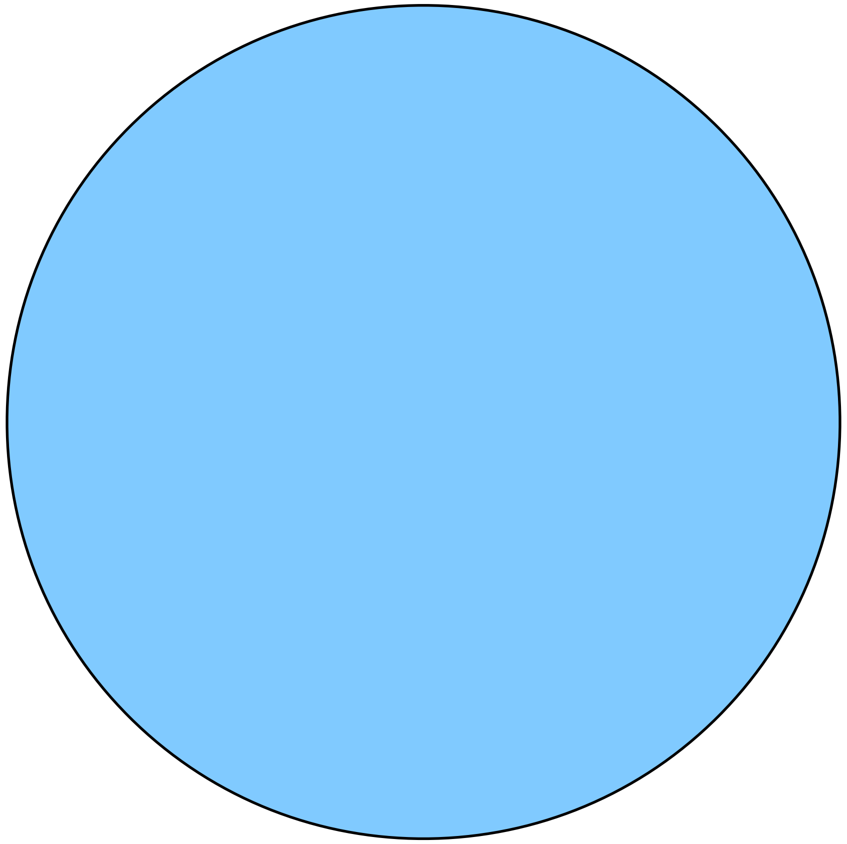
```
stress(ann).
influences(ann,bob).
influences(bob,carl).

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).
```

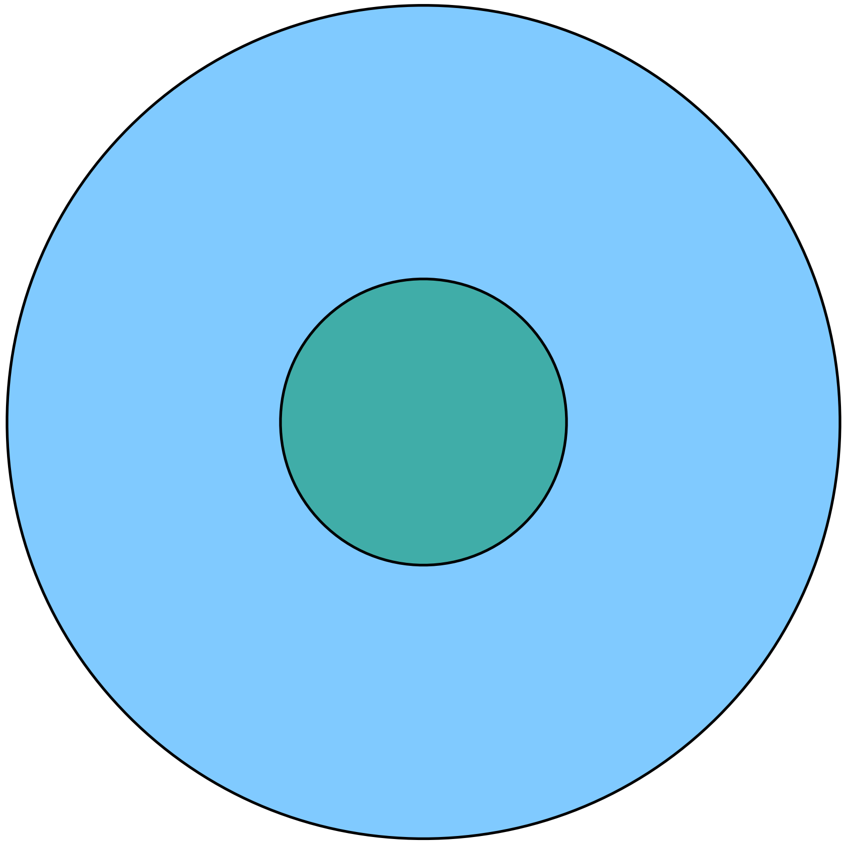
recursion

LOGIC

erc

Various flavours of first-order logic

Logic programs
= programming language



Datalog
= Logic programs
that always terminate

LOGIC

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).
```

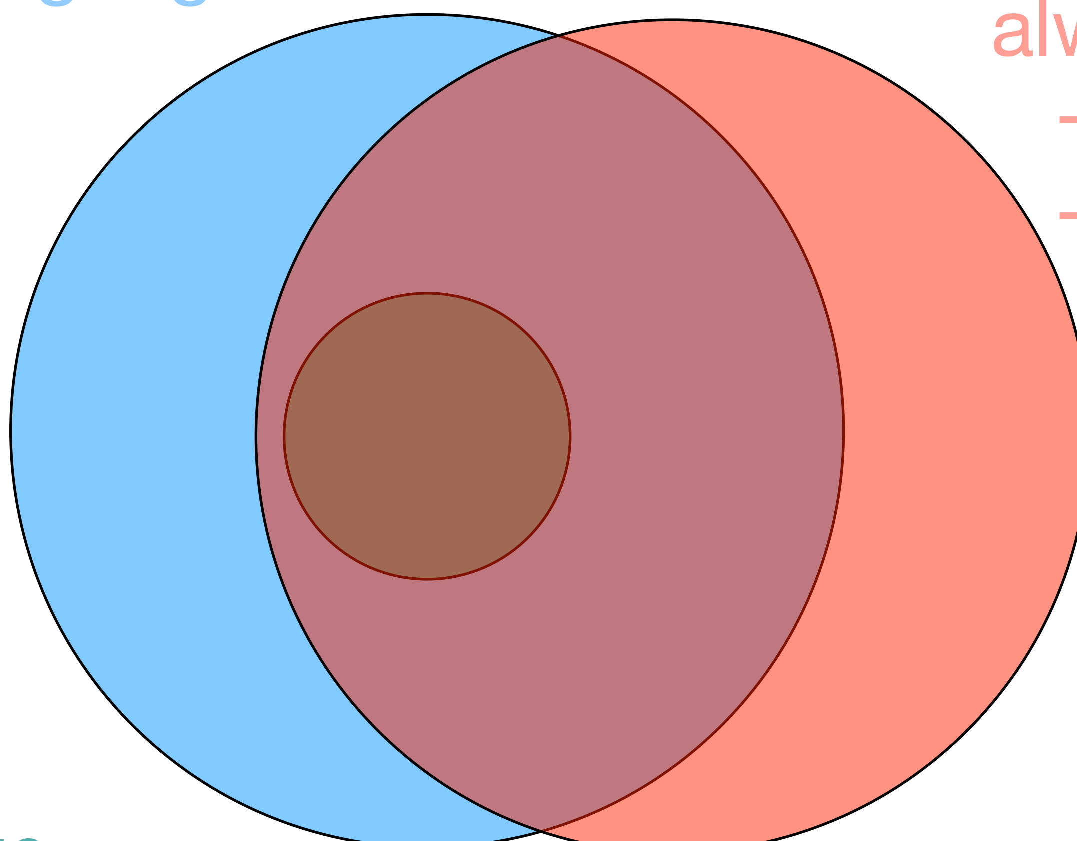
LOGIC

erc

Various flavours of first-order logic

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

Logic programs
= programming language



Datalog
= Logic programs that always terminate

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

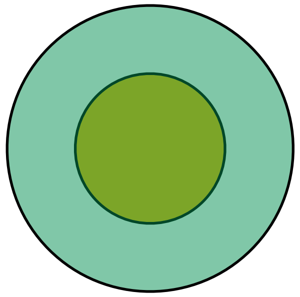
LOGIC

What can it do?

Propositional logic:
simple propositional reasoning

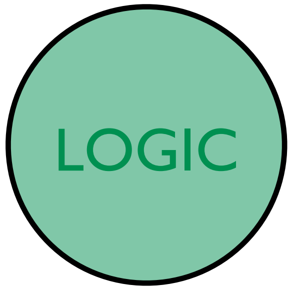
LOGIC

What can it do?

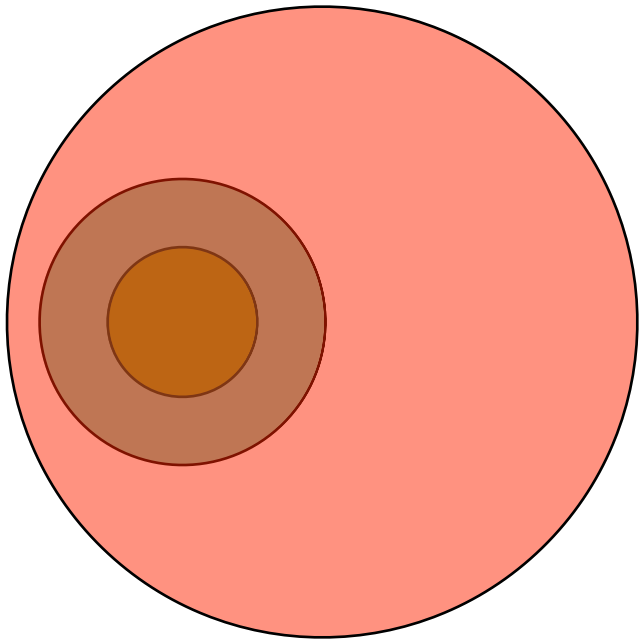


Datalog:
database queries

Propositional logic:
simple propositional reasoning



What can it do?



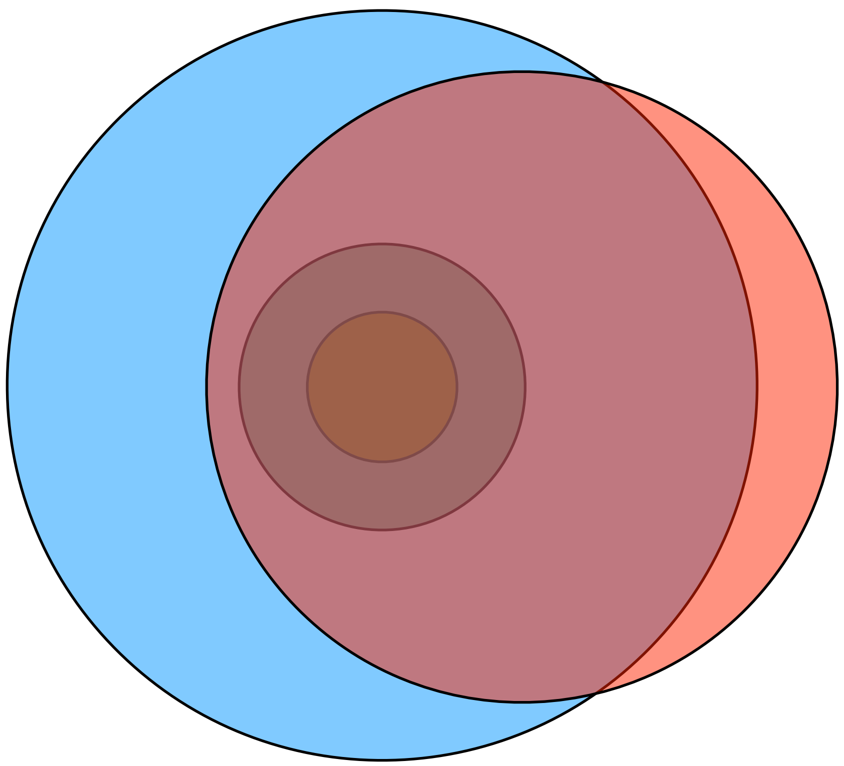
Answer-set programming:
database queries, common-sense reasoning, preferences

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

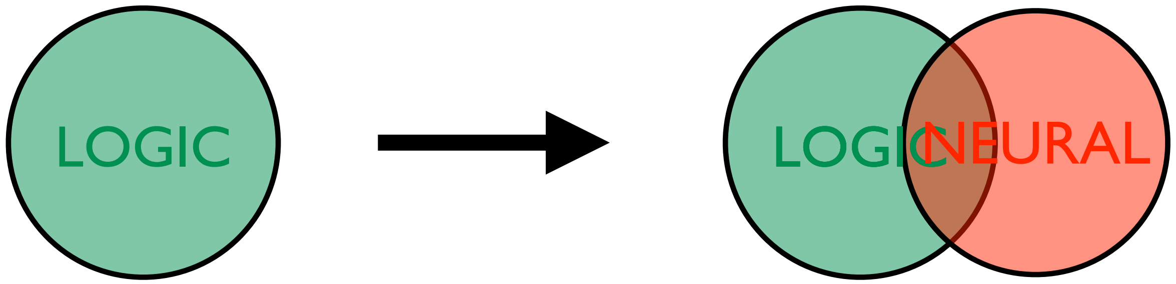
{edge(1,2), path(1,2)}

First-order logic can have many models

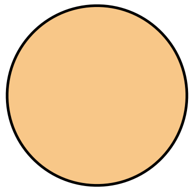
{edge(1,2), path(1,2)}
{edge(1,2), path(1,2), path(1,1)}
{edge(1,2), path(1,2), path(2,1)}

LOGIC

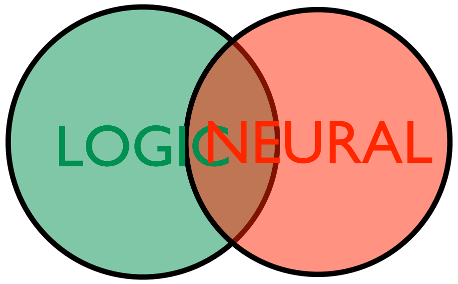
3. Types of Logic



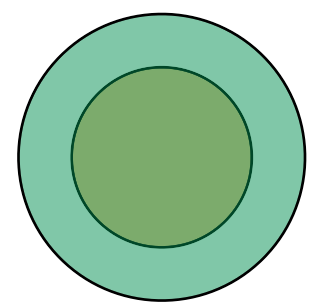
Logic in NeSy - Propositional logic



Semantic loss

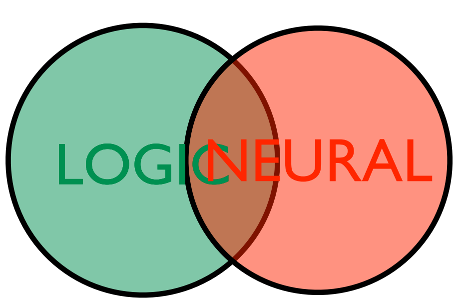


Logic in NeSy - Datalog

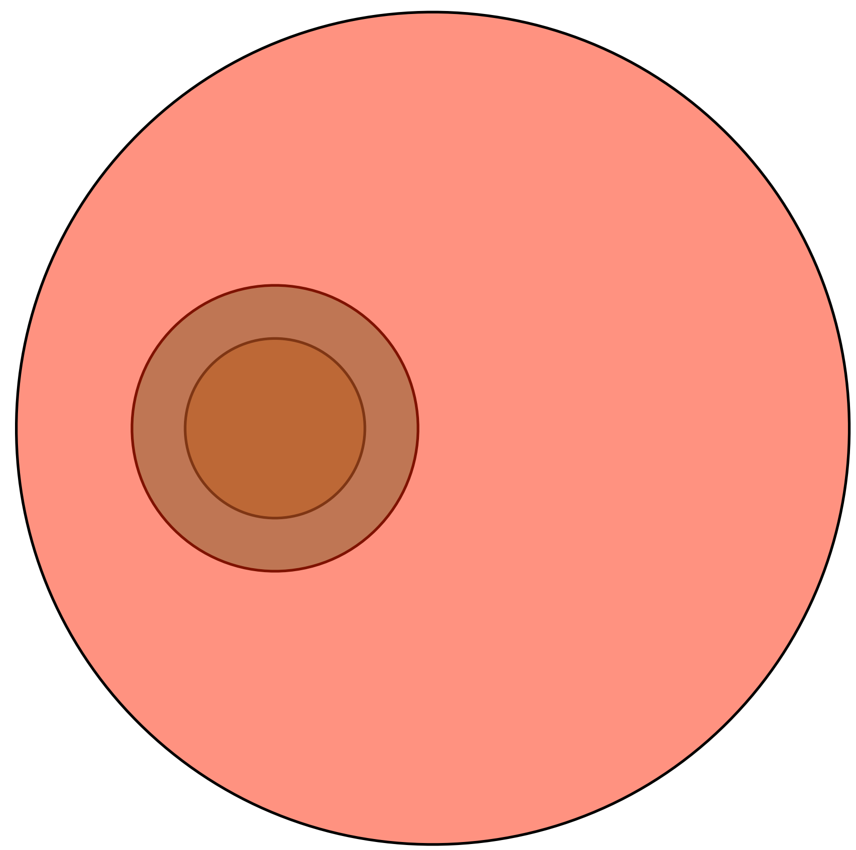


∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



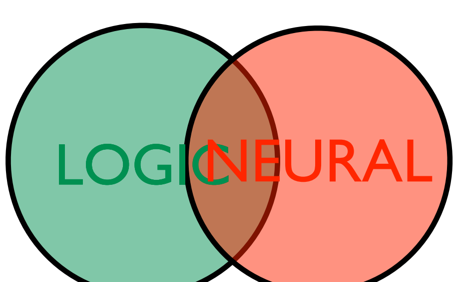
Logic in NeSy - Answer-set programming



NeurASP

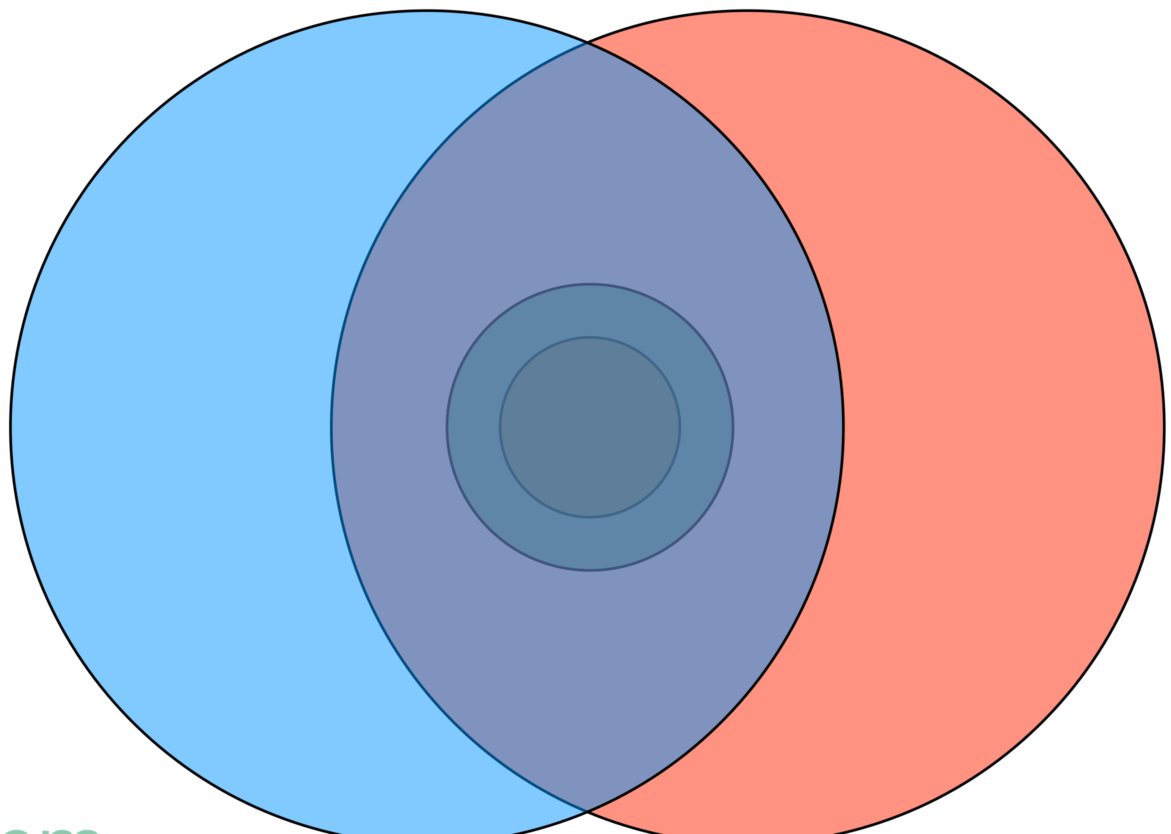
∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



Logic in NeSy - Logic programming

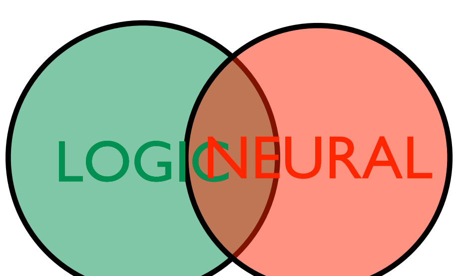
DeepProblog,
NLProlog



NeurASP

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

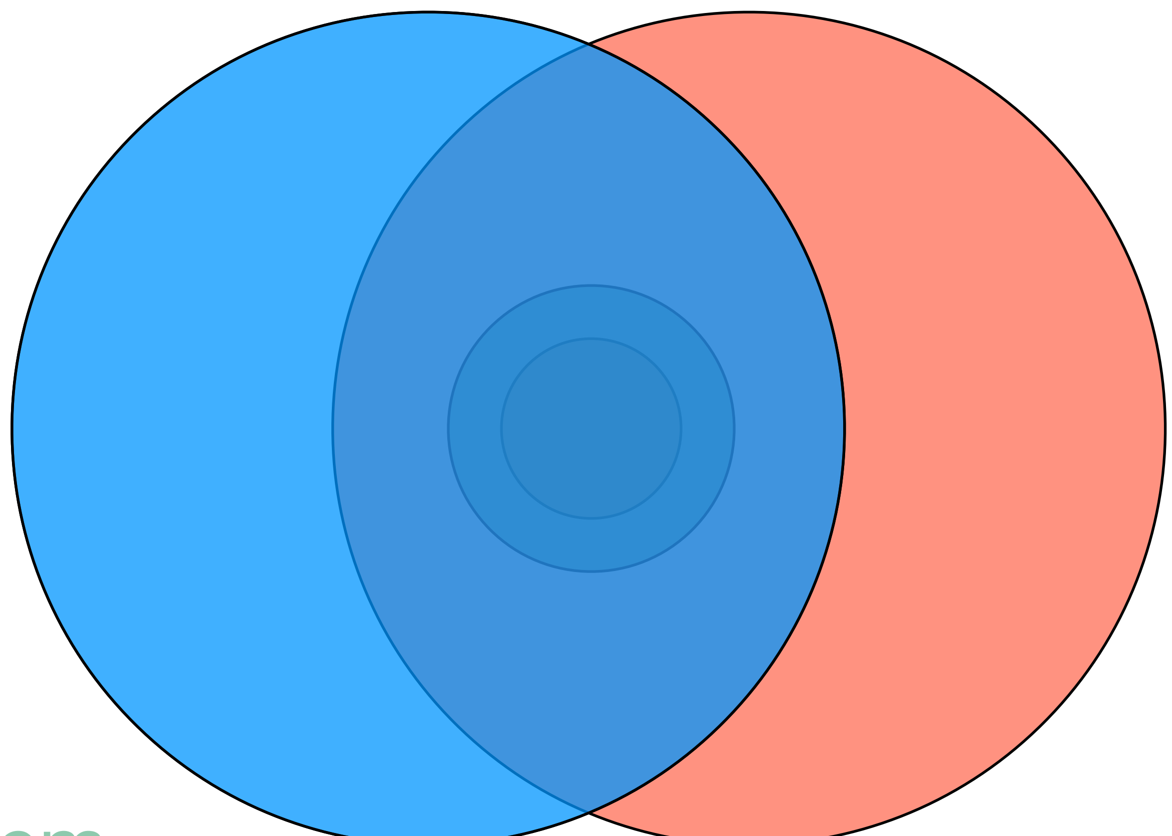
Semantic loss



Logic in NeSy - First-order logic

Logic tensor networks, NMLN,
SBT, RNM

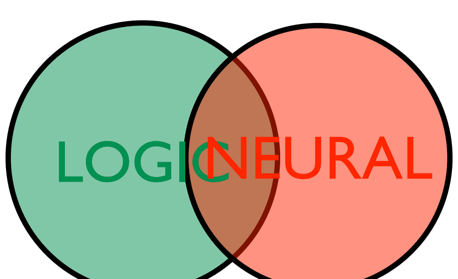
DeepProblog,
NLProlog



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

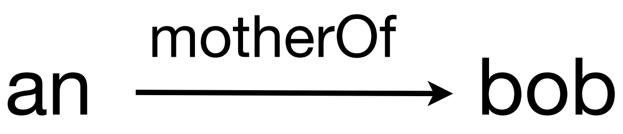
4. Symbolic vs sub-symbolic

4. Symbolic vs sub-symbolic
Key Messages

- Entities are represented very differently in symbolic and sub-symbolic systems, but they are complementary
- NeSy systems can be categorized by how they use symbolic and sub-symbolic intermediate representations

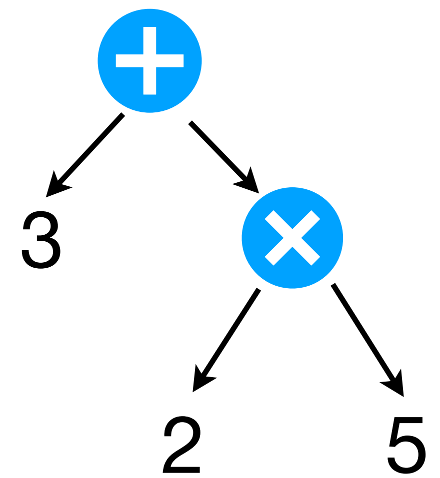
Symbolic representations

- Atoms: an, bob
- Numbers: 4, -3.5
- Variables: X,Y
- Structured terms: $f(t_1,\ldots,t_n)$
  - motherOf(an,bob)
  - [-0.1,1.2,0.5]
  - [[1,2,3],[4,5,6]]
  - plus(3,times(2,5))
  - ...



-0.1	1.2	0.5

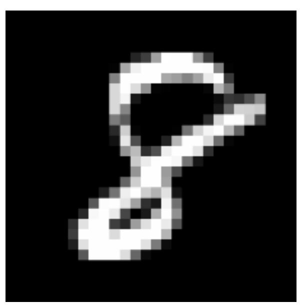
1	2	3
4	5	6



LOGIC

Sub-symbolic representations

- Sub-symbolic systems require numerical representation
- Often, entities are already numerical in nature



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

- Generally, these representations are fixed in size and dimensionality
- Exceptions require special neural architectures, e.g.
    - Recurrent neural networks
    - Fully convolutional networks
    - ...

NEURAL

Comparing symbols: unification

- Powerful mechanism for symbol matching
  - basis for many logic-based 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

Sub-symbols in StarAI

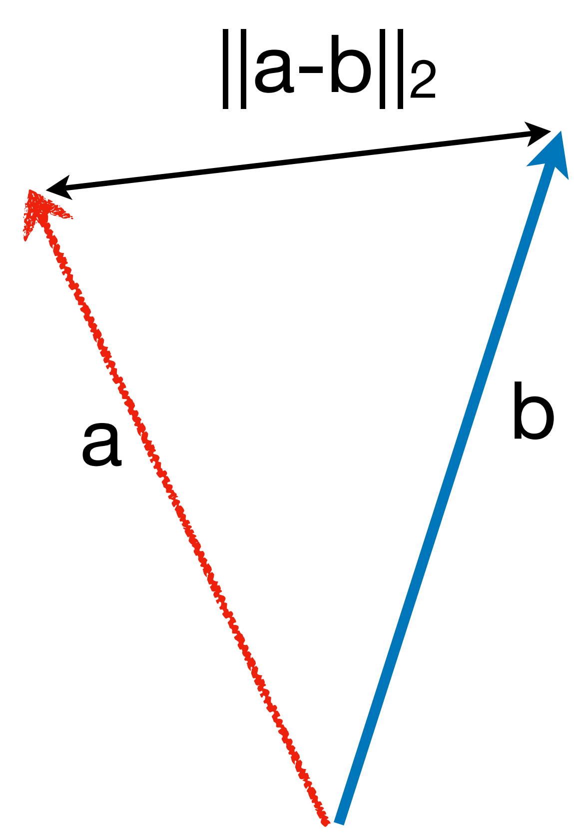
- It is possible to represent these sub-symbols in logic
  - vectors: [0.1, -0.5, 0.6]
  - matrices: [[0.2,0.4],
    [0.3, 0.1]]
  - ...

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

LOGIC

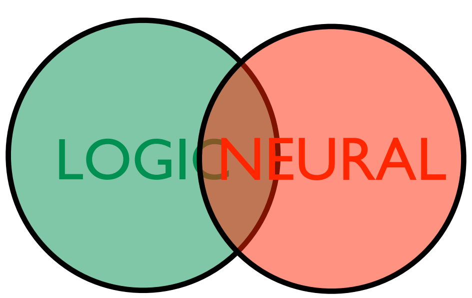
Comparing sub-symbols

- 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?



NEURAL

4. Symbolic vs sub-symbolic
Translating between representations



Symbols to sub-symbols

- A lot of deep learning research is on how to represent symbols

The quick brown fox ... → | 132 | 32 | 204 | ... | → | -0.8 | 0.4 | 0.6 | ... |

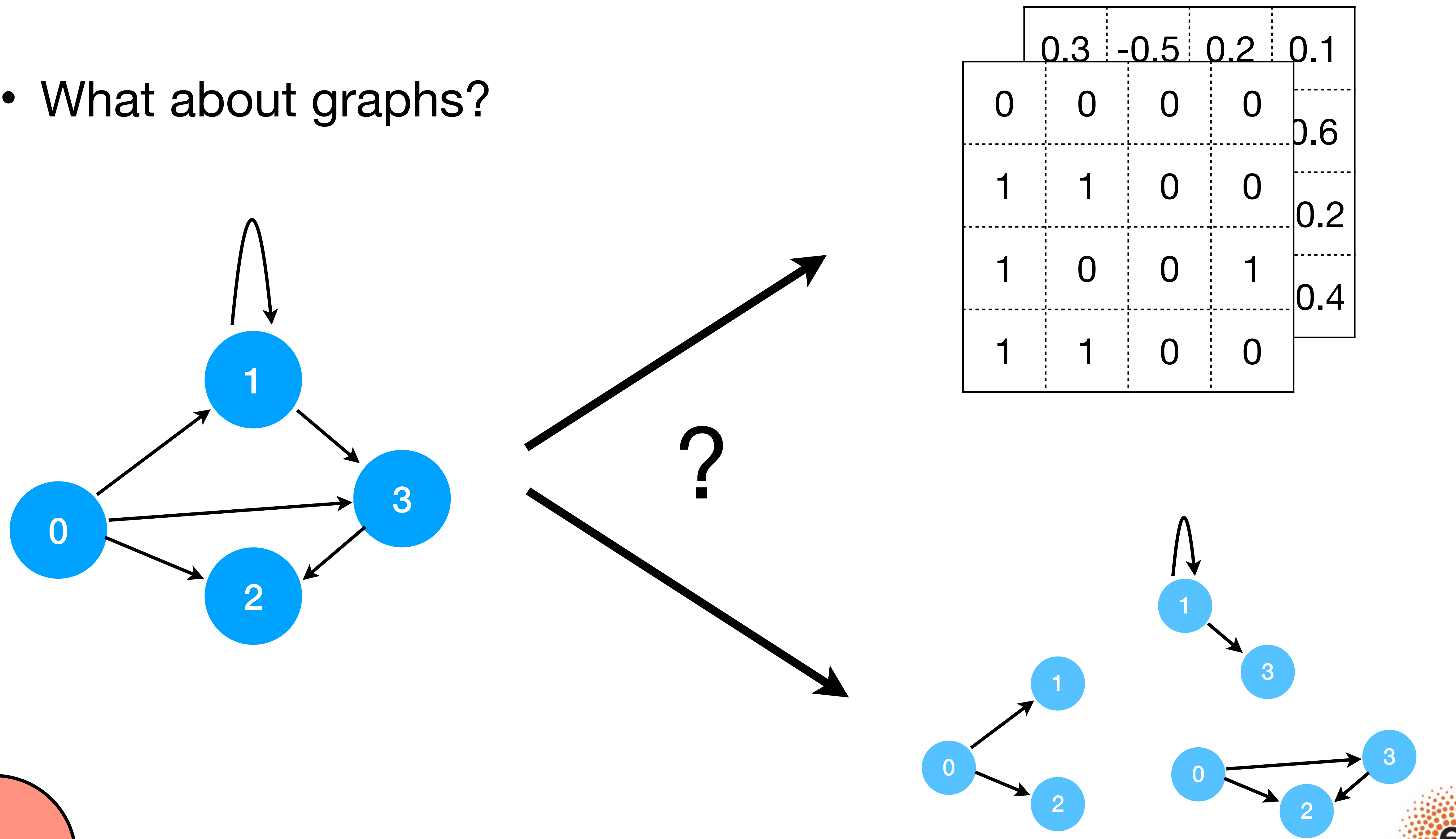
- Encoding relations r(h,t)
  - Many ways to structure embedding space

Models	score function $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$
TransE [2]	$-\|\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|\|_{1/2}$
TransR [10]	$-\|\|M_r\mathbf{h} + \mathbf{r} - \mathbf{M_r t}\|\|_2^2$
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]	$\mathbf{h}^\top \mathbf{M_r t}$
RotatE [15]	$-\|\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|\|^2$

NEURAL

Symbols to sub-symbols

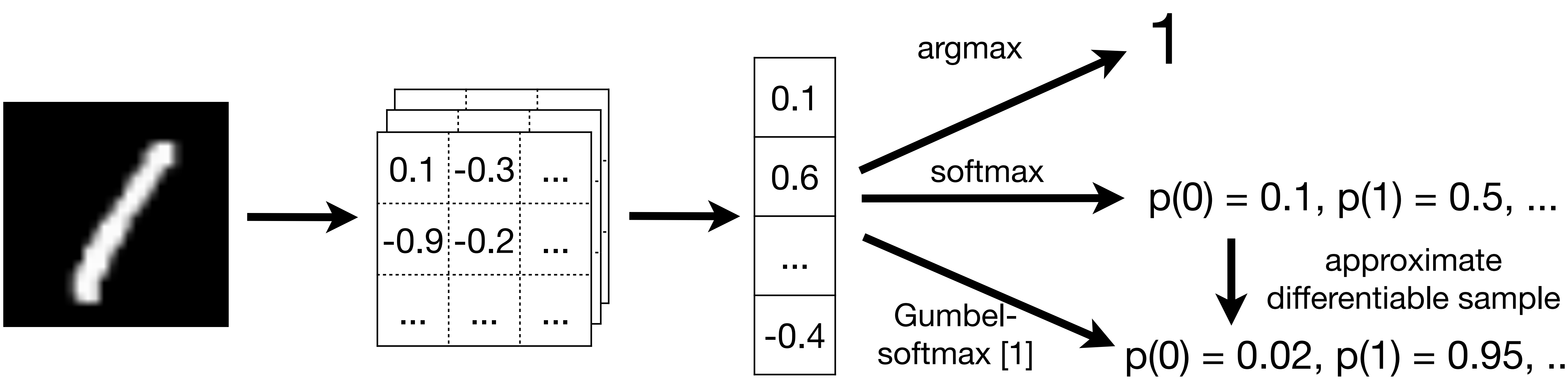
- What about graphs?



NEURAL

Sub-symbols to symbols

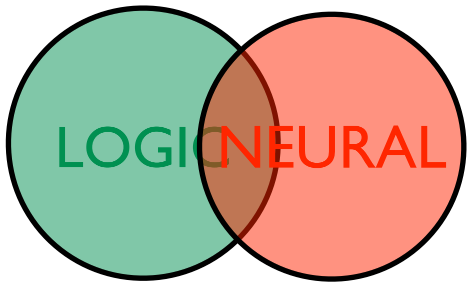
- E.g. in neural network classifiers
  - Turn real-valued vector into discrete classes
  - Final layer with specific activation function



NEURAL

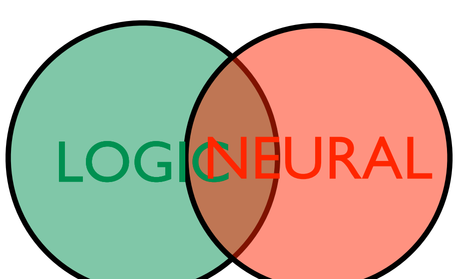
[1] Jang et al.:"Categorical Reparameterization with Gumbel-Softmax", ICLR 2017

4. Symbolic vs sub-symbolic Representations in NeSy

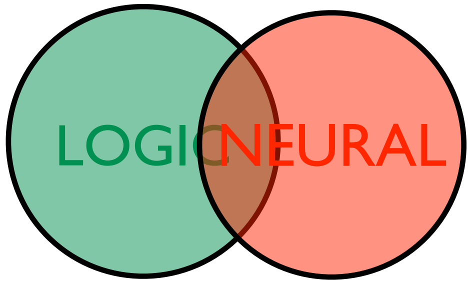


Representation in NeSy

- StarAI
  - Input = intermediate = output = symbolic representation
- Neural methods
  - Input = intermediate = sub-symbolic
  - Output =
    - Symbolic (classifier)
    - Or sub-symbolic (auto-encoder, GAN, regression, ...)
- NeSy
  - Intermediate representation = symbolic or sub-symbolic
  - We discern several approaches



4. Symbolic vs sub-symbolic
Single translation step



Single translation step

- Symbolic input is mapped onto sub-symbols
  - One-hot encoding, relational embeddings, ...
- Afterwards, all reasoning happens in sub-symbolic space
- This approach is seen in most NeSy systems
- Examples include:
  - LTNs[1], SBR[2], NLMs[3], TensorLog[4]

[1] Serafini, et al.: "Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge", NeSy@HLAI 2016
[2] Diligenti et al.: "Semantic based regularization for learning and inference", Artificial Intellligence 2017
[3] Dong et al.: "Neural Logic Machines", ICLR 2019
[4] Cohen et al.: "Deep Learning meets Probabilistic DBs"

Logic Tensor Network

- This translations is 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* $P \in \mathcal{P}$;

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

LOGIC NEURAL

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

Logical Theory

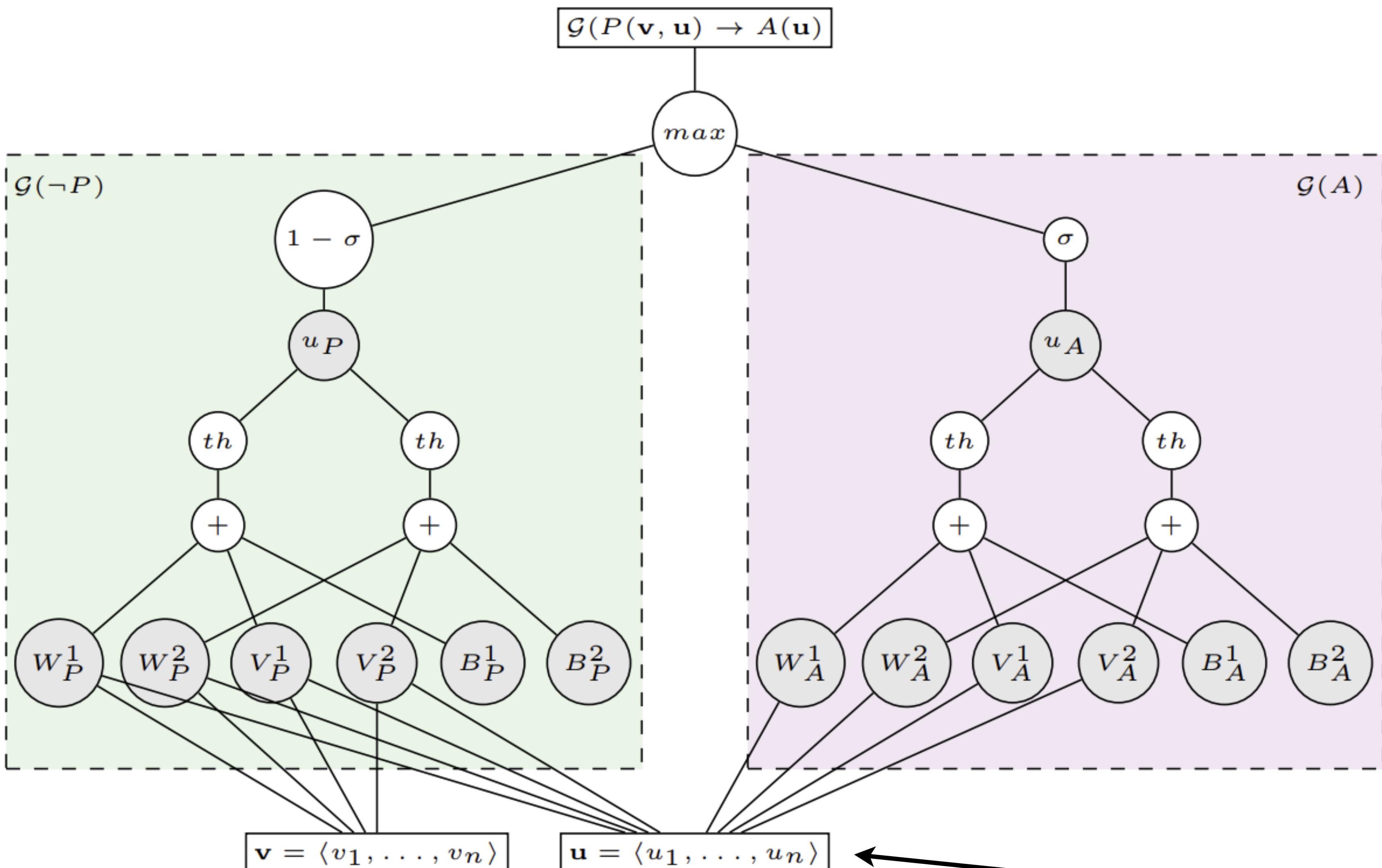
```
stress(ann).
influences(ann,bob).
influences(bob,carl).

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

Logic Tensor Network

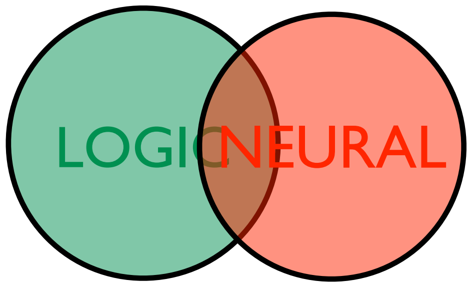


Sub-symbolic computation

Encoding symbols

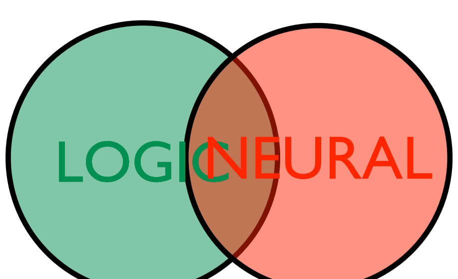
Luciano Serafini, Artur S. d'Avila Garcez: Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge. NeSy@HLAI 2016

4. Symbolic vs sub-symbolic
Alternating symbols and sub-symbols



Alternating symbols and sub-symbols

- Both symbolic and sub-symbolic representations are used
  - Not simultaneously by one component
  - Some components work on symbols, others on sub-symbols
- Indicative of systems that implement an interface
- Very natural for NeSy systems originating from a logical framework
- Examples include:
  - DeepProbLog[1], NeurASP[2], ...
  - ABL[3], NeuroLog[4], ..



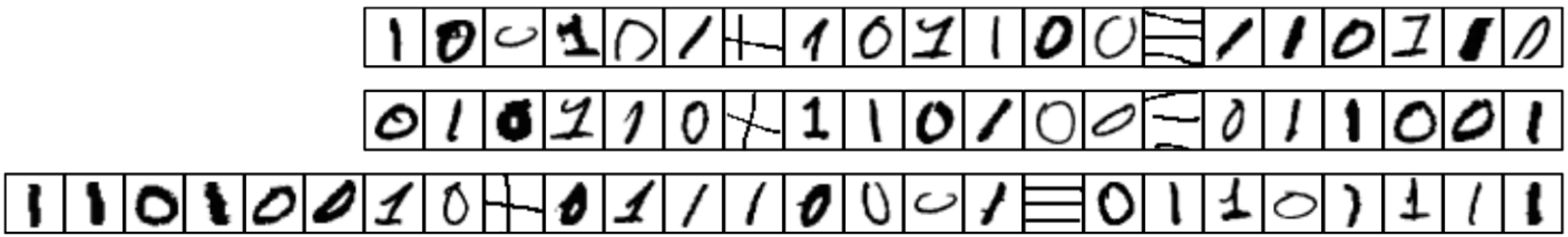
[1] Manhaeve et al: "DeepProbLog: Neural Probablistic Logic Programming", NeurIPS 2018
[2] Yang et al: "NeurASP: Embracing Neural Networks into Answer Set Programming", IJCAI 2020
[3] Dai et al.: "Bridging Machine Learning and Logical Reasoning by Abductive Learning", NeurIPS 2019
[4] Tsamora et al. "Neural-symbolic integration: A compositional perspective"

ABL

- ABL tries to learn:
  - A perception model that interprets sub-symbolic input
  - A set of logical rules (knowledge)
- From
  - A set of examples (sub-symbolic inputs, label)
  - A set of possible labels for the sub-symbolic inputs
  - A set of rules representing background knowledge
- Such that
  - The perception models applies labels to the sub-symbolic inputs
  - These labels are consistent with the knowledge

Dai et al.: Bridging Machine Learning and Logical Reasoning by Abductive Learning. NeurIPS 2019

ABL



Positive

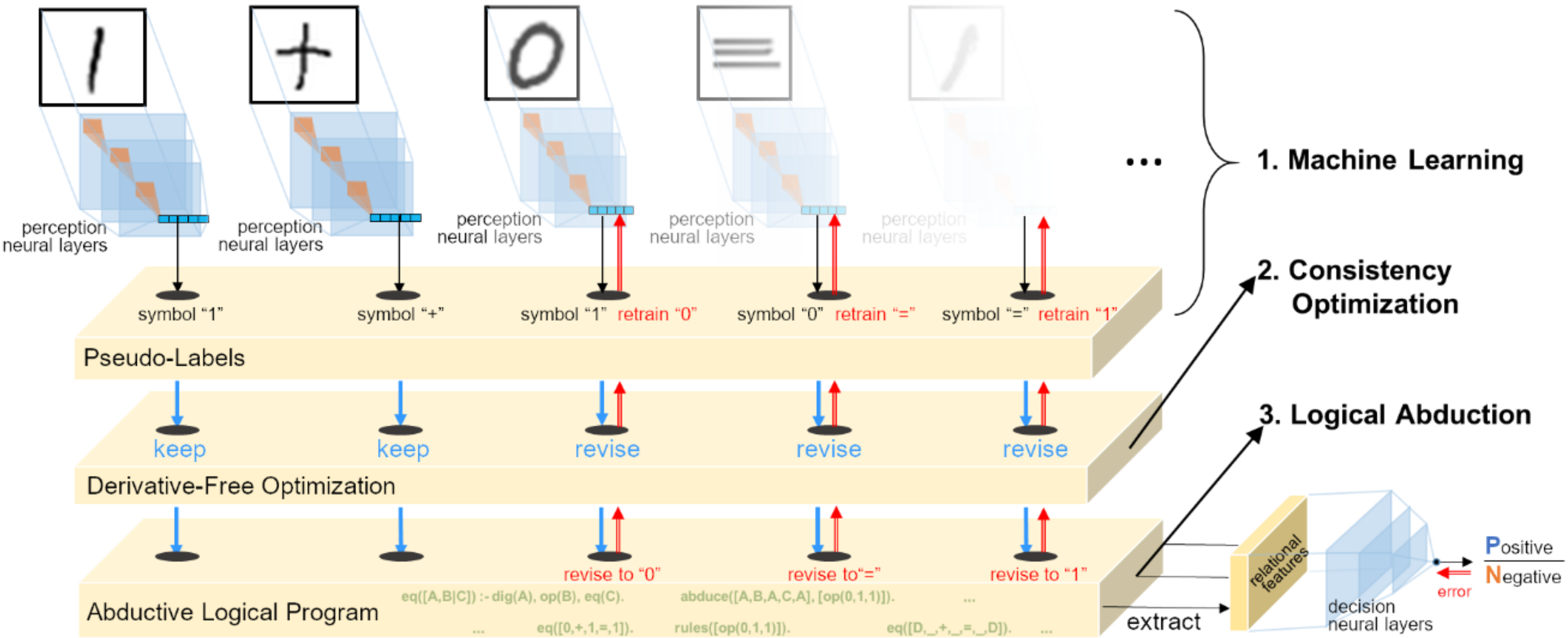
Negative

?

From Dai et al.: Bridging Machine Learning and Logical Reasoning by Abductive Learning. NeurIPS 2019

- Given knowledge:
  - Images represent: 0, 1, + or =
  - Equation: [list of 0 and 1] + [list of 0 and 1s] = [list of 0 and 1s]
- Learn:
  - Classify images into 0, 1, + or =
  - What operation is performed on the first two lists to form the last

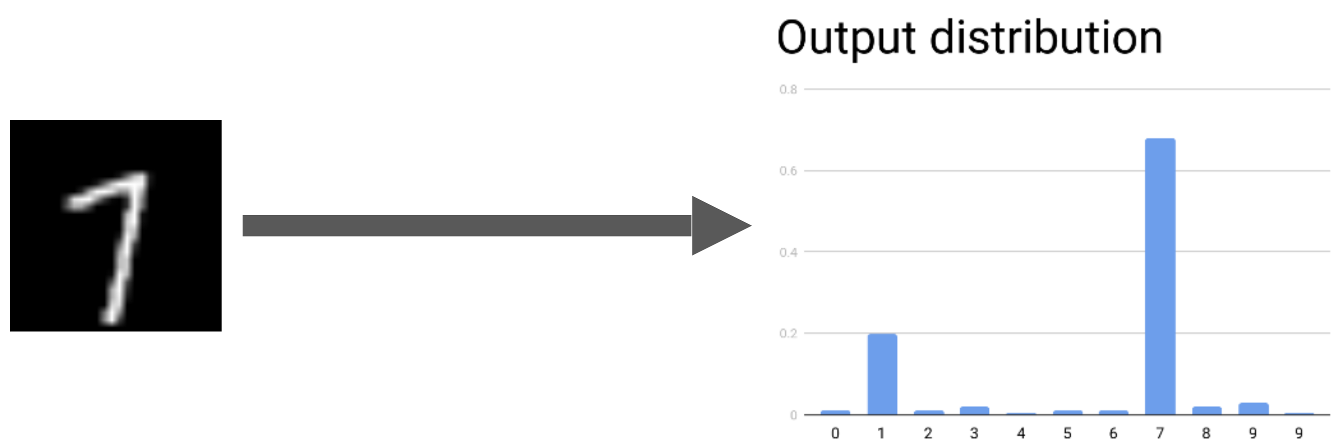
ABL



From Dai et al.: Bridging Machine Learning and Logical Reasoning by Abductive Learning. NeurIPS 2019

LOGIC NEURAL

Neural predicate



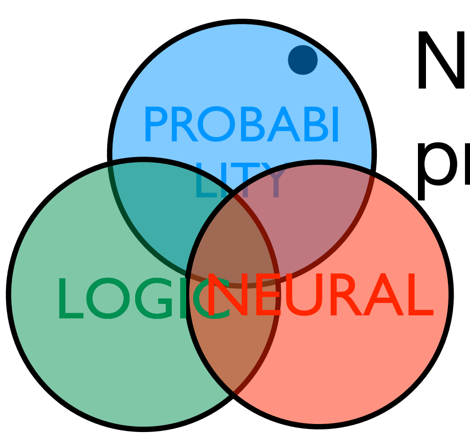
- Neural networks have uncertainty in their predictions
- A normalized output can be interpreted as a probability distribution
- Neural predicate models the output as probabilistic facts
- No changes needed in the probabilistic host language

Key Idea DeepProbLog

unify the basic concepts in logic and neural networks:

neural predicate ~ neural net

an interface between logic and neural nets



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), …

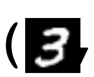
DeepProbLog exemplified: MNIST addition

Task: Classify pairs of MNIST digits with their sum

Benefit of DeepProbLog:

- Encode addition in logic
- Separate addition from digit classification

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

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

addition(,,8) :- digit(,N1), digit(,N2), 8 is N1 + N2.

Examples:

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

Example

Learn to classify the sum of pairs of MNIST digits

Individual digits are not labeled!

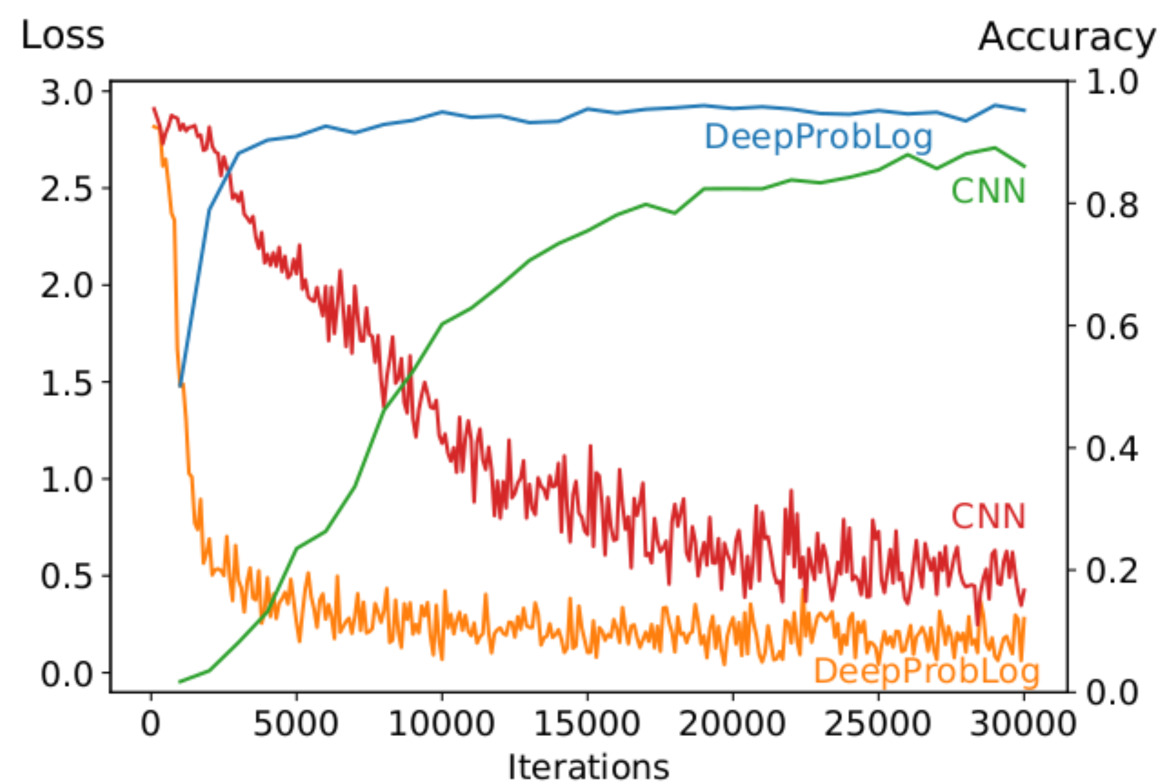
E.g. ( ,  , 8)

Could be done by a CNN: classify the concatenation of both images into 19 classes

However:  +  = ?

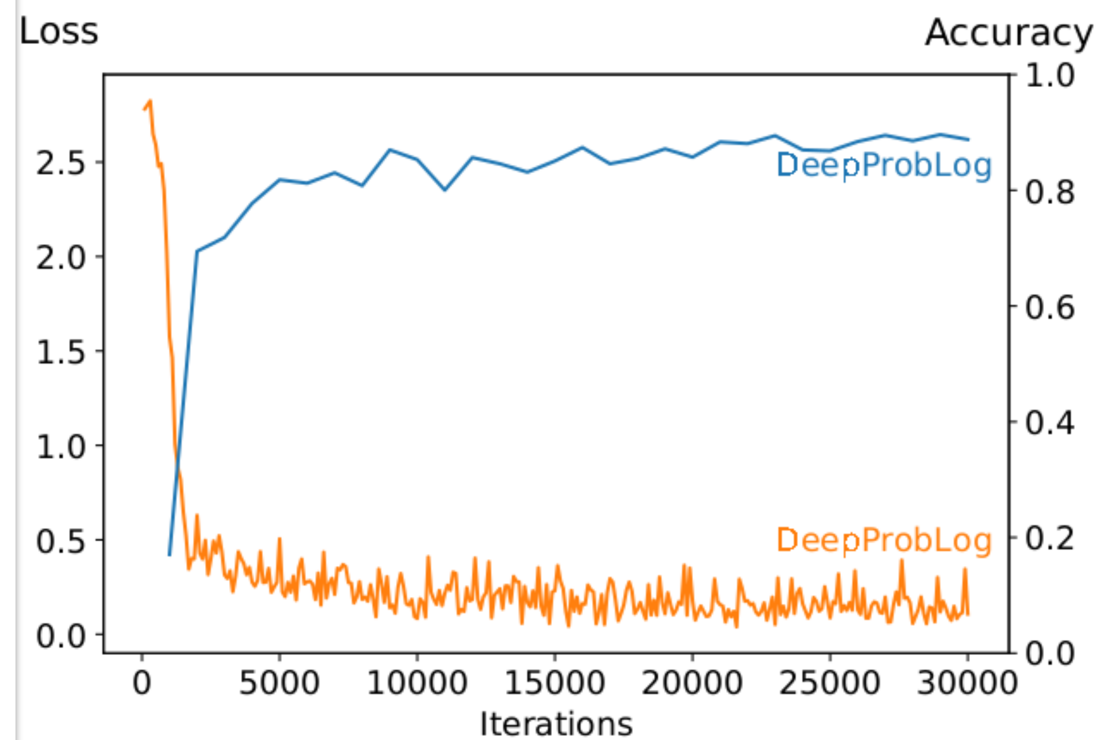
MNIST Addition

- Pairs of MNIST images, labeled with sum
- Baseline: CNN
  - Classifies concatenation of both images into classes 0 ...18
- DeepProbLog:
  - CNN that classifies images into 0 … 9
  - Two lines of DeepProblog code



Multi-digit MNIST addition with MNIST

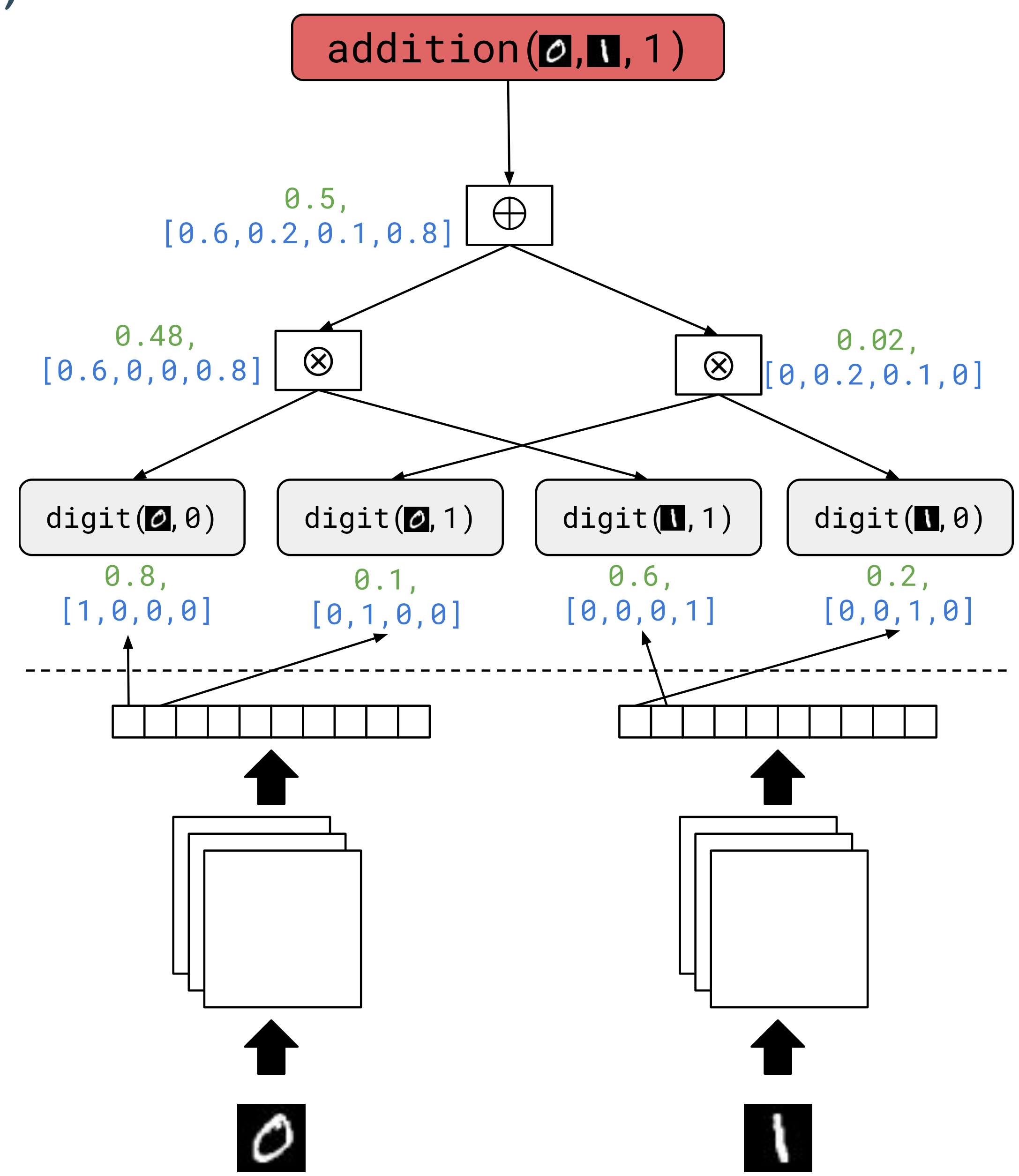
```
number ( [ ] , Result , Result ) .
number ( [H | T ] , Acc , Result) :–
  digit(H, Nr ), Acc2 is Nr +10*Acc ,
  number ( T , Acc2 , Result ) .
number (X,Y) :– number (X, 0 ,Y ) .

multiaddition(X, Y, Z ) :–
    number (X, X2 ) ,
    number (Y, Y2 ) ,
    Z is X2+Y2 .
```



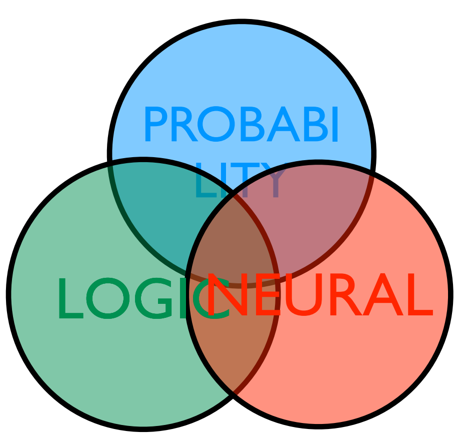
(b) Multi-digit (**T2**)

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



Useful Semirings

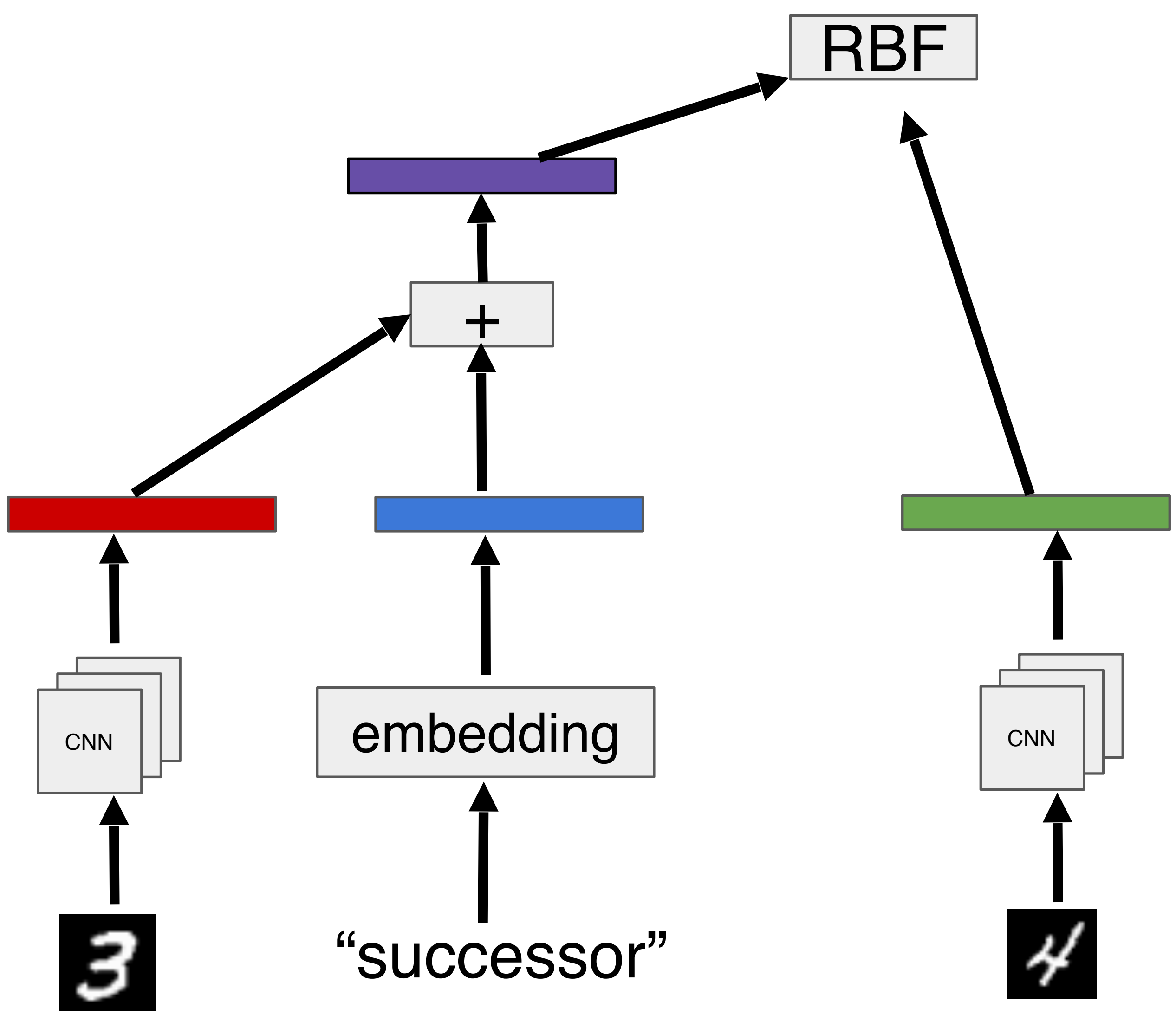
task	$\mathcal{A}$	$e^{\oplus}$	$e^{\otimes}$	$\oplus$	$\otimes$	$\alpha(v)$	$\alpha(\neg v)$	ref
SAT	$\{true, false\}$	$false$	$true$	$\vee$	$\wedge$	$true$	$true$	B, BT, G, GK, K, L, M
#SAT	$\mathbb{N}$	0	1	$+$	$\cdot$	1	1	B, G, GK, K, L
WMC	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in \mathbb{R}_{\geq 0}$	$\in \mathbb{R}_{\geq 0}$	
PROB	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in [0,1]$	$1-\alpha(v)$	B, BT, E, G, K
SENS	$\mathbb{R}[\mathcal{V}]$	0	1	$+$	$\cdot$	v or $\in [0,1]$	$1-\alpha(v)$	K
GRAD	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	$(0,0)$	$(1,0)$	Eq. (4)	Eq. (5)	Eq. (2)	Eq. (3)	E, K
MPE	$\mathbb{R}_{\geq 0}$	0	1	max	$\cdot$	$\in [0,1]$	$1-\alpha(v)$	B, BT, G, K, L, M
S-PATH	$\mathbb{N}^{\infty}$	∞	0	min	$+$	$\in \mathbb{N}$	0	BT, GK, K
W-PATH	$\mathbb{N}^{\infty}$	0	∞	max	min	$\in \mathbb{N}$	∞	BT
FUZZY	$[0,1]$	0	1	max	min	$\in [0,1]$	1	GK, M
kWEIGHT	$\{0,\ldots,k\}$	k	0	min	$+^k$	$\in \{0,\ldots,k\}$	$\in \{0,\ldots,k\}$	M
$\mathtt{OBDD}_<$	$\mathtt{OBDD}_<(\mathcal{V})$	$\mathtt{OBDD}_<(0)$	$\mathtt{OBDD}_<(1)$	$\vee$	$\wedge$	$\mathtt{OBDD}_<(v)$	$\neg\mathtt{OBDD}_<(v)$	K
WHY	$\mathcal{P}(\mathcal{V})$	$\emptyset$	$\emptyset$	$\cup$	$\cup$	$\{v\}$	n/a	GK
$\mathcal{RA}^+$	$\mathbb{N}[\mathcal{V}]$	0	1	$+$	$\cdot$	v	n/a	GK

Table 1: Examples of commutative semirings and labeling functions. The **WHY** and $\mathcal{RA}^+$ provenance semirings apply to positive literals only. Reference key: B (Bacchus et al., 2009), BT (Baras and Theodorakopoulos, 2010), E (Eisner, 2002), G (Goodman, 1999), GK (Green et al., 2007), K (Kimmig et al., 2011), L (Larrosa et al., 2010), M (Meseguer et al., 2006); more examples can be found in these references.

From Kimmig, Vanden Broeck and De Raedt, 2016

DeepProbLog: Embeddings as symbols

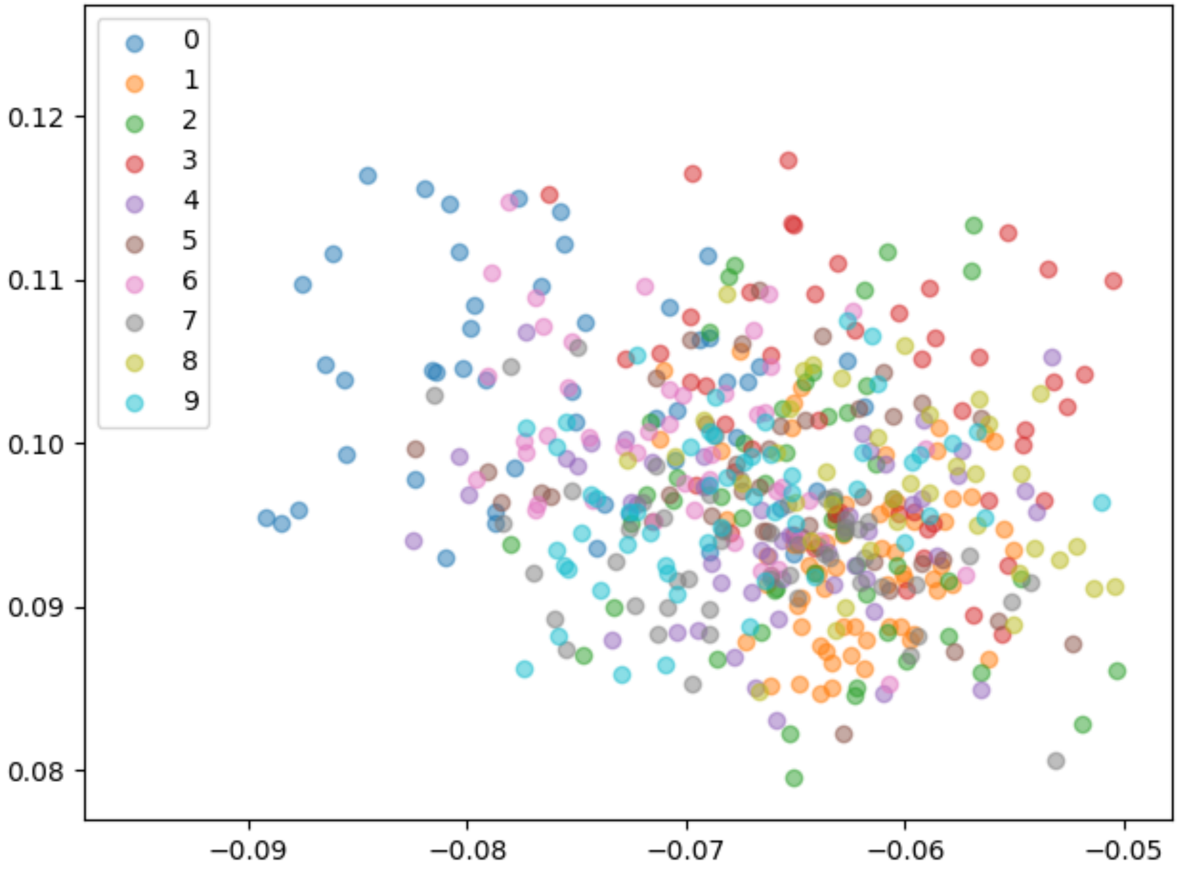
Computational Graph

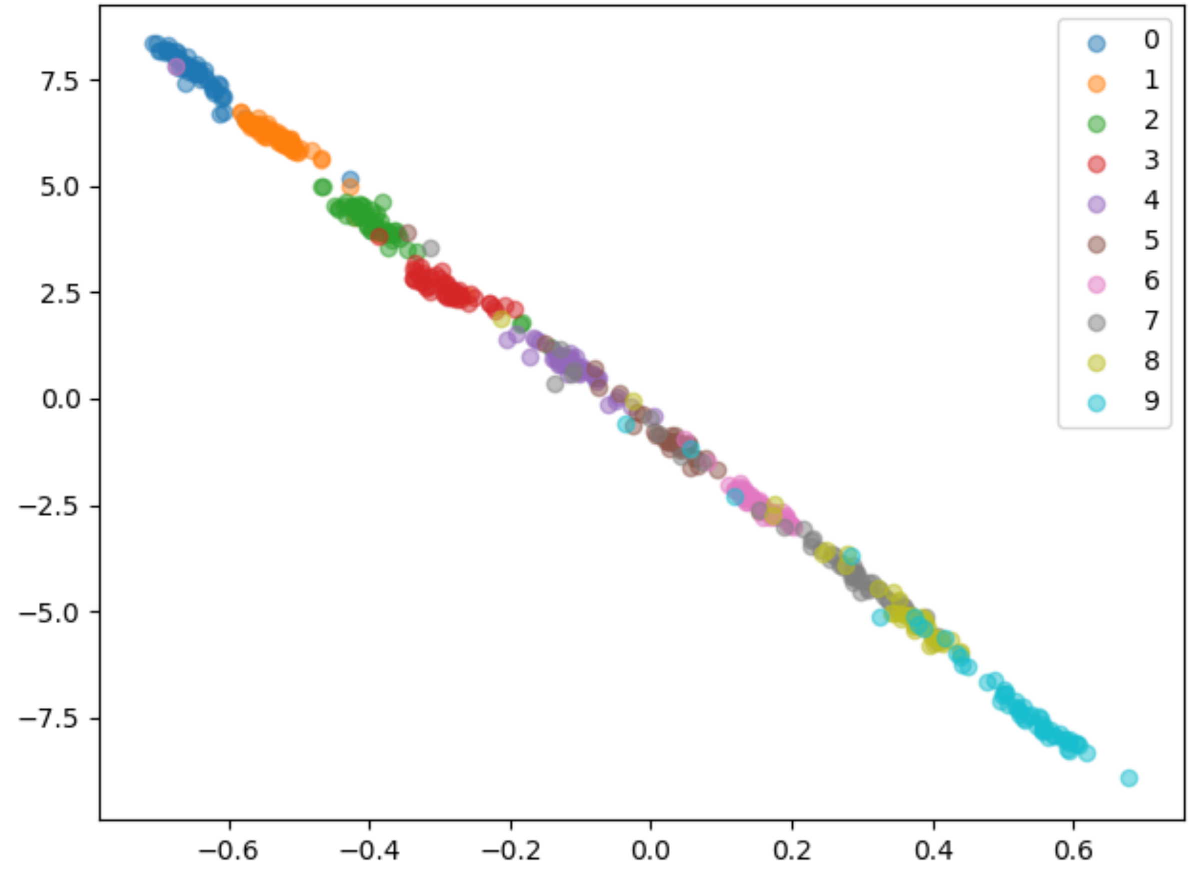



```
succesor( , ) :-
        cnn_embed( ,e1),
        cnn_embed( ,e2),
        embed("successor",r),
        add(r,e1,e3),
        rbf(e2,e3).
```

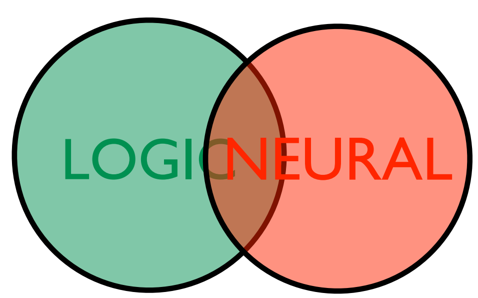
Idea of TransE [Bordes et al]

2D MNIST image embeddings



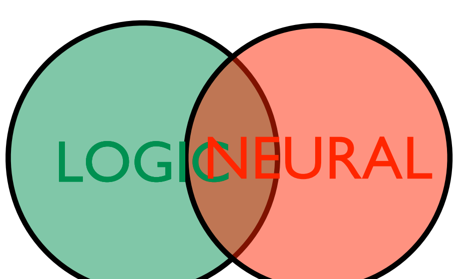


4. Symbolic vs sub-symbolic
Simultaneously symbolic and sub-symbolic



Simultaneously symbolic and sub-symbolic

- Both symbolic and sub-symbolic representations are used
  - All entities have both representations
  - Reasoning uses both **simultaneously**
- Reasoning mechanism is extended
- Only used in a few systems
  - E.g. NTP[1], CTP[2]

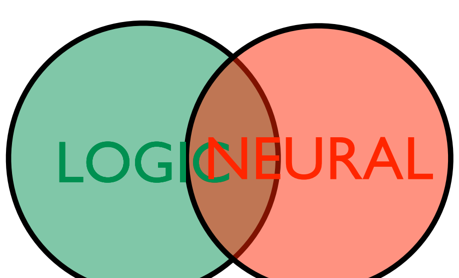


[1] Rocktäschel et al.: "End-to-end differentiable proving.", NeurIPS 2017.
[2] Minervini et al.: "Learning Reasoning Strategies in End-to-End Differentiable Proving", ICML 2020

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:
  - $sim(x,y) = \exp(- ||T_x - T_y||_2)$



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

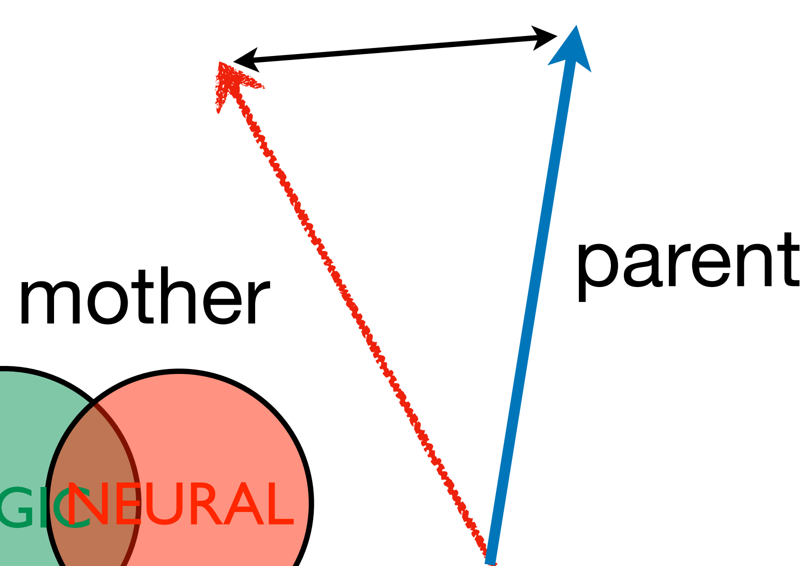
Soft unification

- 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 | bob = bob

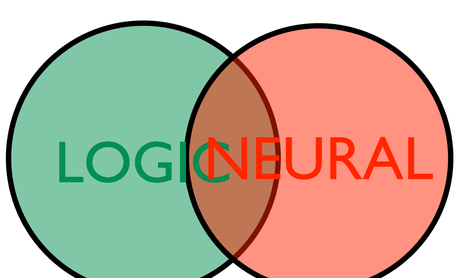
$\theta_1 = \{X = an\}$ | $\theta_2 = \{\}$



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

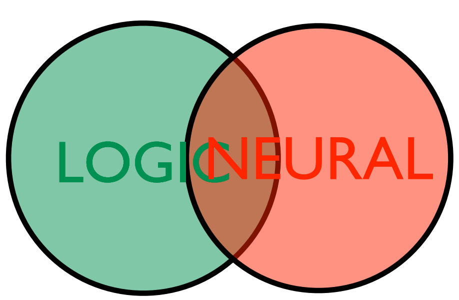


Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." Advances in Neural Information Processing Systems. 2017.

Differentiable rule learning

- Add parameterized rules
- r1(X,Y) :- r2(Y,X)
- r3(X,Y) :- r4(X,Y), r5(Y,Z)

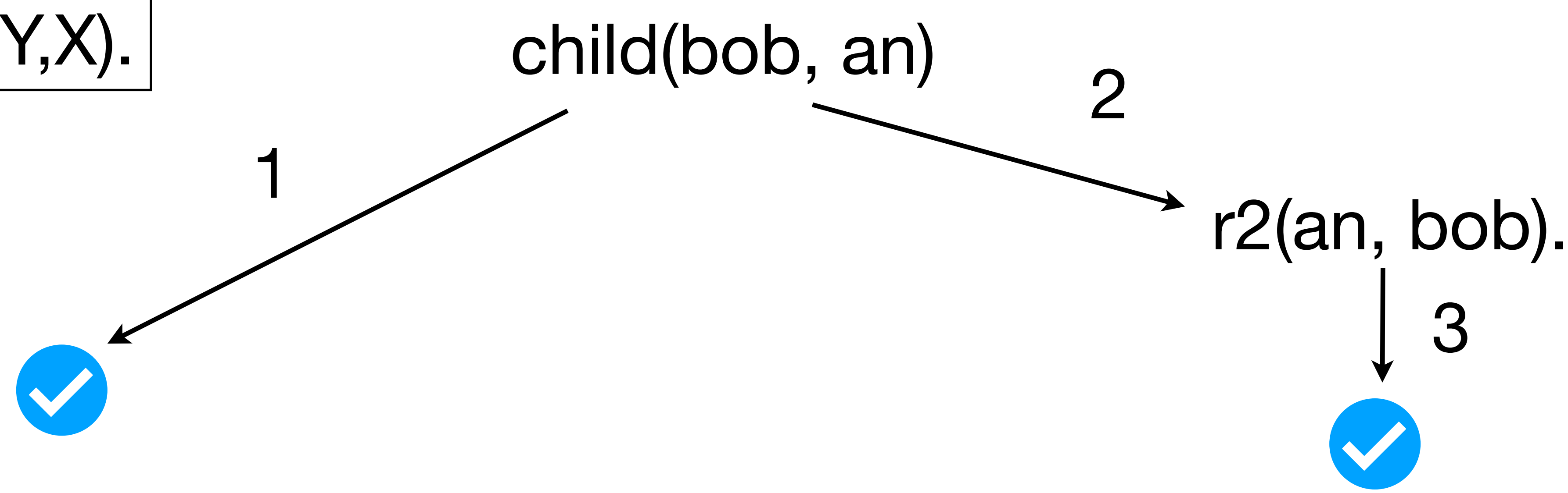
- Sub-symbols for r1, …, r5 move closer to other predicates
- For example, if r1 is close to parent and r2 is close to child, equivalent to:
- parent(X,Y) :- child(Y,X).



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).

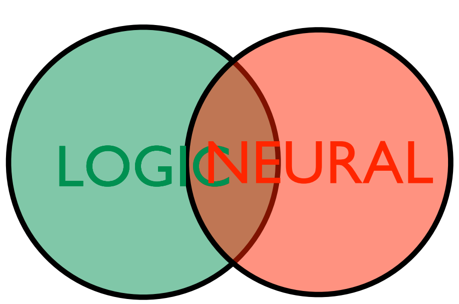


Unifications

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)

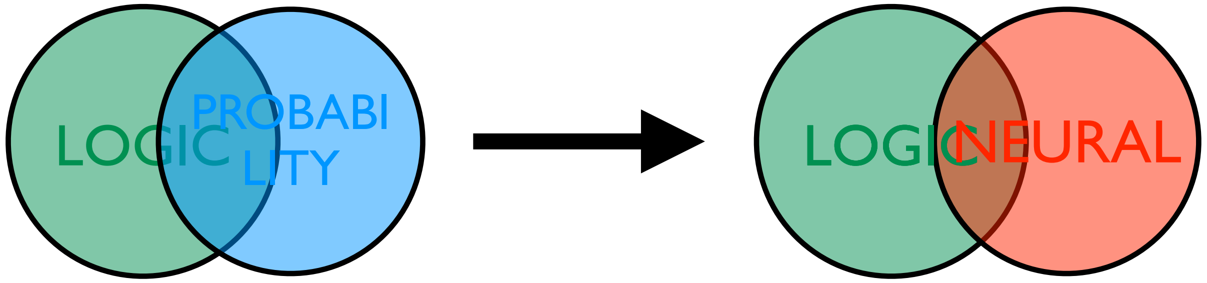


Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." Advances in Neural Information Processing Systems. 2017.

4. Symbolic vs sub-symbolic
Key Messages

- Entities are represented very differently in symbolic and sub-symbolic systems, but they are complementary
- NeSy systems can be categorized by how they use symbolic and sub-symbolic intermediate representations

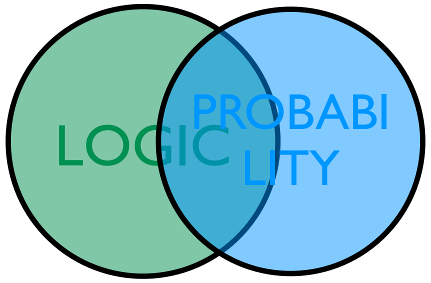
5. Structure vs parameter learning



5. Learning
Key Messages

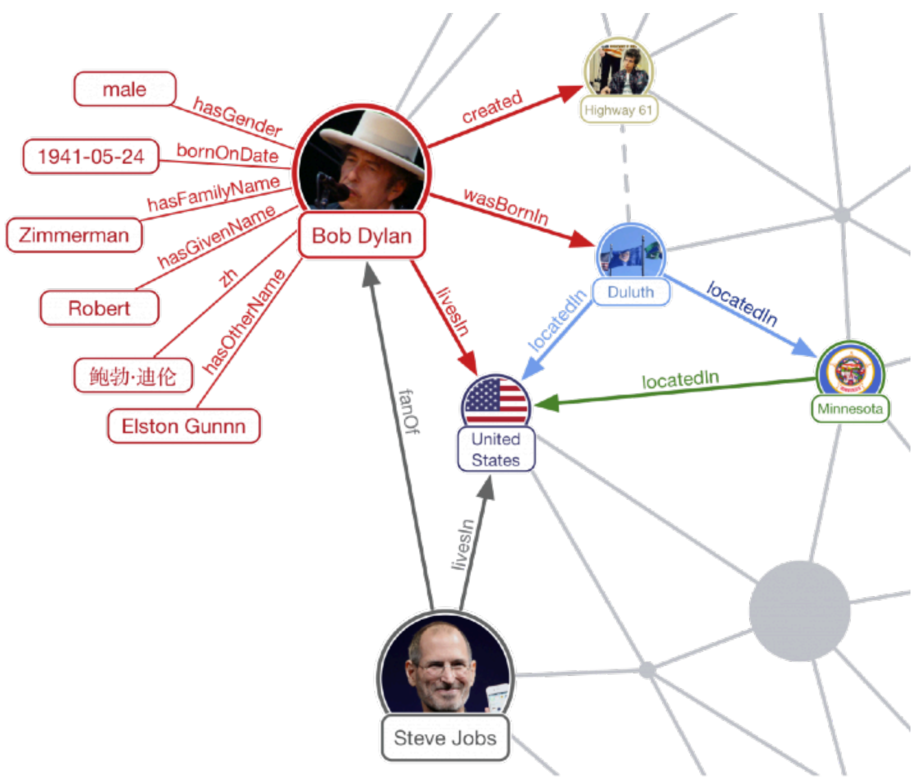
- Learning: finding logical formulas and estimating probabilities
- Structure learning: both formulas and probabilities
- Parameter learning: only probabilities
- Many flavours of learning in NeSy

5. Structure vs parameter learning



Learning in StarAI

Obtaining models from data



$\mapsto$

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).

LOGIC PROBABILITY

StarAI learning paradigms

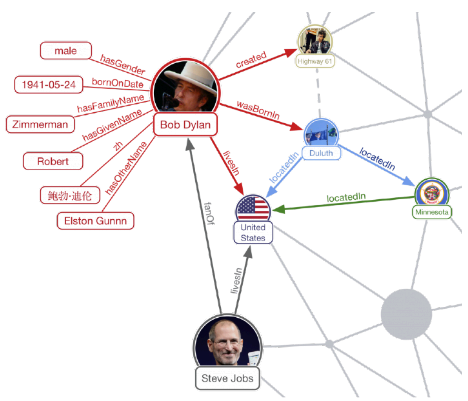
	Structure learning	**Parameter learning**
What is provided?	Data	Data and discrete structure
What is the learning goal?	Structure and parameters	Parameters

Learning types: Parameter learning

Learning the probabilities/weights of a specified model

Model (the formulas) are given

the goal of learning




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

↦

```
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).
```

LOGIC PROBABILITY

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: Parameter

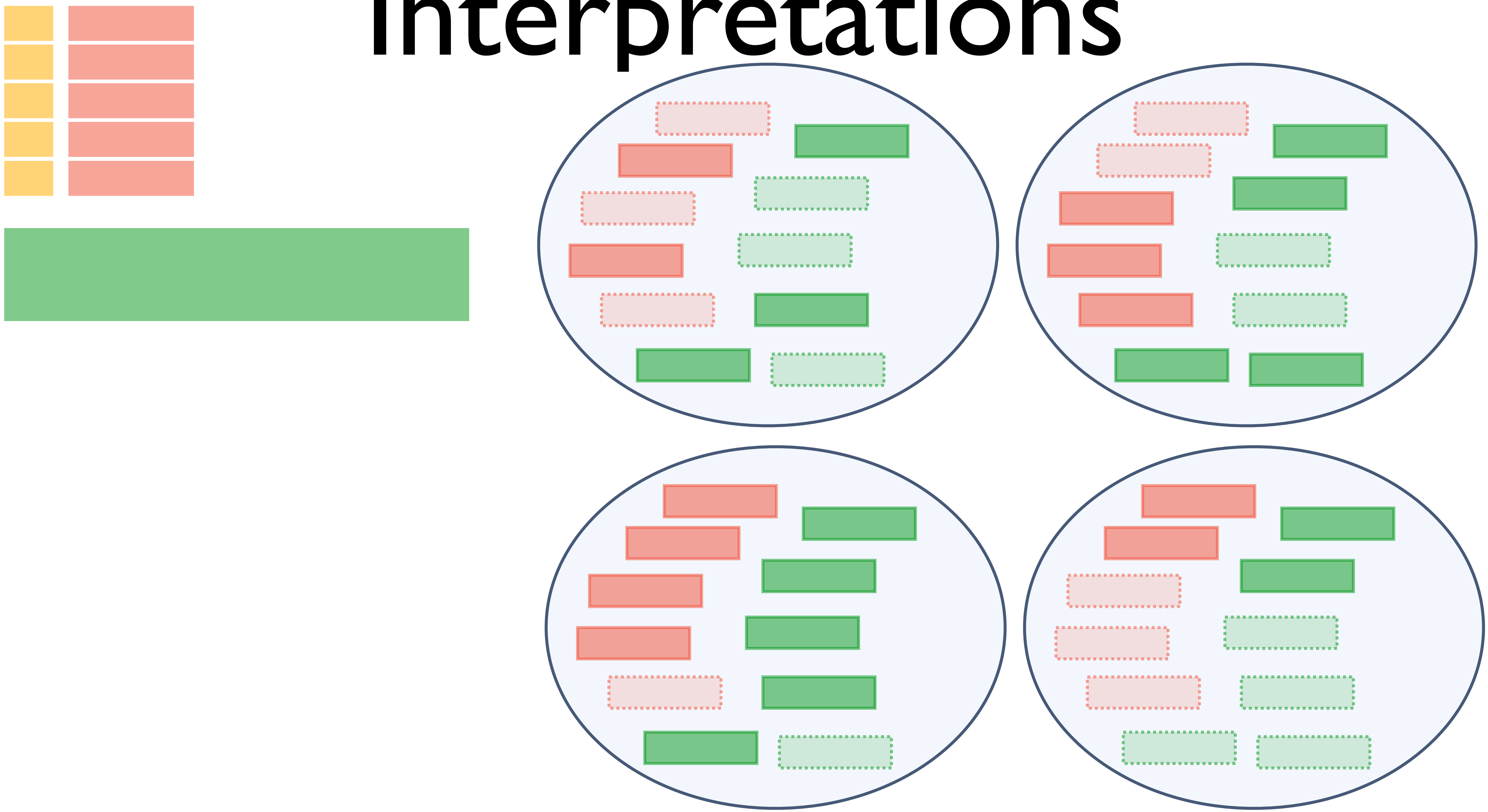
e.g., webpage classification model

for each ***CLASS1, CLASS2*** and each ***WORD***

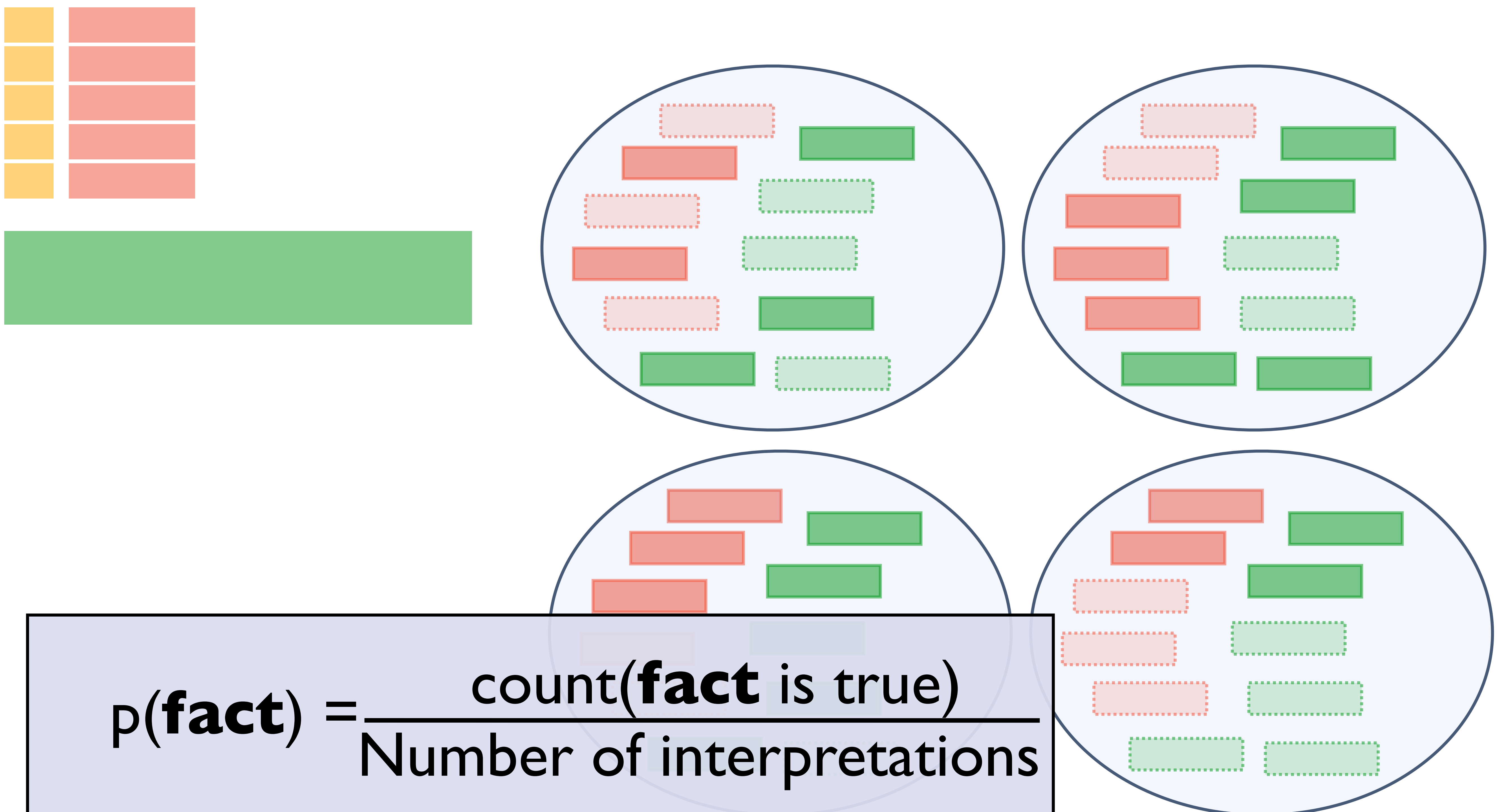
?? :: link_class(Source,Target,***CLASS1***,***CLASS2***).
?? :: word_class(***WORD***,***CLASS***).

```
class(Page,C)    :-   has_word(Page,W), word_class(W,C).

class(Page,C)         :-   links_to(OtherPage,Page),
class(OtherPage,OtherClass),
                           link_class(OtherPage,Page,OtherClass,C).
```

Sampling Interpretations

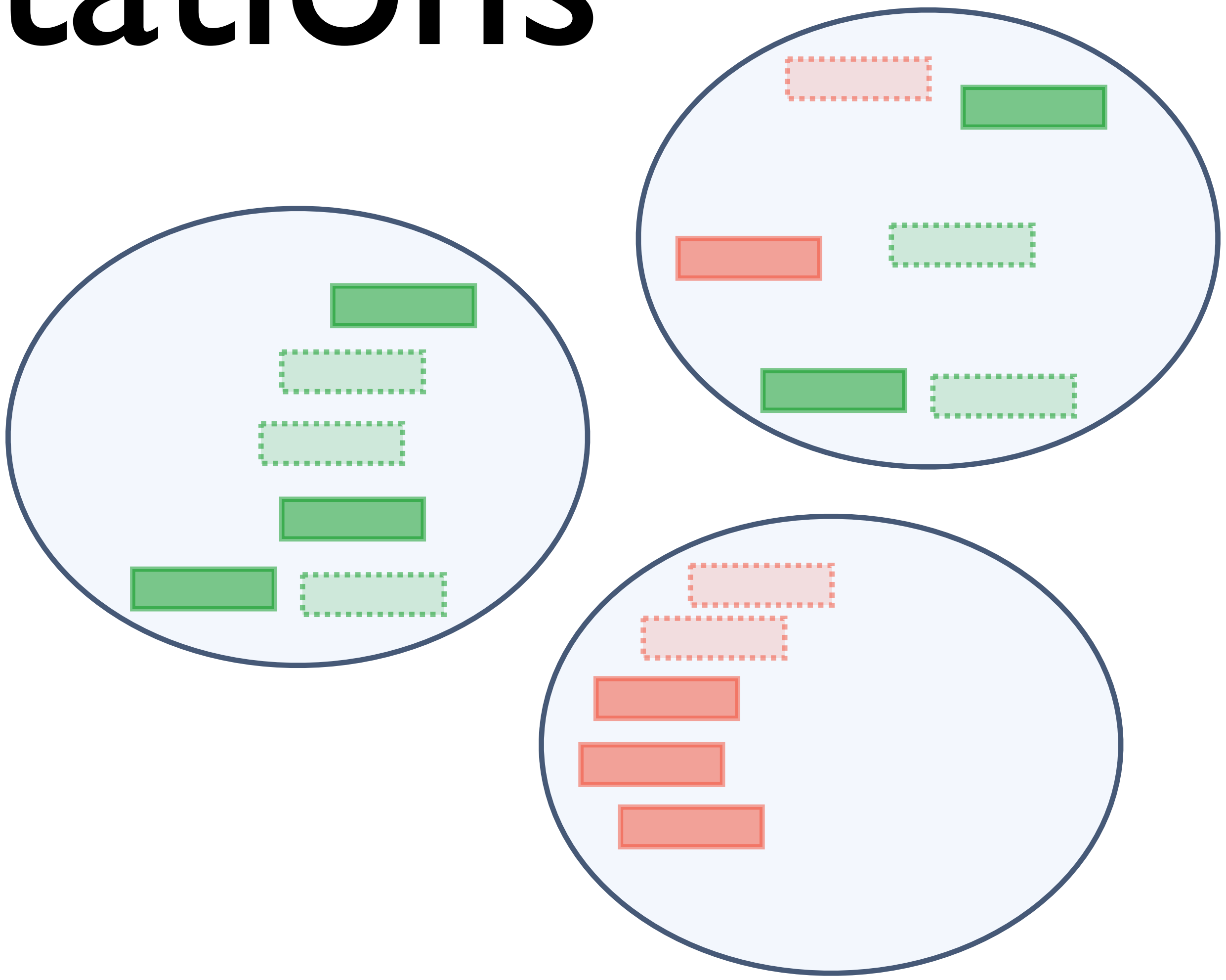


Parameter Estimation



$$p(\textbf{fact}) = \frac{\text{count}(\textbf{fact} \text{ is true})}{\text{Number of interpretations}}$$

Learning from partial interpretations



- Not all facts observed
- Soft-EM
- use **expected count** instead of **count**
- **P(Q |E) -- conditional queries !**

 [Gutmann et al, ECML 11; Fierens et al, TPLP 14]

Learning from partial interpretations

Key Points for parameter learning in SRL

— Parameters have to be tied together
— Similar to CNNs and HMMs
— Control the groundings

- Not all fa
- Soft-EM
- use **expected count** instead of **count**
- **P(Q |E) -- conditional queries !**

[Gutmann et al, ECML 11; Fierens et al, TPLP 14]

Markov Logic

- A Markov Logic Network (MLN) is a set of pairs (F, w) where
  - F is a formula in first-order logic
  - w is a real number
- An MLN defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w
- Probability of a world

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x) \right)$$

Weight of formula *i* (w_i)

No. of true groundings of formula *i* in *x* ($n_i(x)$)

erc

Parameter Learning

$$\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w\left[n_i(x)\right]$$

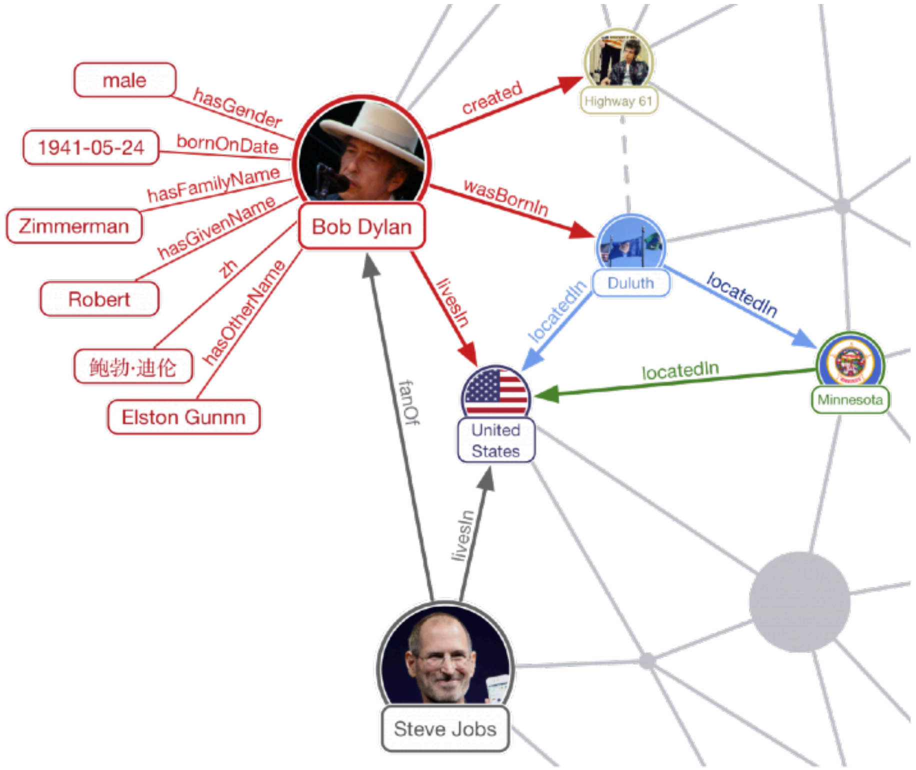
No. of times clause *i* is true in data

Expected no. times clause *i* is true according to MLN

Has been used for generative learning (Pseudolikelihood);
Many variations (also discriminative);
applications in networks, NLP, bioinformatics, …

Learning types: Structure learning

Finding the clauses/logical formulas of a model



↦

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

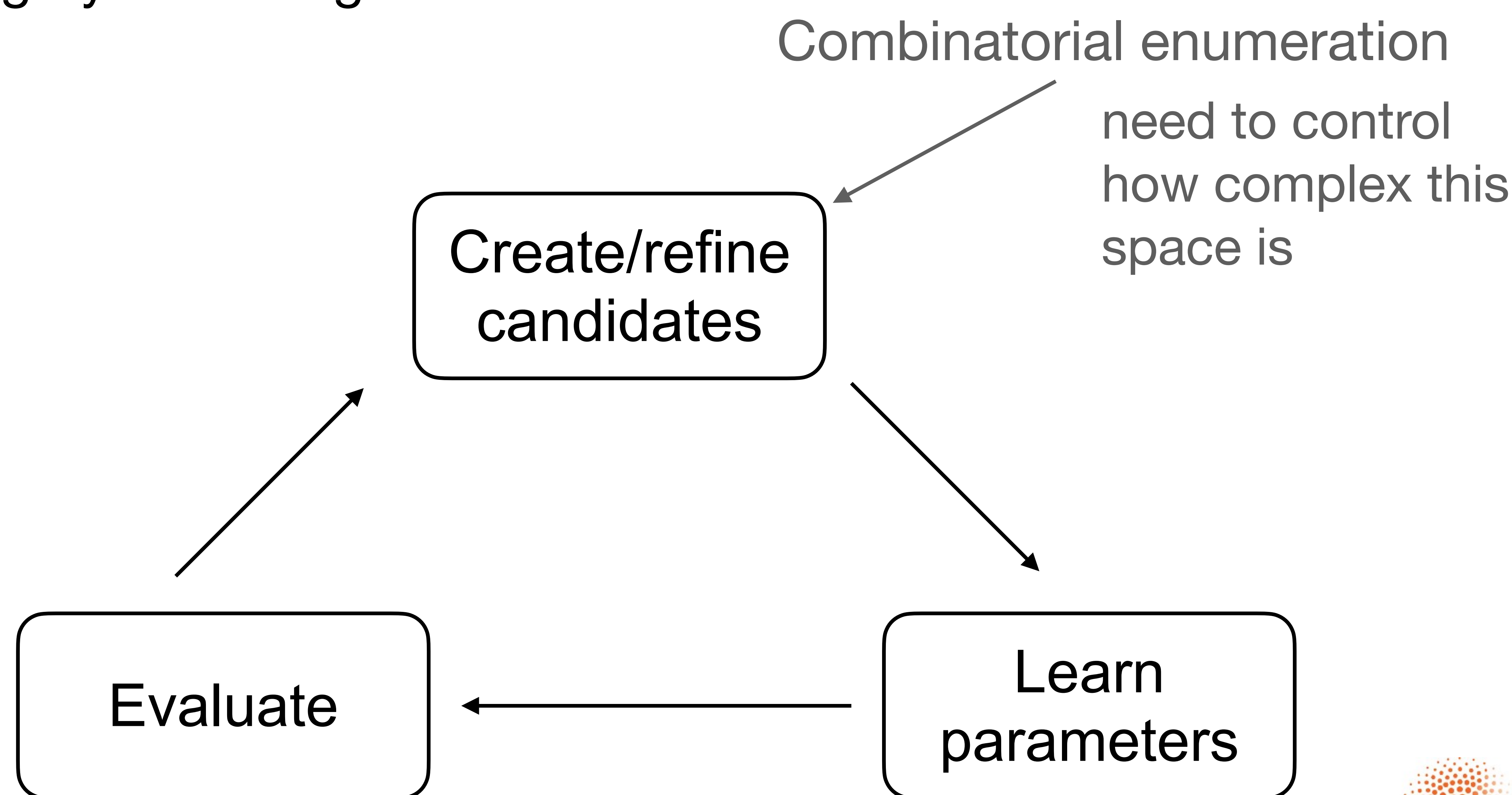
- specific target relation
- separate background knowledge

Generative

- no specific target relation
- learning generative process behind data

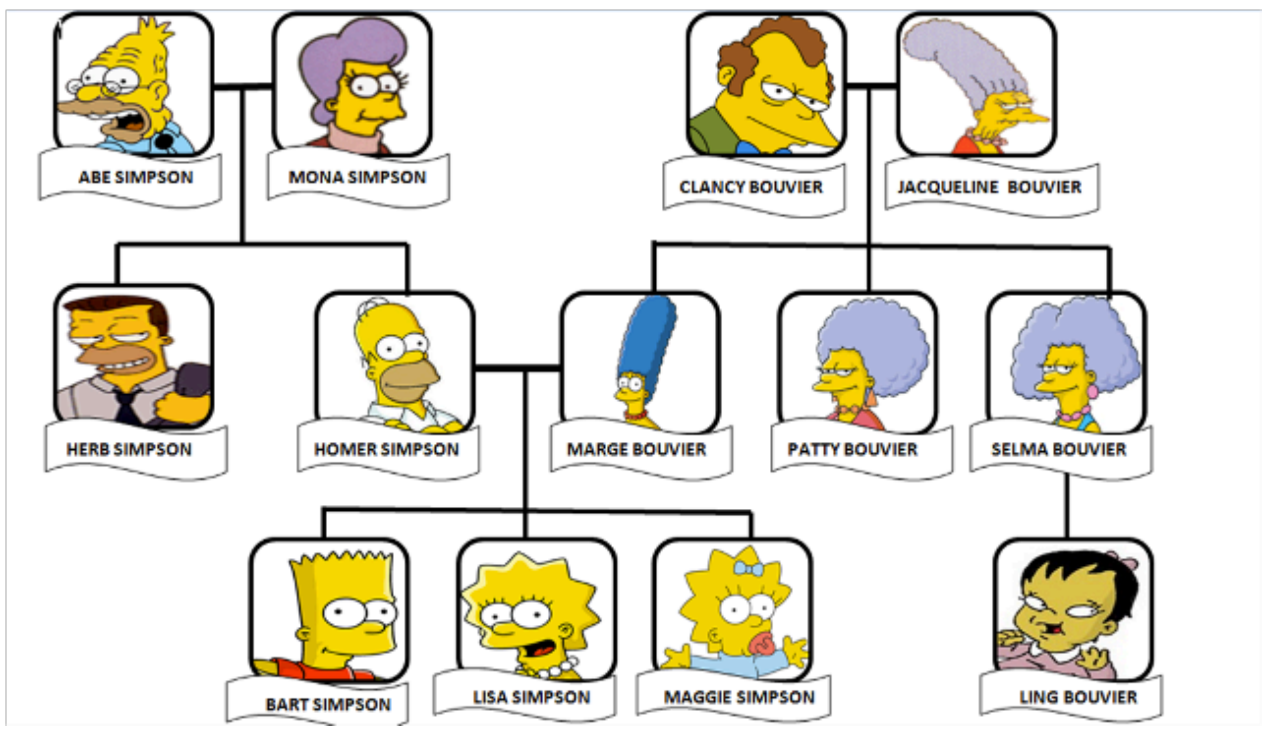
Learning types: Structure learning

Learning by searching



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)

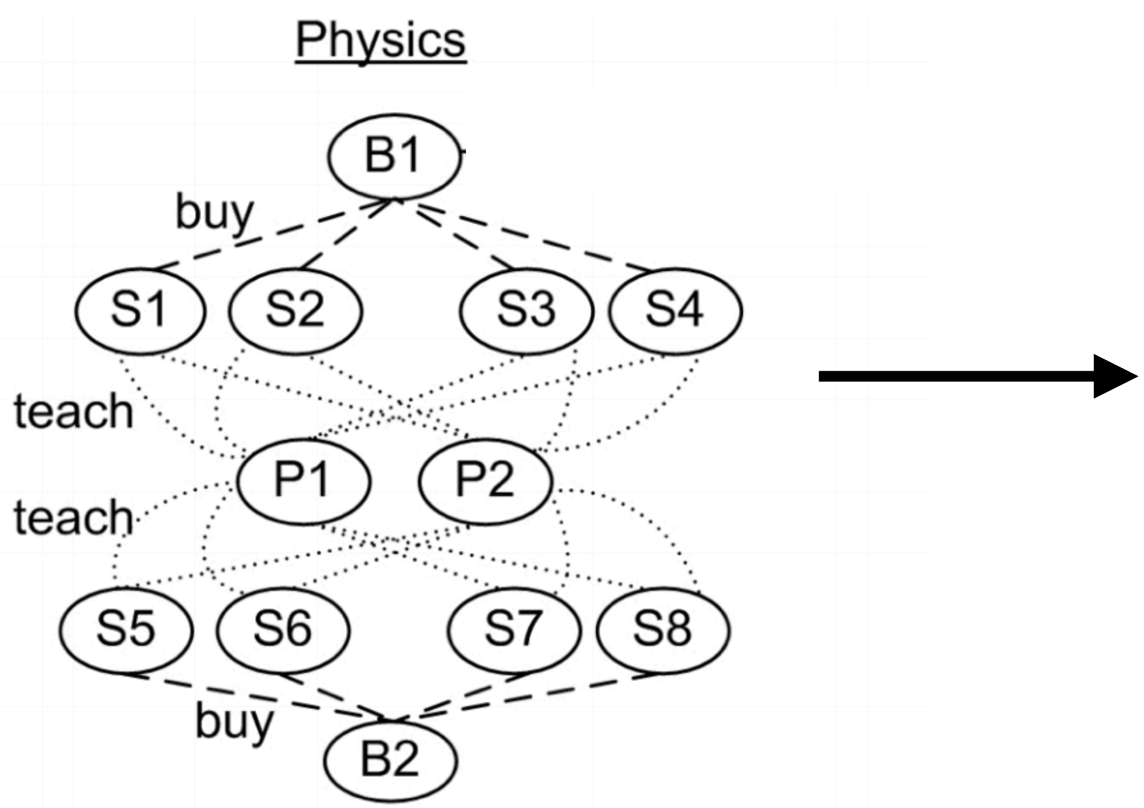
p:: grandparent(X,Y) ← mother(X,Z),mother(Z,Y)

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

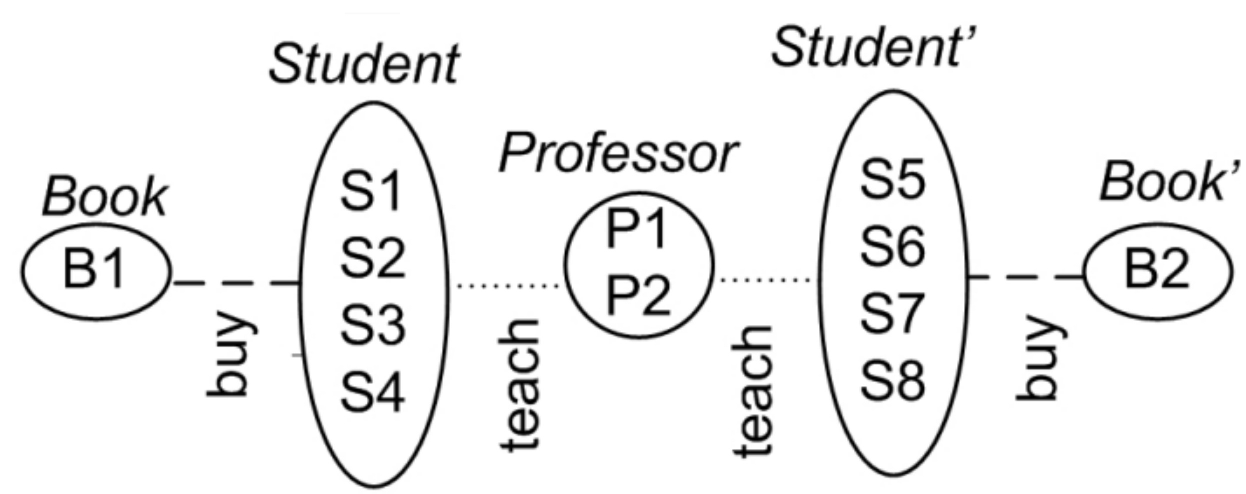
.....

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

erc

Learning in StarAI - overview

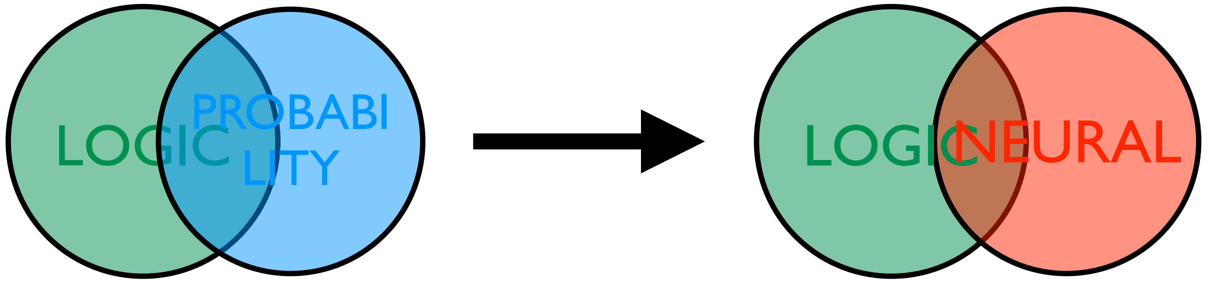
Structure learning

- ➕ 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

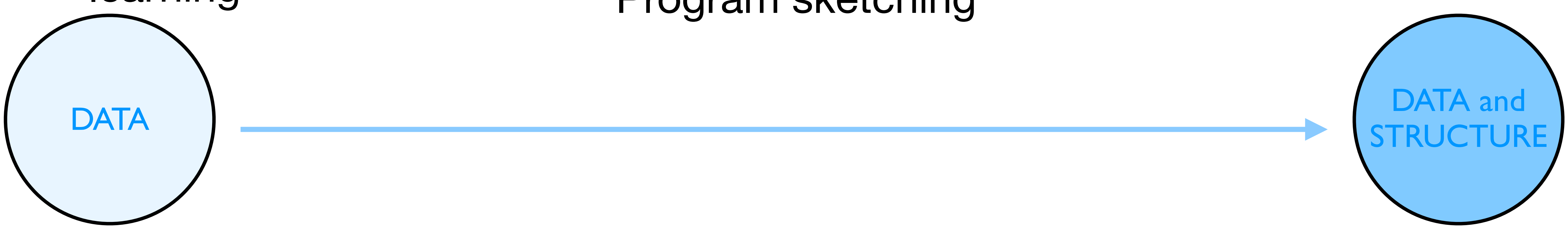
Soft patterns

Neural generation

Structure via parameter learning

Neurally-guided learning

Program sketching

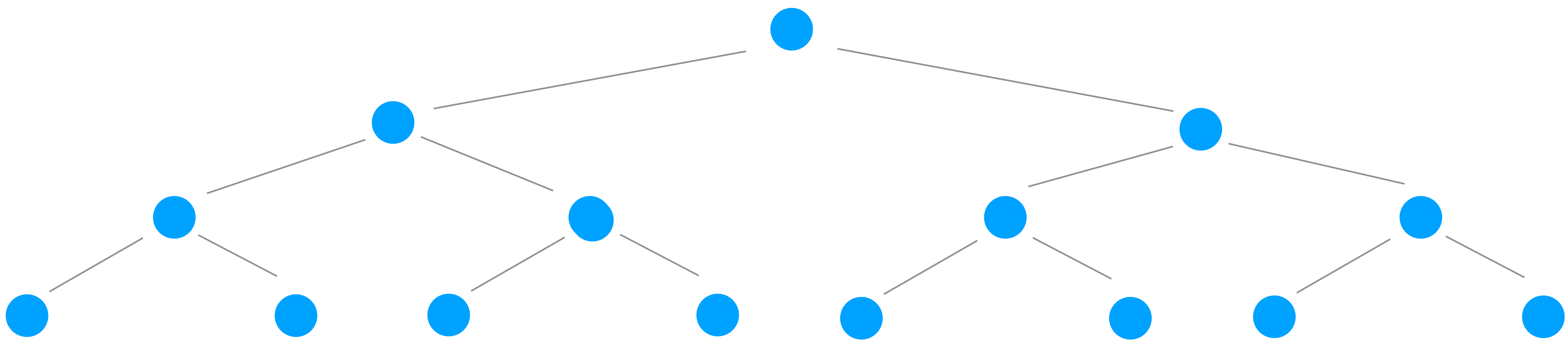


Structure learning

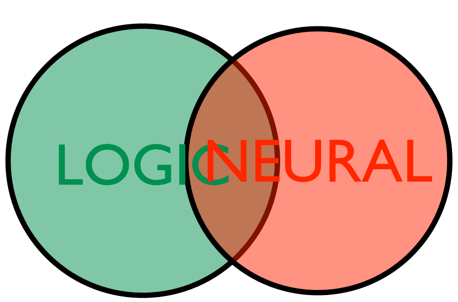
Parameter learning

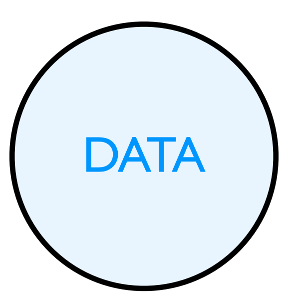
DeepCoder

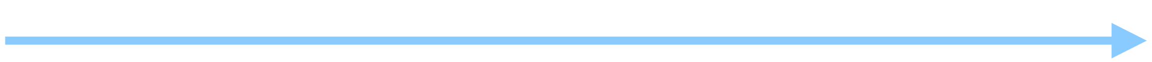
[Balog et al, 2017]

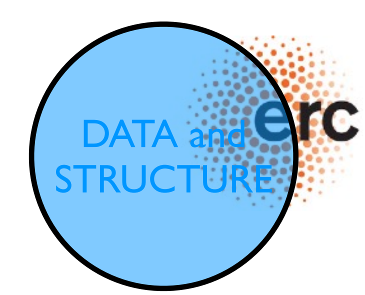


StarAI techniques search for clauses/rules systematically





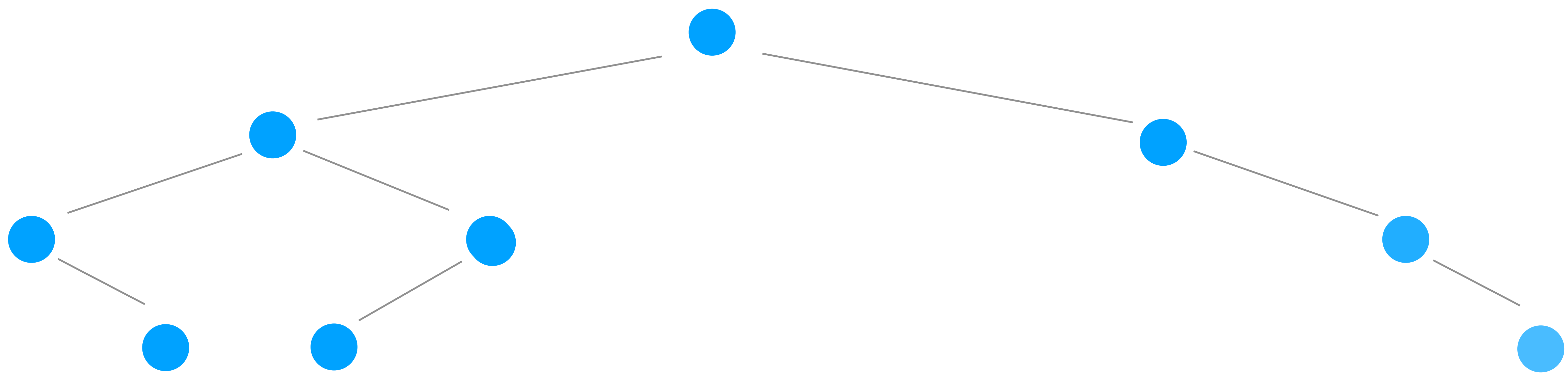




DeepCoder

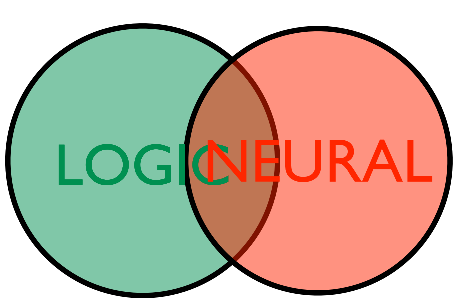
[Balog et al, 2017]

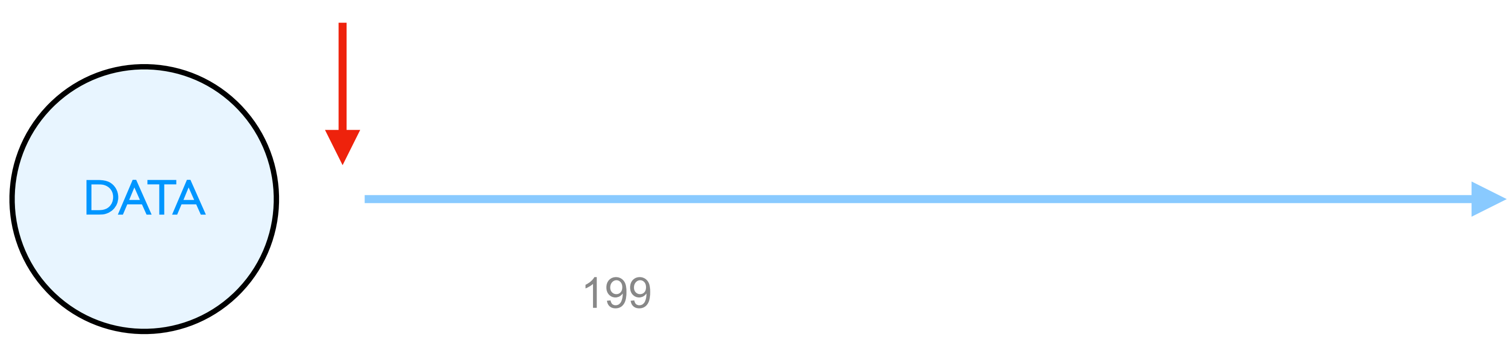
Preferences of learning ‘primitives’

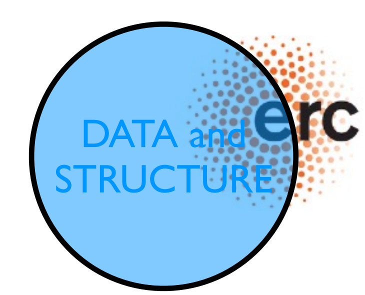


Explore the subpart of the space with primitives that are likely to solve the problem

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
.0	.0	.1	.0	.0	.0	.0	.0	**1.0**	.0	.0	**1.0**	.0	.2	.0	.0	**1.0**	**1.0**

SORT	REVERSE	TAKE	DROP	ACCESS	ZIPWITH	SCANL1	+	-	*	MIN	MAX	COUNT	MINIMUM	MAXIMUM	SUM
1.0	.7	.0	.1	.0	.4	.0	.0	.1	.0	.2	.1	.0	.0	.0	.0

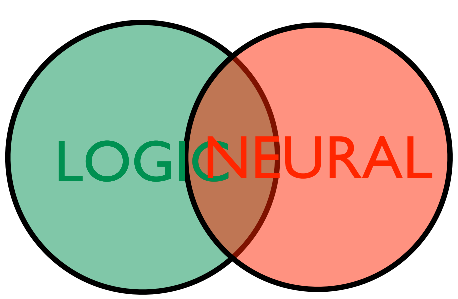
DreamCoder

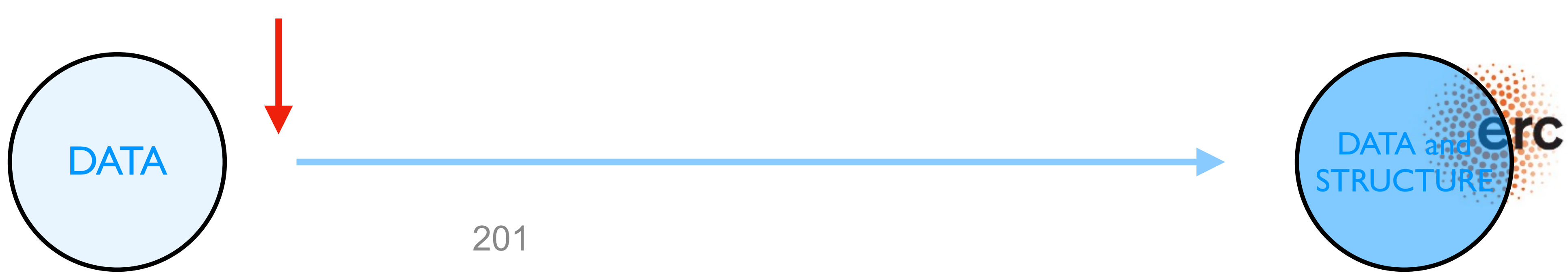
[Ellis et al, 2018]

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

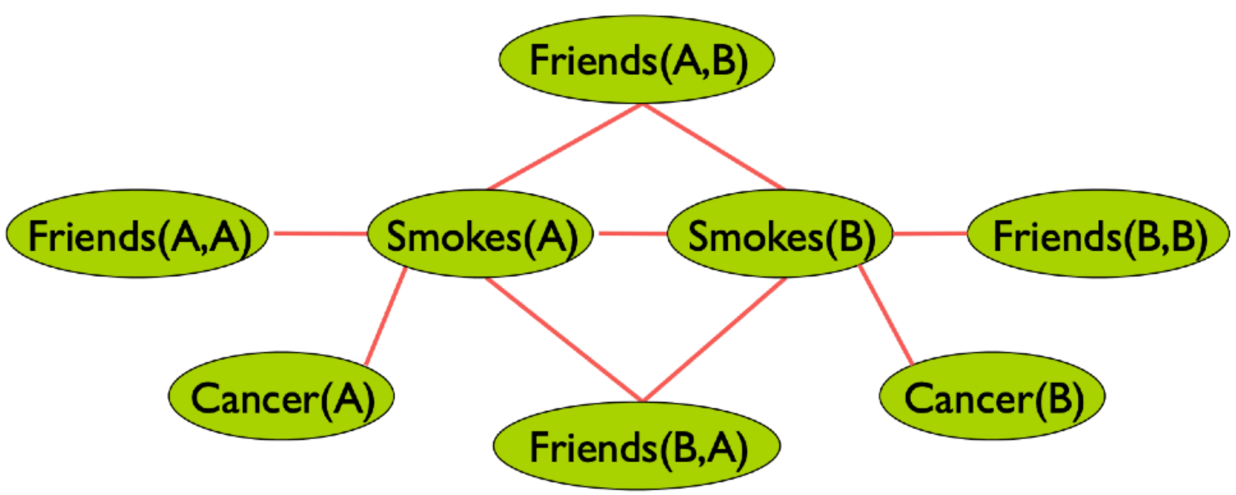




Neural Markov Logic Networks

[Marra et al, 2020]

MLNs can be interpreted as log-linear models



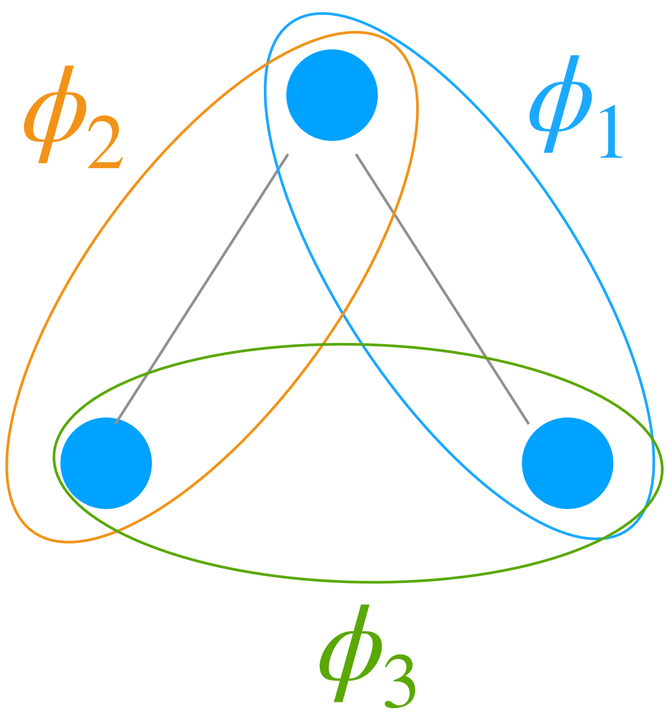
$$P(X = x) = \frac{1}{Z}\prod_i \phi_i(x_{\{i\}})^{n_i(x)}$$

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

Neural Markov Logic Networks

[Marra et al, 2020]

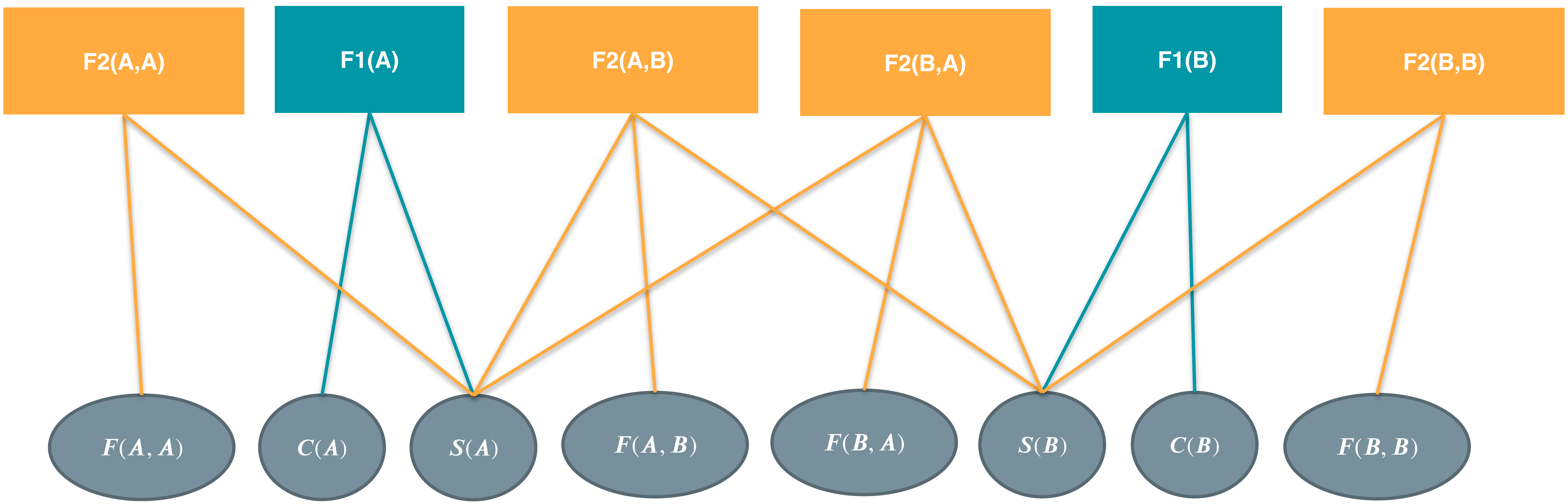
Learn neural potentials from fragments of data



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

potentials come from fragments of data (knowledge graph)

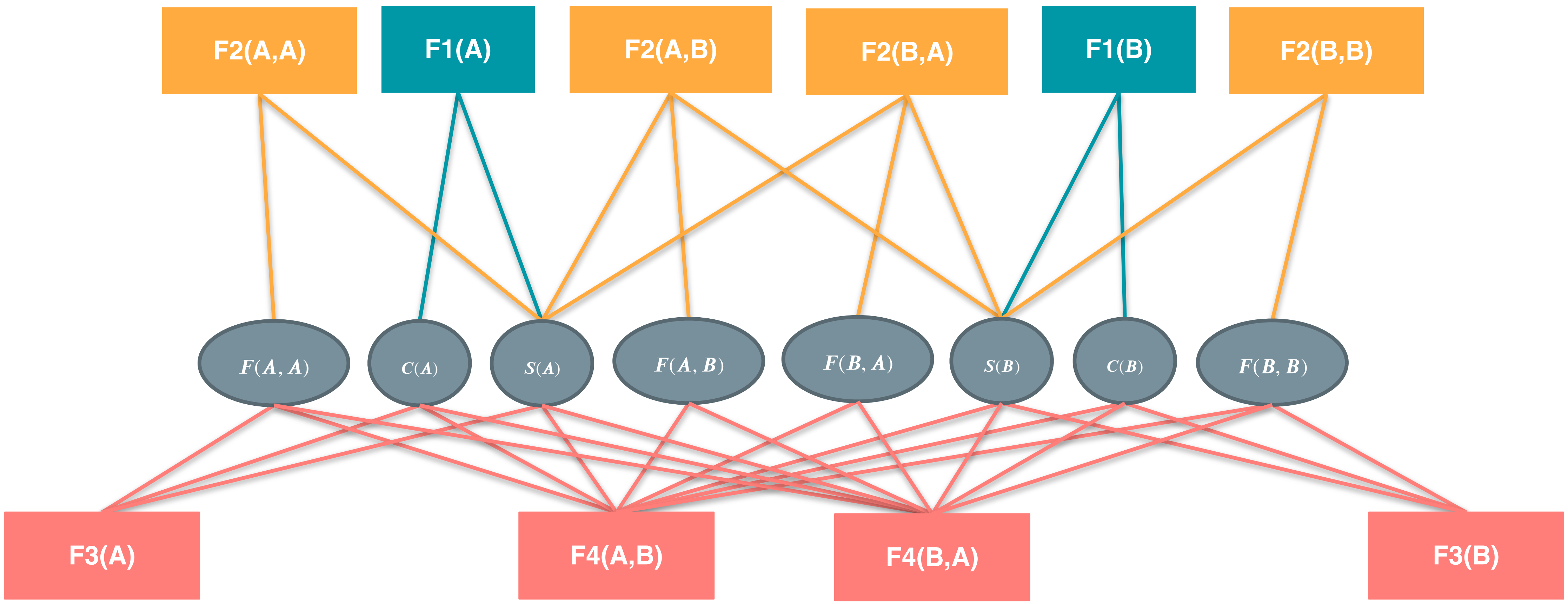
Markov Logic



represented as a factor graph

$$P(Interpretation) \propto \prod_i F_i(X, Y) = \prod_i exp(w_i \mathbb{1}(Interpretation \models F_i))$$

Neural Markov Logic



F3 and F4 are trainable factors

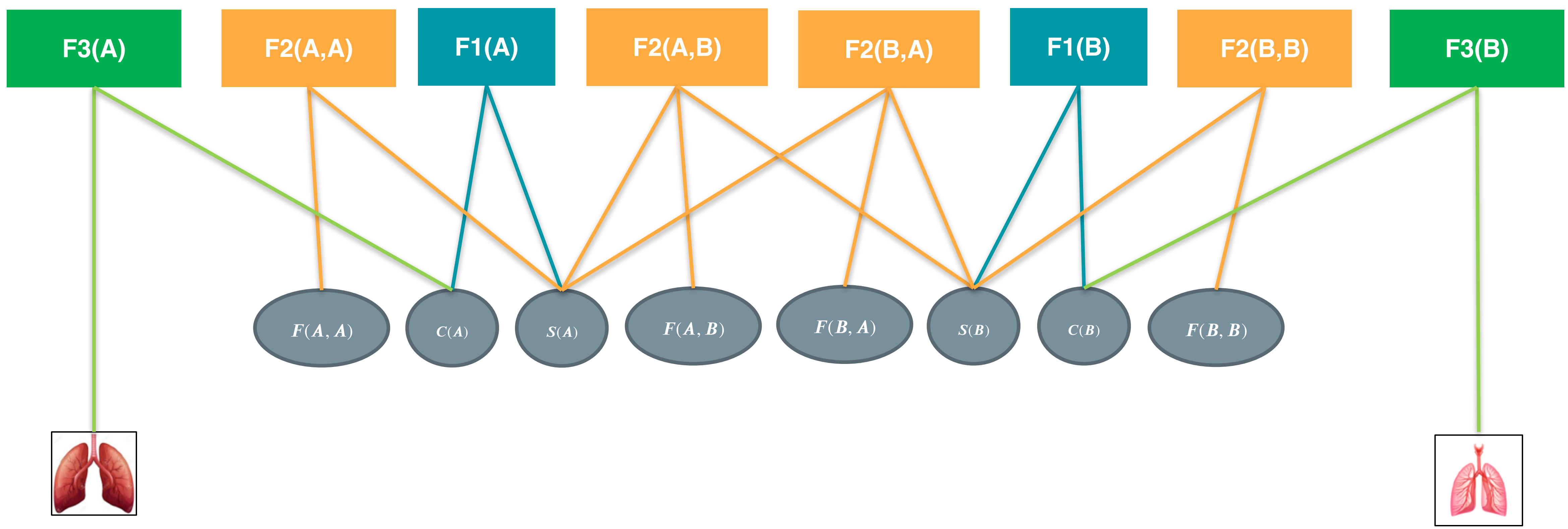
very much like in probabilistic graphical models and embeddings/hidden layers of a NN

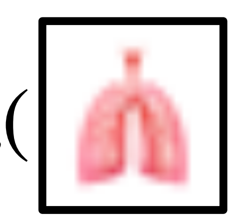
F3 and F4 correspond in a sense to the logical rules in the other factors
this gives a kind of structure learning
F3 and F4 will not be "interpretable"

Marra and Kuzelka

Relational Neural Machines

[Marra et al ECAI 20]



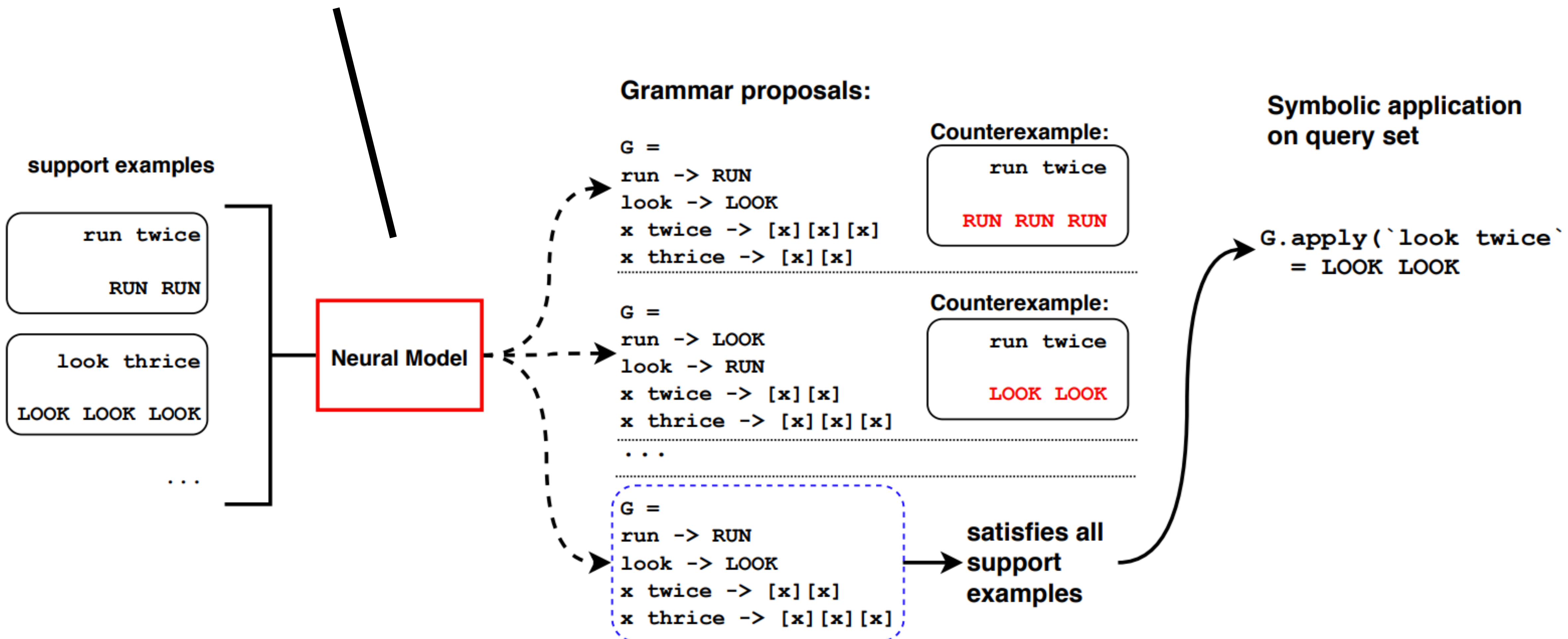
$F3\Big(\omega_{Cancer(Alice)},$  $\Big) = 1 - \Big(CNN_{cancer}($  $) - \omega_{Cancer(Alice)}\Big)^2$

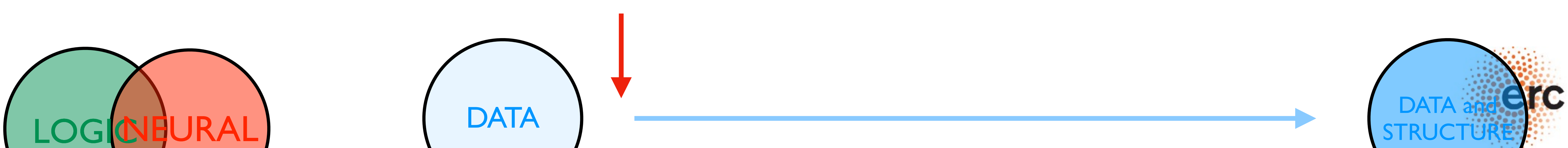
The Neural Network is trained to become a FACTOR (or a part of it)

Neural Generation

[Nye et al, 2020]

Neural model generates discrete structure





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
```

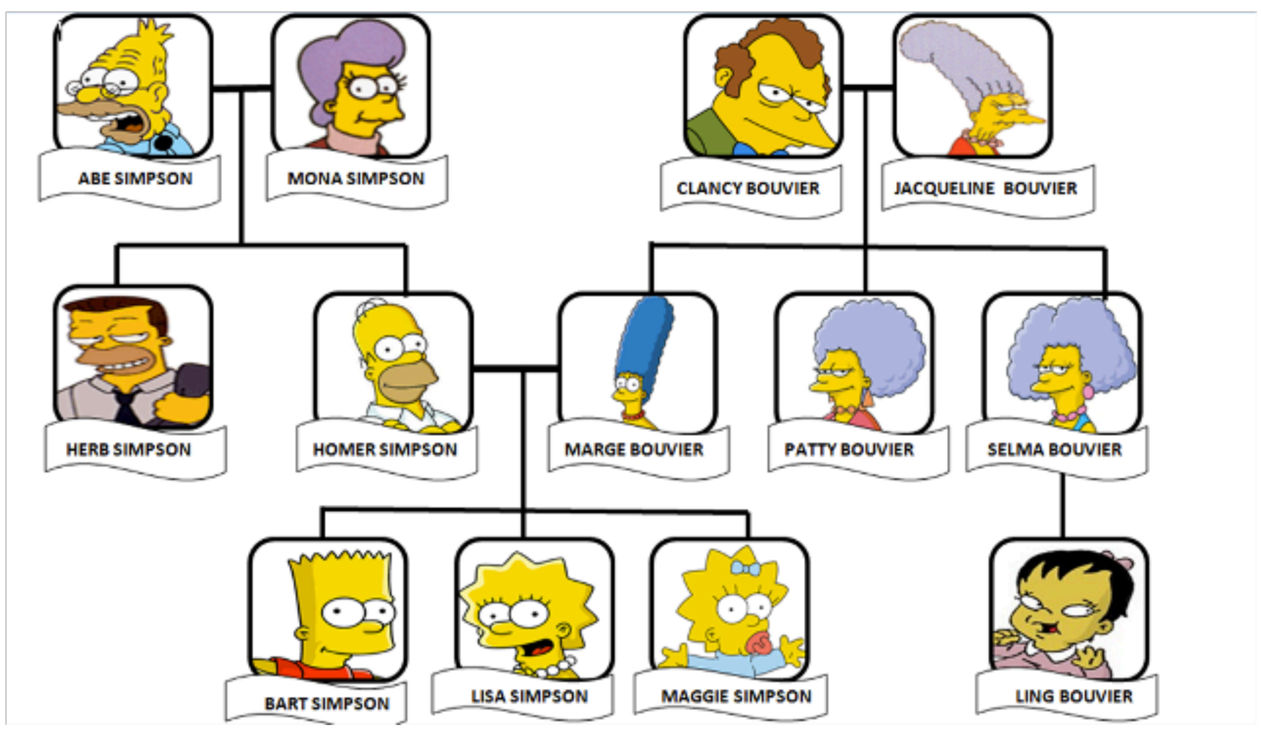
partial functionality
that needs to be learned

Structure learning via parameter learning

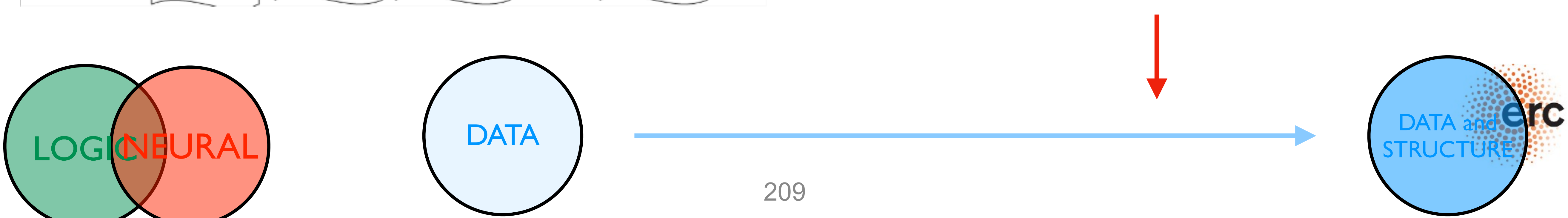
[Su et al, 2019]

Enumerate (lots of) logical formulas from templates

and learn their probabilities/weights



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

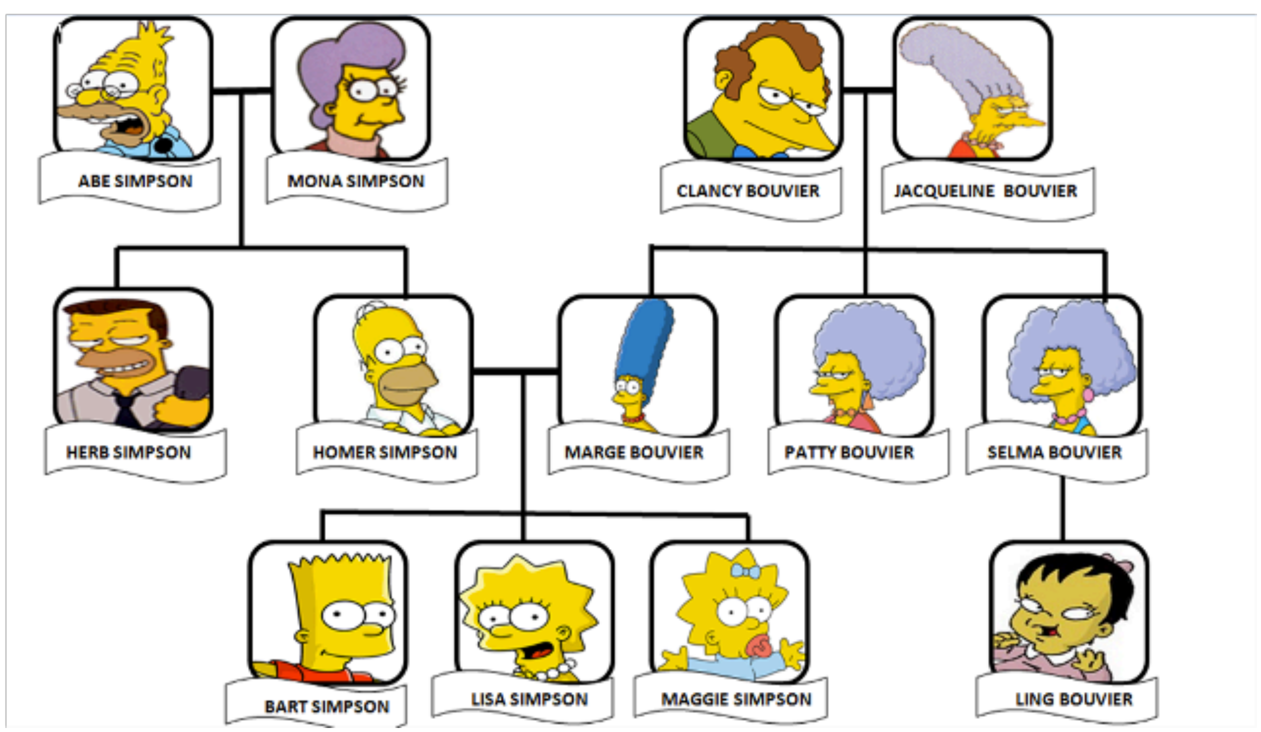


Program sketching

[Su et al, 2019]

Enumerate (lots of) logical formulas from templates

and learn their probabilities/weights



Program templates

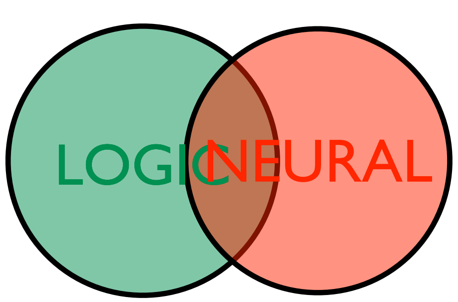
$T(X,Y) \leftarrow P(X,Y).$

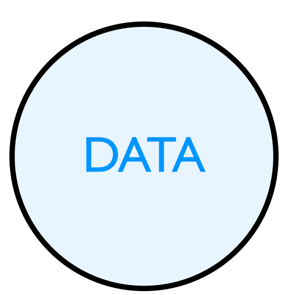
$T(X,Y) \leftarrow P(Y,X).$

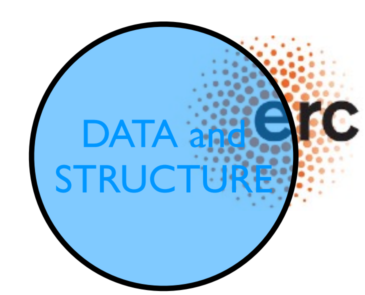
$T(X,Y) \leftarrow P(X,Z), Q(Z,Y).$

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) ← P(X,Y).
T(X,Y) ← P(Y,X).
T(X,Y) ← P(X,Z), Q(Z,Y).

Target: grandparent

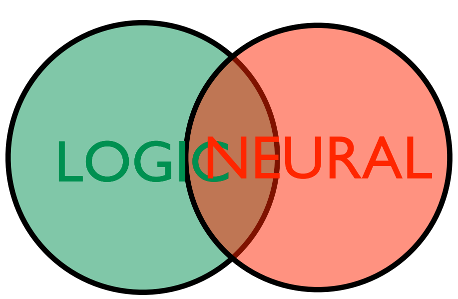
Other predicates: father, mother

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).
……

	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



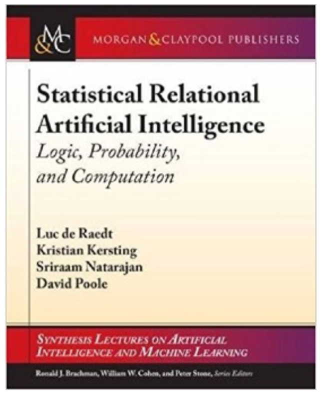
5. Learning
Key Messages

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

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

2. Directed vs Undirected the PGM / StarAI dimension



0.1 :: burglary.

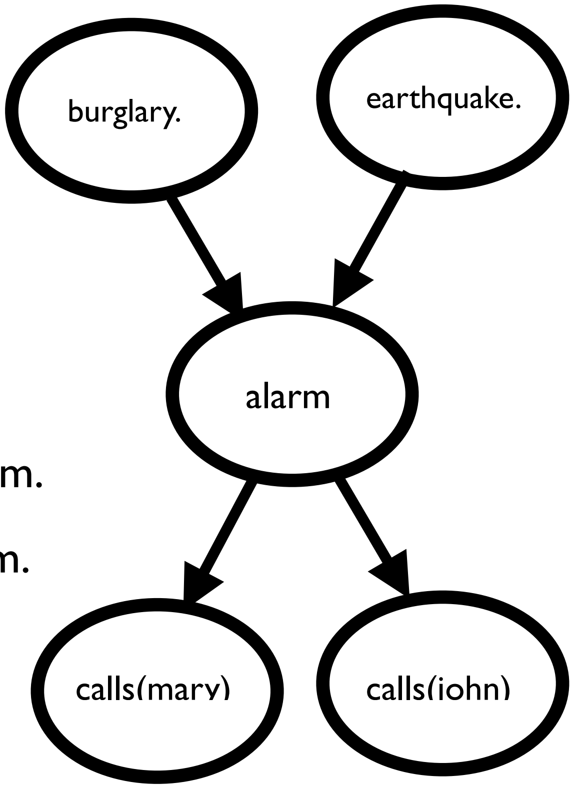
0.05 :: earthquake.

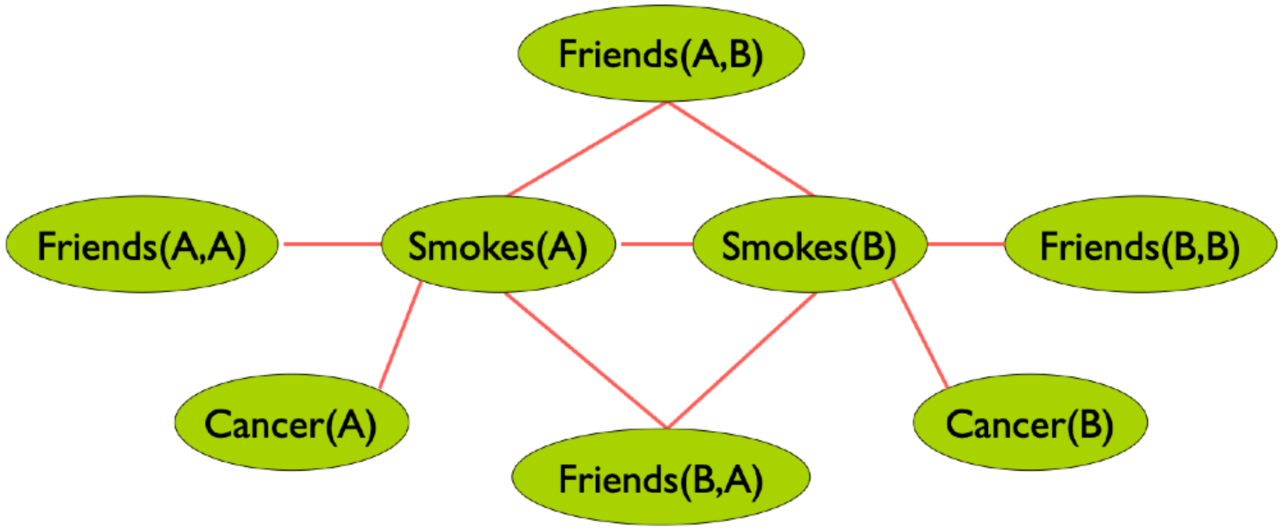
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.





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

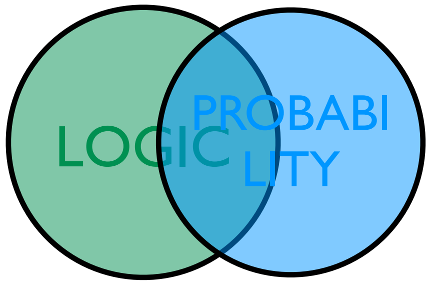
$$1.1 \quad \forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

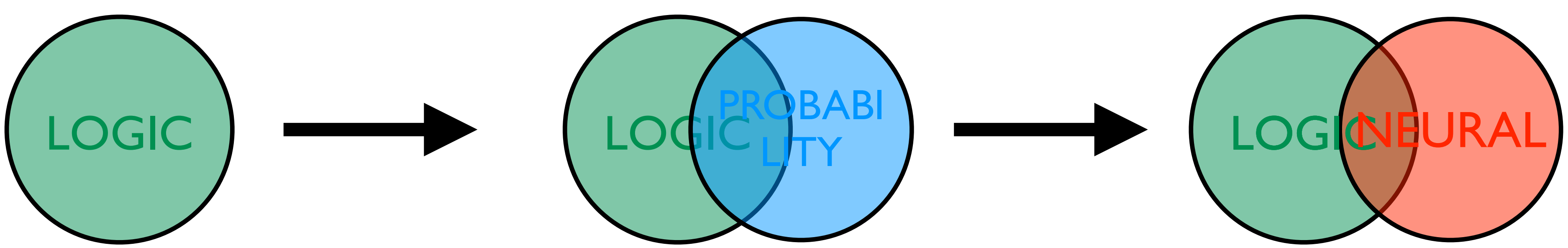
Markov Logic

undirected
Markov Net
model theoretic



key representatives

6. Semantics



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.

"42(47)"

42 is the property "being human" (or human/1)

47 is a constant referring to a particular human "Socrates"

human(Socrates) = True

Semantics

- We are interested in answering the following family of questions:

*Given a **sentence** of a propositional (or propositionalized through grounding) language, what is its **value?***

The nature of what **value** is differs in the different semantics.

Semantics

For simplicity,

- **labelling function** is the function ℓ_S that assigns, to the **sentence Q,** the value **v** according to **semantics S.**

$$\ell_S(Q) = v$$

e.g.

$\ell_B(human(socrates)) = True$

$\ell_F(tall(john)) = 0.8$

…

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

- Boolean truth values:

$$\{True, False\}$$

Three steps:

1. Truth values for propositions
2. Truth values for operators
3. Labelling formulas

LOGIC

Semantics in Boolean Logic

1. Providing the labels for propositions
L = {burglary, earthquake, hears_alarm(john)}

$$\ell_B(burglary) = True$$
$$\ell_B(earthquake) = False$$
$$\ell_B(hears\_alarm(john)) = True$$

*This is a **model** or a **possible world**, a "potential" assignment of truth values to all the propositional variables in the language.*

LOGIC

Semantics in Boolean Logic

2. Providing the semantics for operators

p	q	p∧q
T	T	T
T	F	F
F	T	F
F	F	F

$\ell_B^{\wedge}$

p	q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

$\ell_B^{\rightarrow}$

LOGIC

erc

Semantics in Boolean Logic

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

$$\ell_B(earthquake \wedge burglary) = \ell_B^{\wedge}(\ell_B(earthquake), \ell_B(burglary))$$

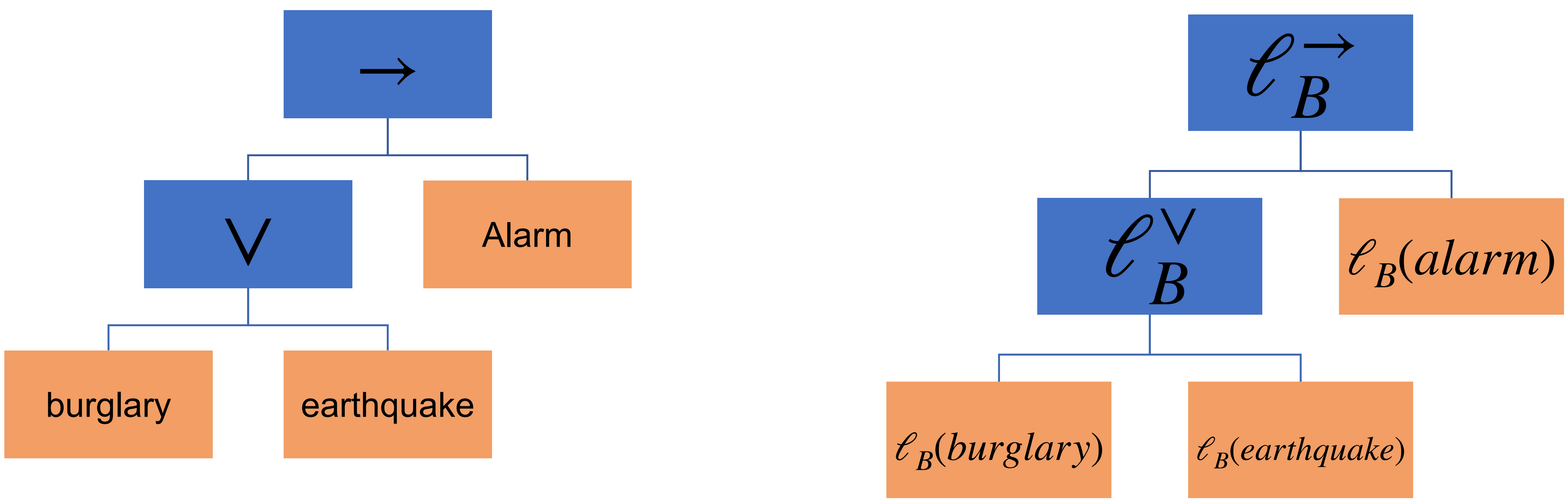
This recursive evaluation of formulas is said to be extensional approach.

LOGIC

erc

Semantics in Boolean Logic

- Consider: $(burglary \vee earthquake) \rightarrow alarm$

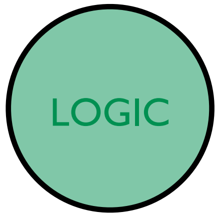


6. Semantics

Fuzzy logic

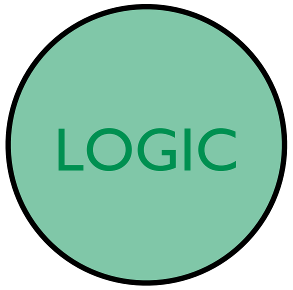
LOGIC

Semantics in Fuzzy Logic

- Still a pure **logic** semantics: 

- There are many fuzzy logics

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



Semantics in Fuzzy Logic

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

- Fuzzy truth/membership degrees:

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

Three steps:

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

LOGIC

Semantics in Fuzzy Logic

1. Providing the labels for propositions

L = {burglary, earthquake, hears_alarm(john)}

$$\ell_F(burglary) = 0.9$$

$$\ell_F(earthquake) = 0.1$$

$$\ell_F(hears\_alarm(john)) = 0.8$$

Note: $\ell_F(earthquake) = 0.1$ -> very mild earthquake,

($\neq$ probability of earthquake = 0.1)

LOGIC

fuzzy is a measure of intensity/vagueness not of uncertainty

erc

Semantics in Fuzzy Logic

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

They are the continuous version of truth tables!!

Semantics in Fuzzy Logic

- 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

They are the continuous version of truth tables!!

LOGIC

Semantics in Fuzzy Logic

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

$$\ell_F(burglary \rightarrow alarm) = \ell_F^{\rightarrow}(\ell_F(burglary), \ell_F(alarm))$$

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

e.g.

$\ell_F(burglary) = 0.9$, $\ell_F(alarm) = 0.3$,

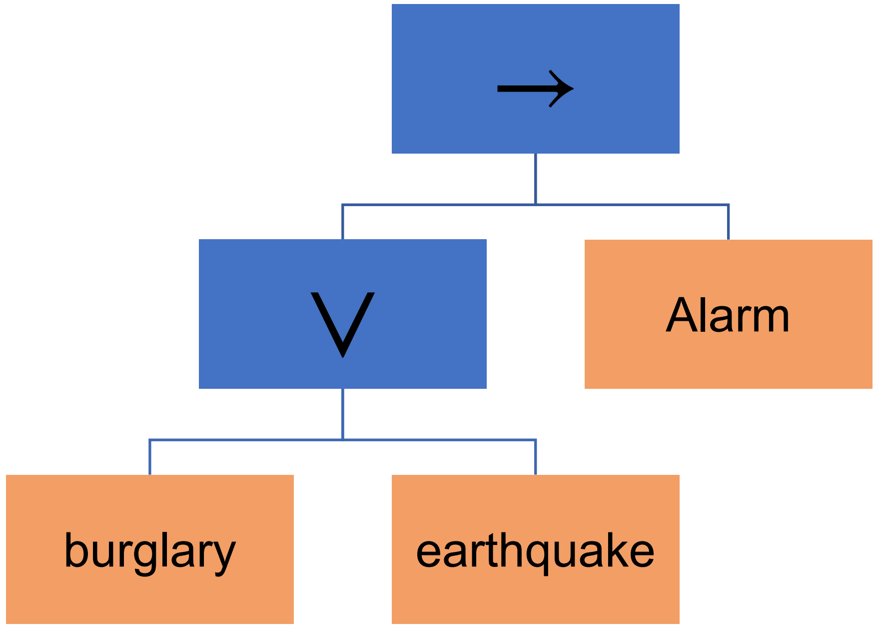
$\ell_F^{\rightarrow} = \min(1, 1 - x + y) = \min(1, 1 - 0.9 + 0,3) = 0.4$

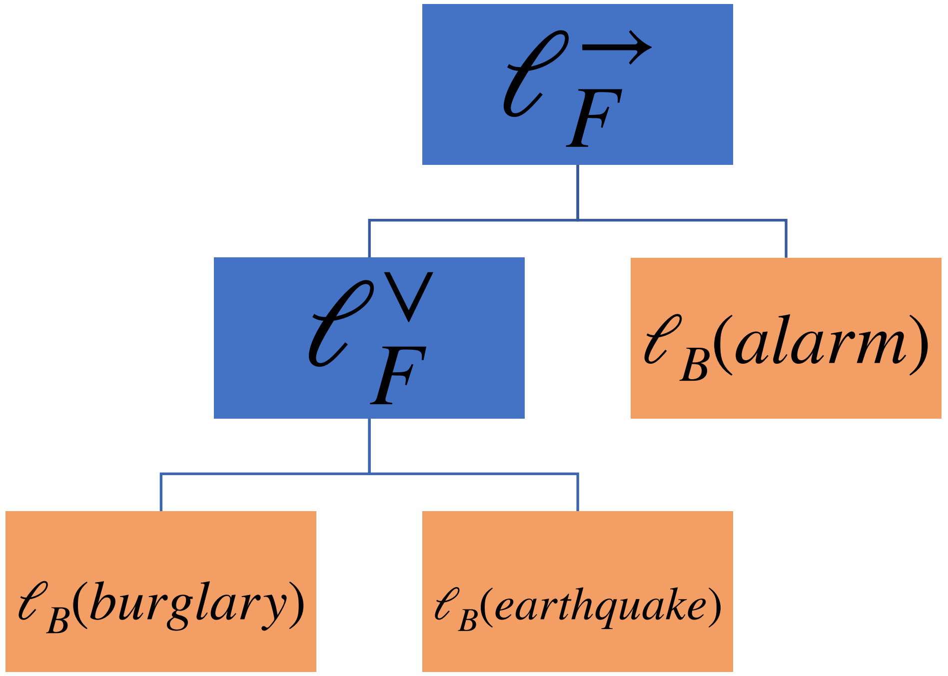
LOGIC

erc

Semantics in Fuzzy Logic

- Consider: $(burglary \lor earthquake) \to alarm$





LOGIC

Fuzzy Logic Semantics

- 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)

- But, $\ell_F(human(Socrates)) = 0.5$?????
- $\ell_F(bat(Socrates)) = 0.5$

LOGIC

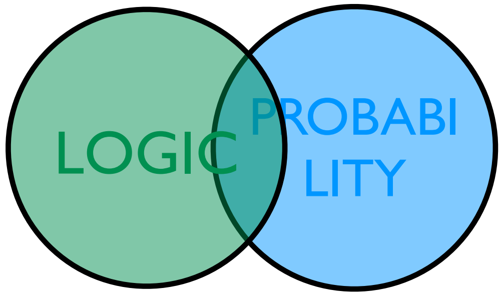
Fuzzy vs Boolean

- Fuzzy and Boolean have different properties
- When fuzzy is used as a "relaxation" (**fuzzification)** of Boolean **undesired effects** can happen.
- Suppose: $A \vee B \vee C \vee D \vee E = 1$
- Satisfying assignments (Lukasiewicz)
  - $A = B = C = D = E = 1$ (all true)
  - $A = 1, \quad B = C = D = E = 0$ (at least one true)
  - $A = B = C = D = E = 0.2$

LOGIC

Semantics

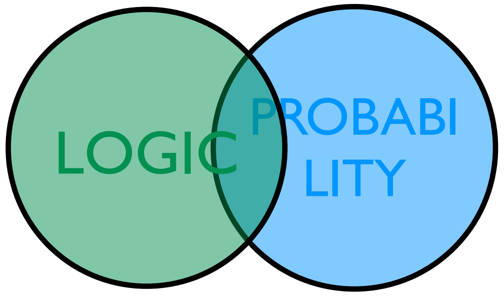
Probabilistic logic



Probabilistic Logic Semantics

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

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



Probabilistic Logic Semantics

Two steps:

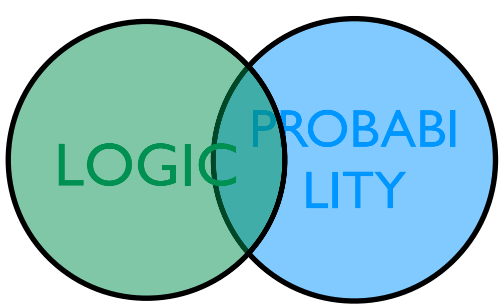
- Define a **probability distribution over interpretations / worlds (i.e. boolean semantics)**

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

(E.g. $p(\ell_B(burglary) = True, \ell_B(earthquake) = False, \ldots)$

- Define a **the probability of sentence Q of L:**

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



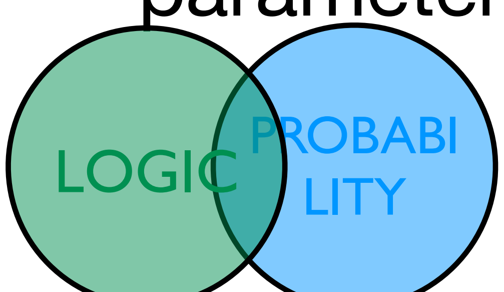
Probabilistic Logic Semantics

Problog

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

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

parameters = the labels for propositions (i.e. probabilistic facts)



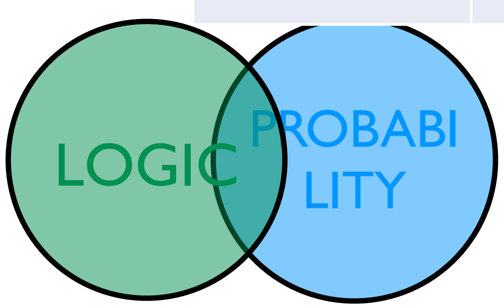
Probabilistic Logic Semantics
Problog

e.g. in ProbLog:

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

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
Markov Logic

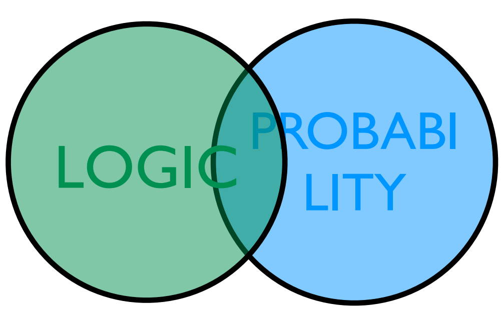
1.5 : calls(Mary) <- hears_alarm(Mary), alarm

2.0 : alarm <- earthquake

0.5 : alarm <- burglary

Weight formula   1 if α is True otherwise 0

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



Probabilistic Logic Semantics

Markov Logic

1.5 : calls(Mary) <- hears_alarm(Mary), alarm

2.0 : alarm <- earthquake

0.5 : alarm <- burglary

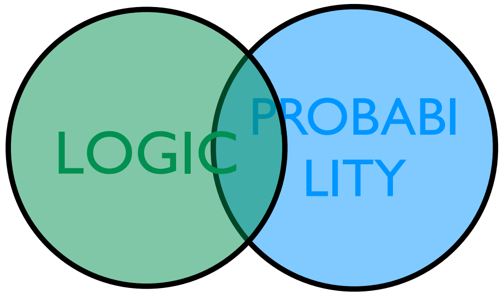
B	E	A	H	C	p	
T	F	T	T	T	0.05	$\propto \exp(1.5 + 2.0 + 0.5)$
T	F	T	T	F	0.01	$\propto \exp(0 + 2.0 + 0.5)$
…	…	…	…	…	…	

Probabilistic Logic Semantics

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

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

WMC - Weighted Model Counting
(for both ProbLog and Markov Logic)



Probabilistic Logic Semantics

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

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

Query = burglary ^ hears_alarm(john)

$$Q = B \wedge H$$

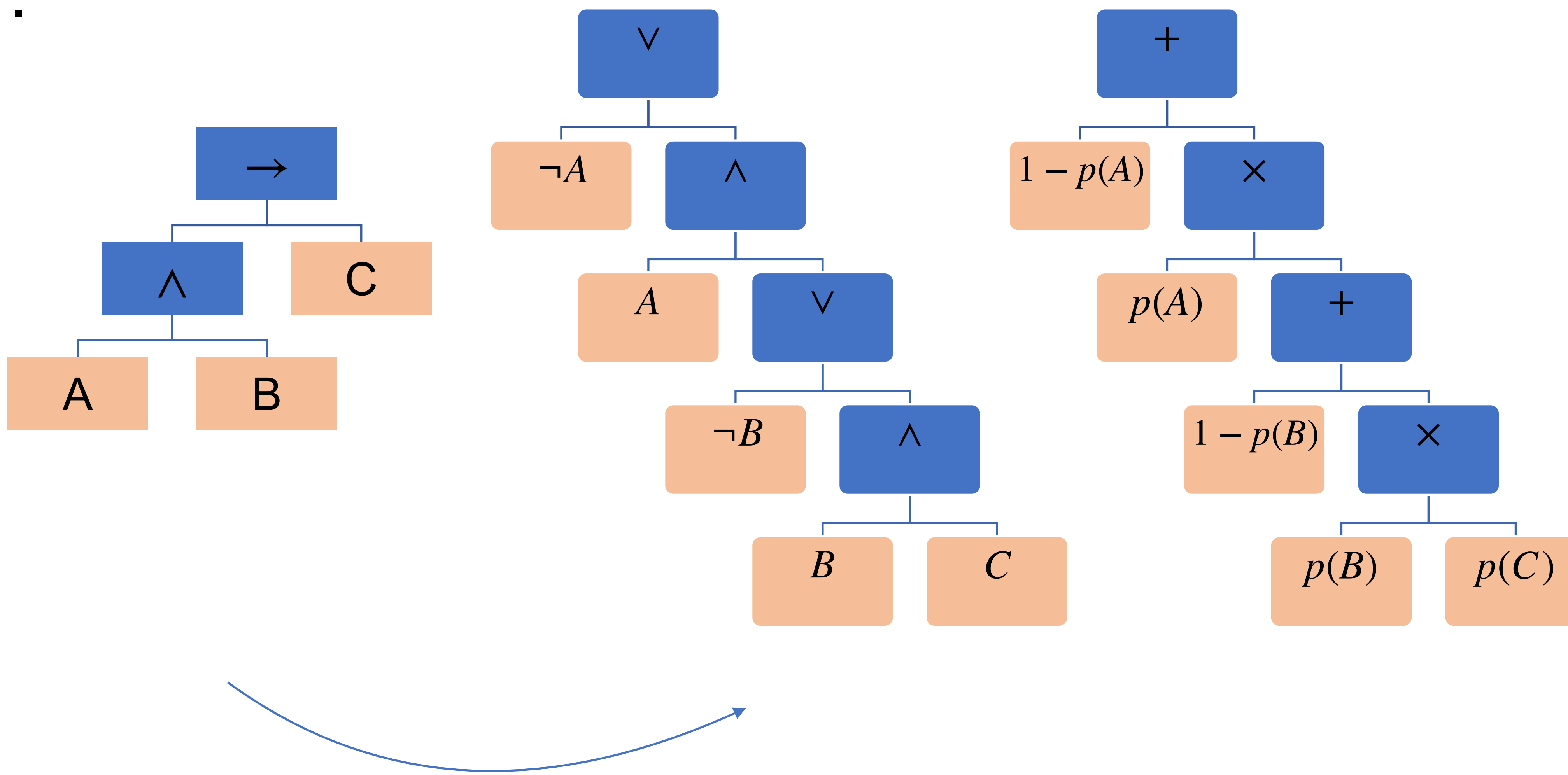
$$p(Q) = 0.06$$

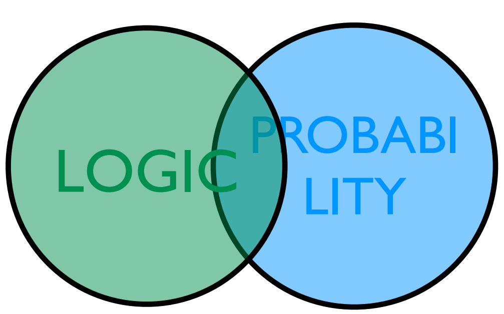
Probabilistic Logic Semantics

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

- Consider:

$(A \wedge B) \to C$



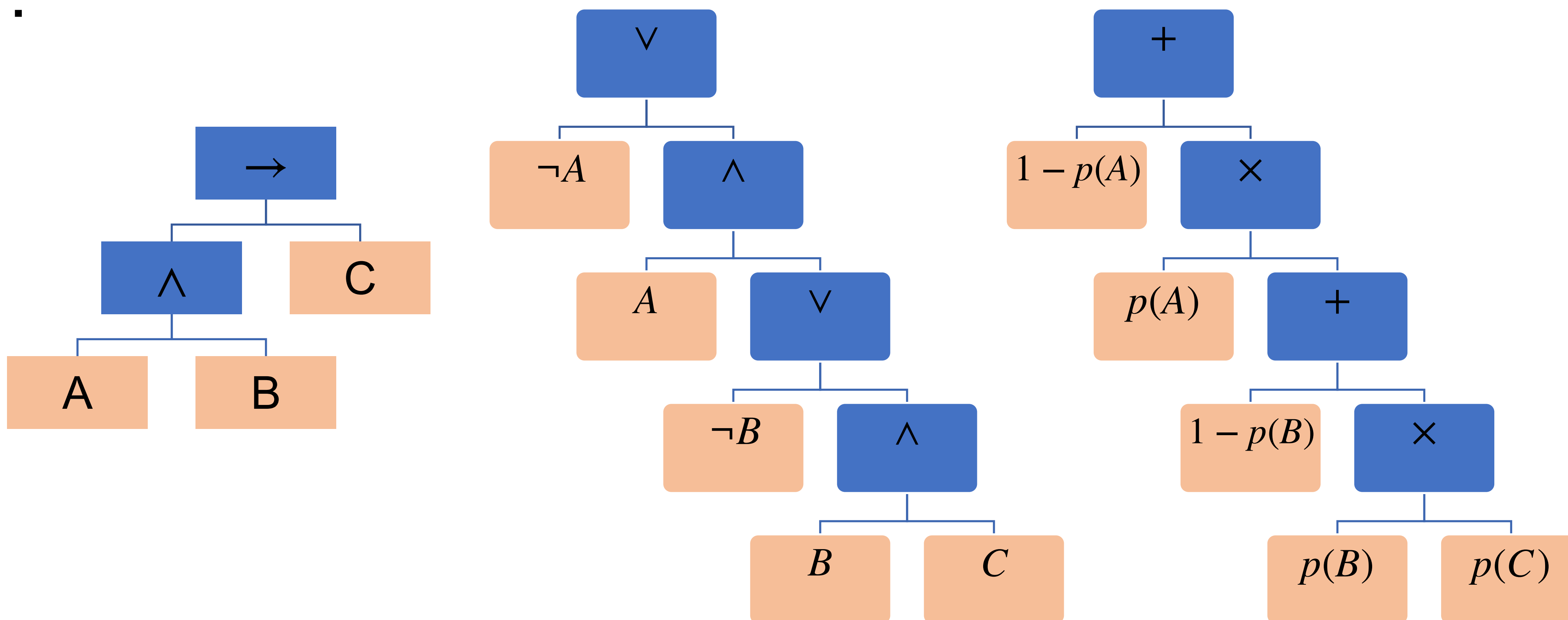


The probabilistic structure is now explicit in the compiled formula.

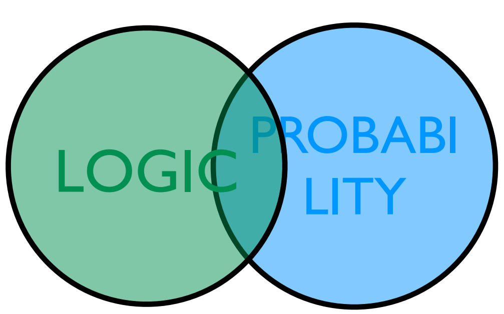
Probabilistic Logic Semantics

- Consider:

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



The circuit is differentiable!



Probabilistic Logic Semantics

- WMC:

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

- Another important inference task in MPE inference (connected to maxSAT)

$$\ell_B^\star(x_1), \ldots, \ell_B^\star(x_n) = \max_{\ell_B(x_1),\ldots,\ell_B(x_n) \models Q} p(\ell_B(x_1), \ldots, \ell_B(x_n))$$

Boolean vs Fuzzy vs Probability

- Boolean and Fuzzy logic are two **alternative** logical semantics

- Probability is a semantics that is built on top of a logical one (i.e. “which is the **probability** of a given **truth assignments** / world?”)

- Can we have a probabilistic fuzzy logic as well?

Probabilistic Soft Logic (PSL)

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

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

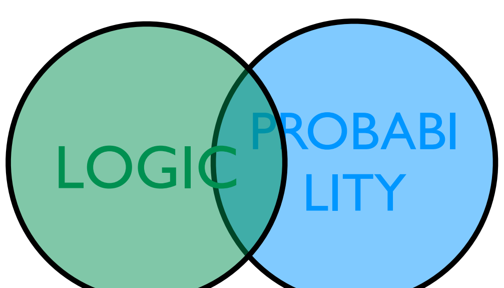
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} w_\alpha \, \ell_B(\alpha) \right)$$

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

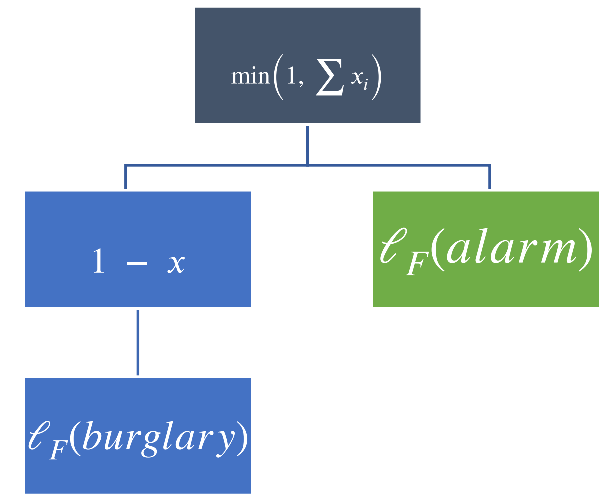
$$p(\ell_F(x_1), \ldots, \ell_F(x_n)) = \frac{1}{Z} \exp\left(\sum_{\alpha} w_\alpha \, \ell_F(\alpha) \right)$$

Weight formula

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



Probabilistic Soft Logic (PSL)



$\alpha : burglary \rightarrow alarm$

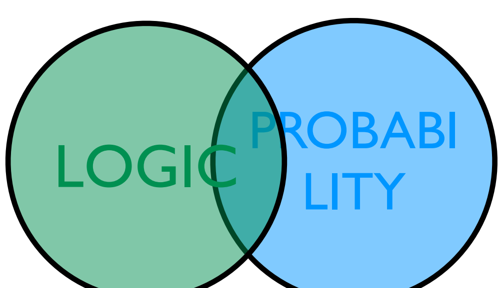
$\ell_F(\alpha) = \min(1, 1 - \ell_F(burglary + \ell_F(alarm))$

MPE:

$$\max_{\ell_F(burglary), \ell_F(alarm)} w_\alpha \ell_F(\alpha)$$

This is soft SAT using fuzzy logic

$$\ell_F(burglary) = \ell_F(burglary) + \lambda \frac{\partial w_\alpha \ell_F(\alpha)}{\partial \ell_F(burglary)}$$



Probabilistic vs Fuzzy

- Fuzzy is an alternative logical semantics and it can still coupled with the probabilistic ones
- Fuzzy logic is **sometimes** used as an approximation of MPE in probabilistic logic
- Fuzzy logic is **sometimes** used to solve **satisfiability** faster
  - **However,** it does not guarantee solutions coherent with the Boolean logic theory.
  - (Remember $A = B = C = D = E = 0.2$)

Logic as constraints

Propositional logic

calls(mary) <- hears_alarm(mary) ∧ alarm

calls(john) <- hears_alarm(john) ∧ alarm

alarm <- earthquake v burglary

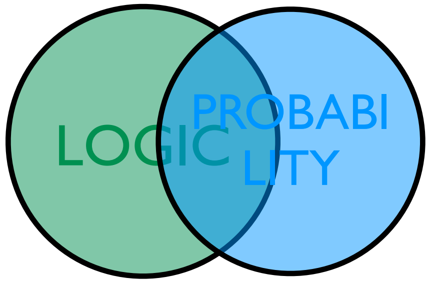
Model / Possible World

0.1	{ burglary,
0.4	hears_alarm(john),
...	alarm,
...	calls(john)}

probability of world ∽ 0.1 x 0.4 x ...

SEMANTIC LOSS =
probability that a random possible world satisfies the formula

using weighted model counting (WMC)
weights/probabilities are on the literals



Logic as soft constraints
Markov Logic

Propositional logic

10 : f1 <-> calls(mary) <- hears_alarm(mary) ∧ alarm

20 : f2 <-> calls(john) <- hears_alarm(john) ∧ alarm

30 : f3 <-> alarm <- earthquake v burglary

Model / Possible World

e^10 { f1,
e^20 f2,
e^30 f3,
burglary, hears_alarm(john),
alarm, calls(john),}

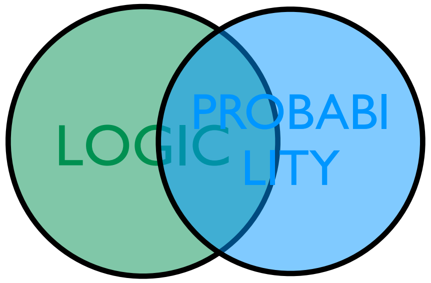
probability of world ~ e^10 x e^20 x e^30

using weighted model counting (WMC)

weights/probabilities are on the formulae (soft constraints)

the higher the weight , the harder or more logical the constraint

w(f1) = e^10 w(not f1) =e^0 = 1
w(f2) = e^20 w(not f2) =e^0 = 1
w(f3) = e^30 w(not f3) =e^0 = 1



(need to normalise to get probability distribution)

Logic as soft constraints

Probabilistic Soft Logic [Bach & Getoor]

Propositional logic

10 : calls(mary) <- hears_alarm(mary) ∧ alarm

20 : calls(john) <- hears_alarm(john) ∧ alarm

30 : alarm <- earthquake v burglary

Model / Possible World

{0.7 burglary,

0.8 hears_alarm(john),

0.5 alarm,

0.3 calls(john),}

atoms are no longer true or false in worlds but true or false to a certain degree

logic : a constraint is satisfied (1) or not (0) by a world
fuzzy logic : the distance to satisfaction
the higher the distance, the less likely the world

Lukasiewicz T-norm

For 0 and 1 we get boolean logic

$A \vee B = min(1, A + B)$

$A \wedge B = min(1, A + B - 1)$

$A \leftarrow B = min(1, 1 + A - B)$ **(residuum)**

evaluates to 1 when rule is satisfied

when $B \leq A$

calls(john) <- hears_alarm(john) ∧ alarm

≥ 0.5 0.7 0.8

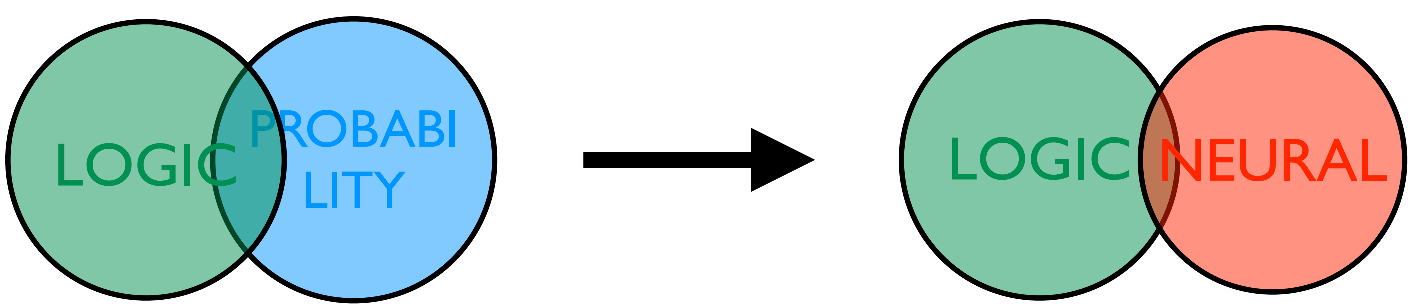
$$A \wedge B = min(1, 1.5 - 1) = 0.5$$

Rule evaluates to $min(1, 1 - 0.5 + 0.3) = 0.8$ when calls(john) =0.3

w= e^ [—20 x (1-0.8)]

6. Semantics

Neural Symbolic



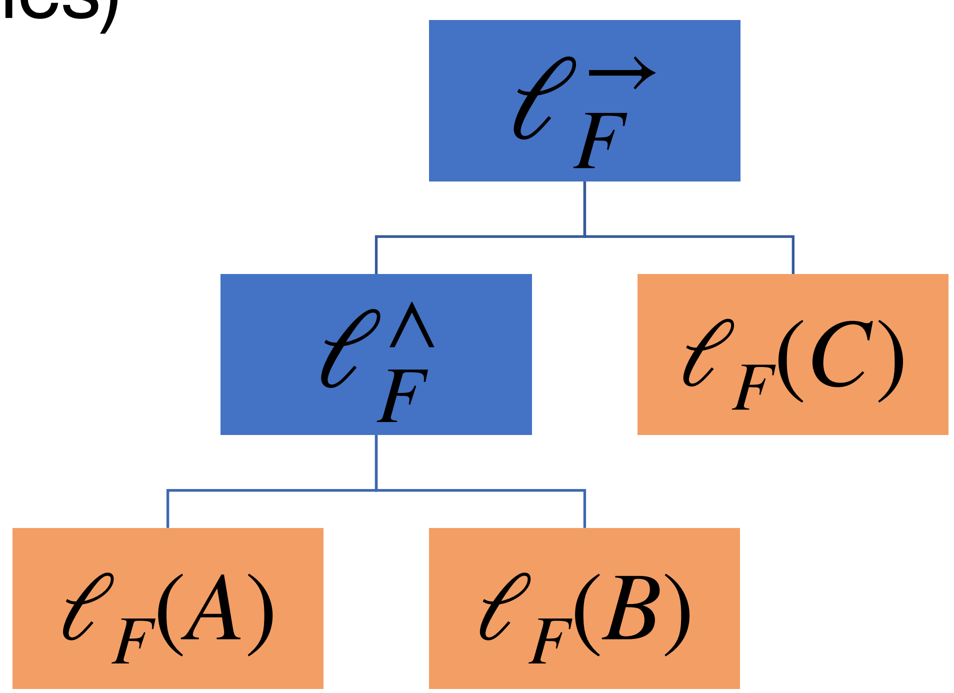
Neural Symbolic

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

$$\ell(Q)$$

Labelling functions (semantics) = Parametric circuit

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



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

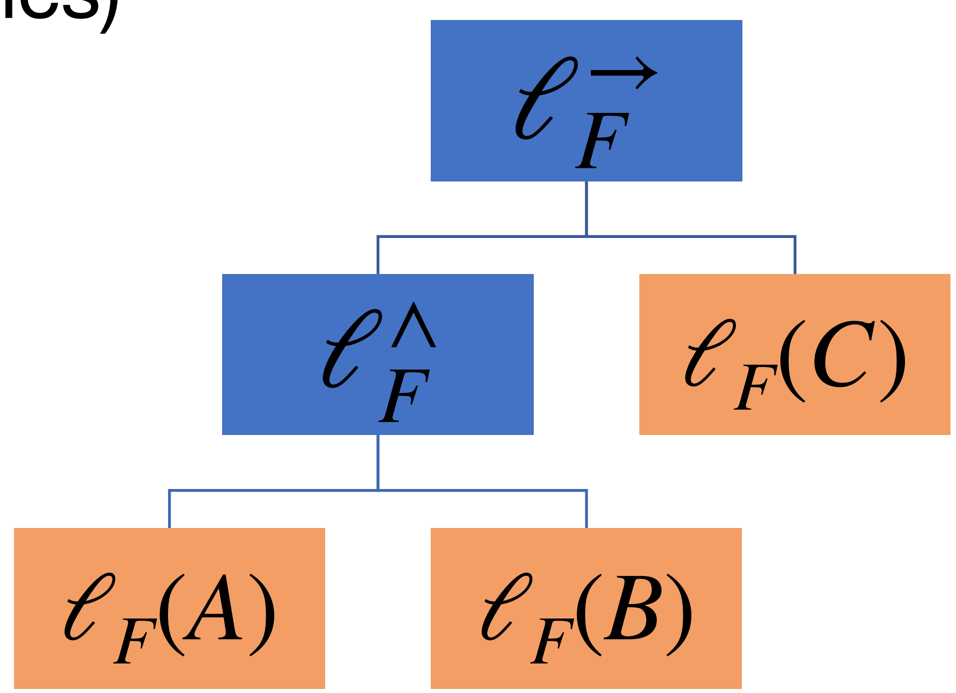
Neural Symbolic

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

$$\ell(Q)$$

Labelling functions (semantics) = Parametric circuit

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



The leaves represent the scalar parameters

Neural Symbolic

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

- Atomic labels are just scalar tables of parameters

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

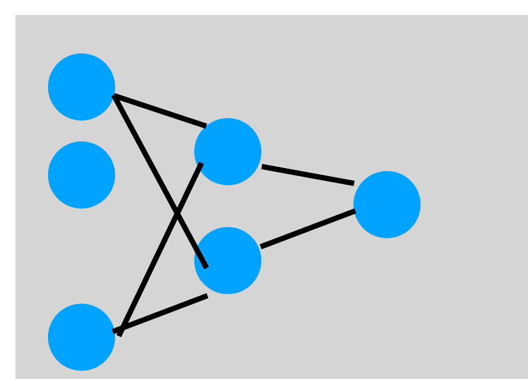
L	p
Burglary	0.1
Earthquake	0.05
…	

Neural Symbolic

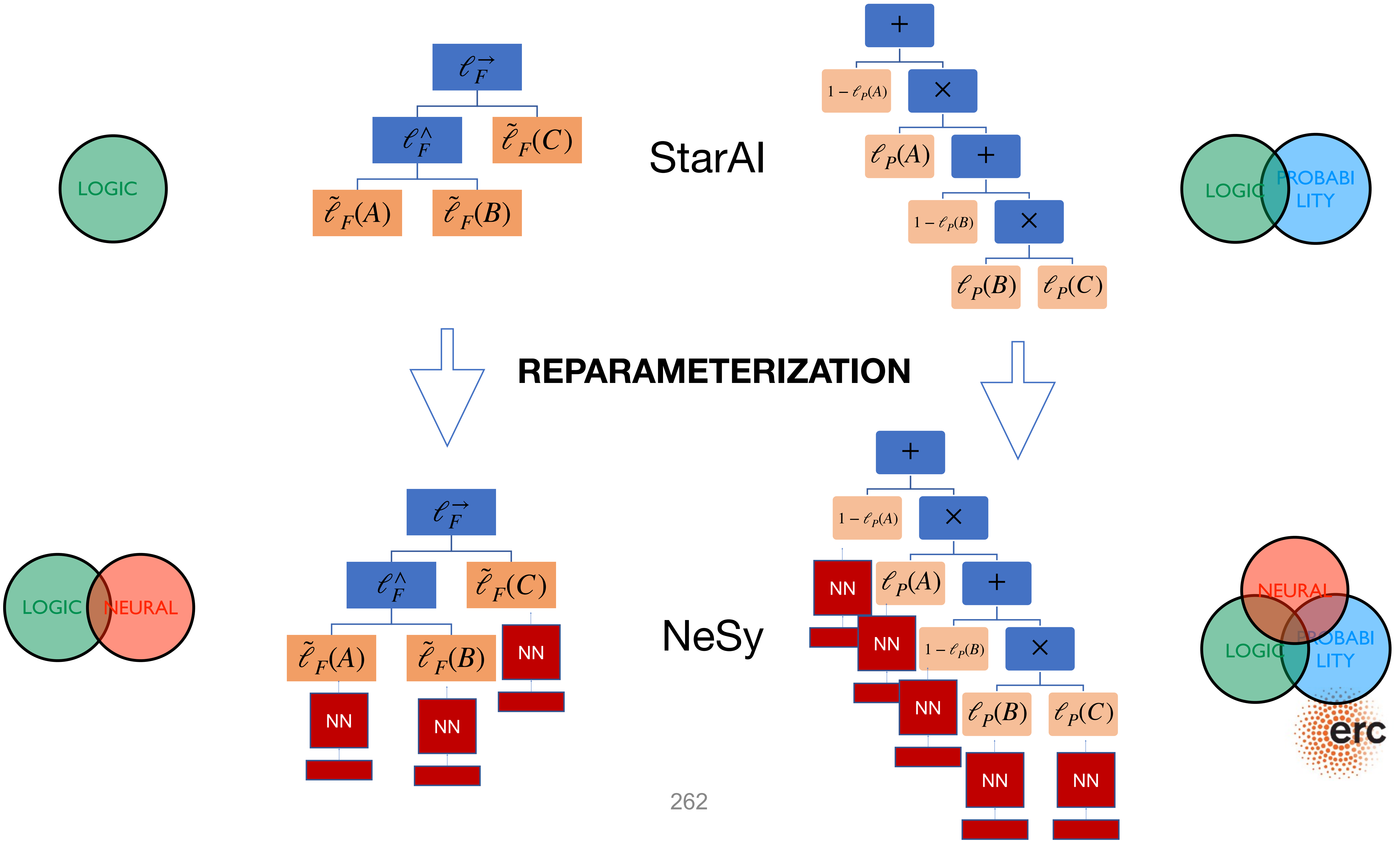
How to carry over concepts from the semantics of StarAI to neural symbolic?

- What if atomic labels are just neural networks?

? :: burglary()
? ::earthquake. ()
? ::hears_alarm(john).
alarm :– earthquake.
alarm :– burglary.

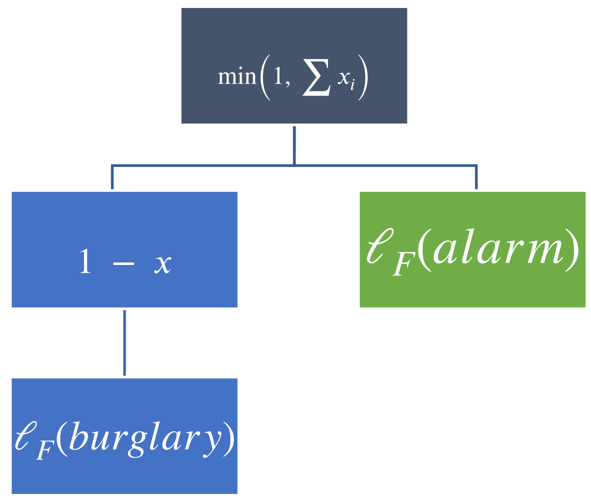


StarAI to Neural Symbolic



Fuzzy Reparameterization

$\alpha : burglary \rightarrow alarm$

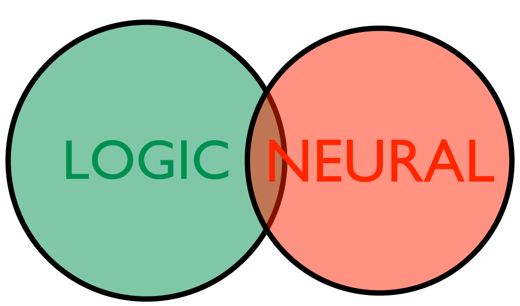


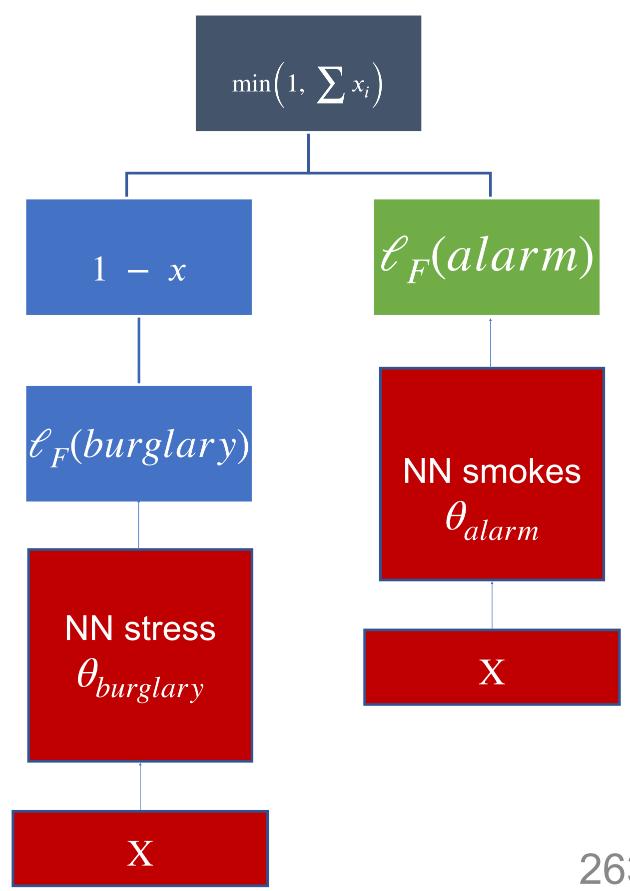
StarAI (PSL)

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

Semantic Based Regularization (Diligenti et al, AI 2017)

Logic Tensor Network (Donadello et at, IJCAI 2017)





NeSy (SBR, LTN)

$$\max_{\theta_{burglary},\theta_{alarm}} w_\alpha \ell_F(\alpha)$$

Parameters of the neural nets

Probabilistic Reparameterization

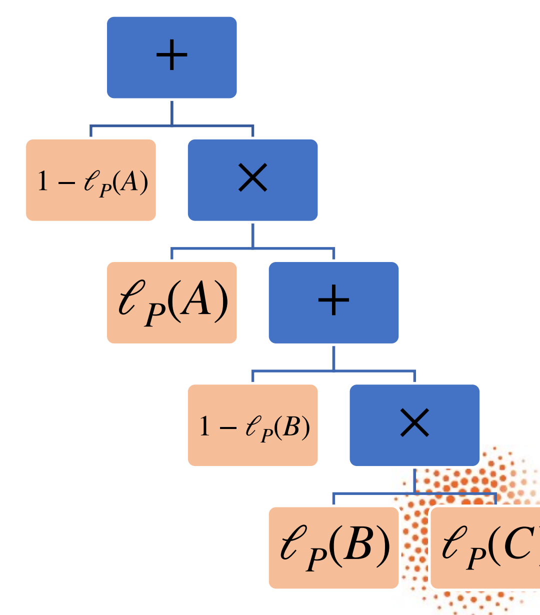
Probabilistic parameters

- ProbLog:

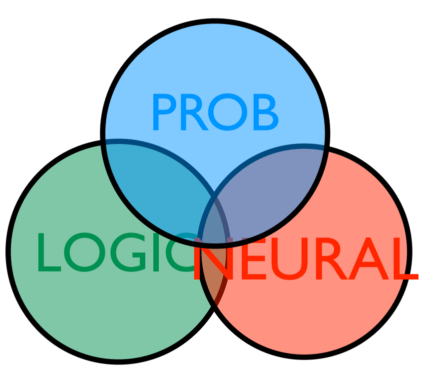
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \prod_{i:\ell_B(x_i)=True} p(x_i) \prod_{i:\ell_B(x_i)=False} (1-p(x_i))$$

- Markov Logic:

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



WMC



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

Probabilistic Reparameterization

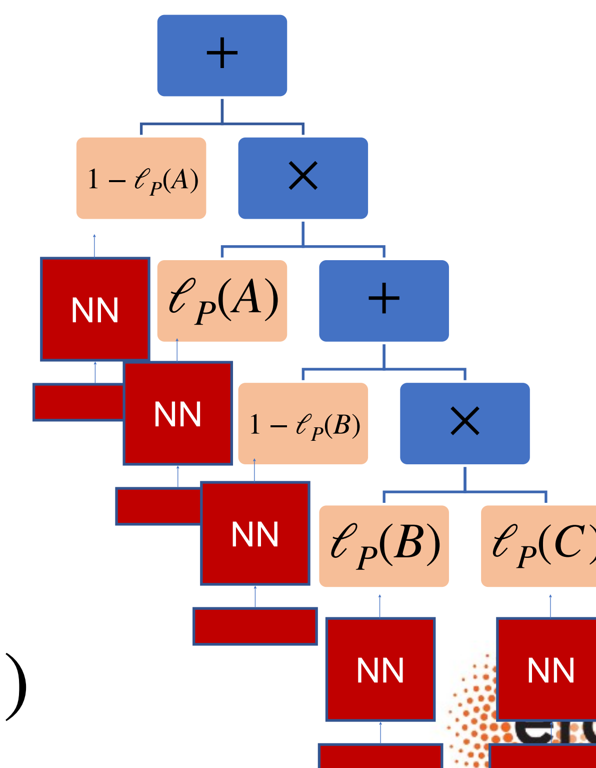
■ Neural parameters

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

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

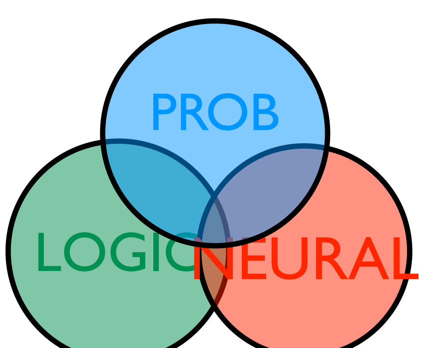
- **Relational Neural Machines** (Marra et al, ECAI 2020)

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



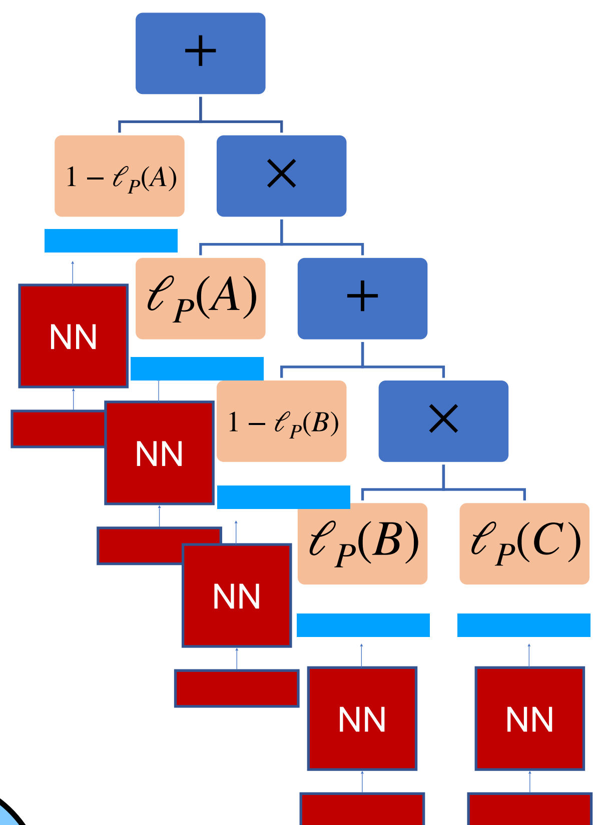
WMC

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



Probabilistic Reparameterization

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



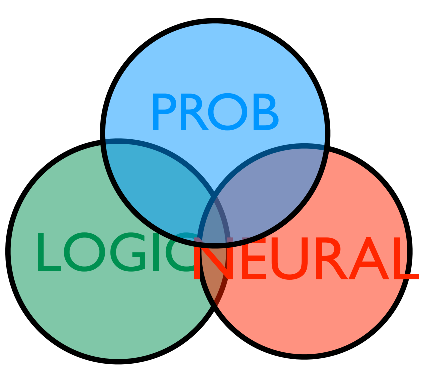
Interface

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

Inference
from SAT to WMC

From SAT to #SAT and WMC

- SAT : does there exist a model for a logical theory ?
- #SAT : how many models are there ? (model counting)
- WMC : what is the *weighted* model count ?

From SAT to #SAT and WMC

- For the previous theory, there were 6 models.
- In the Bayesian network, each possible world had a probability of 1/8 and each literal of 0.5 (weight of 0.5). We can now define the weighted model counting problem (WMC).
  - Given is a logical theory T (usually in CNF),
  - for each literal l, there is a (non-negative) weight $w(l)$.

  The weighted model count of the theory $wmc(T)$ is then :

  - $wmc(T) = \sum_{M \models T} w(M)$
    (where M is model for T, M is the set of all true literals)
  - $w(M) = \prod_{l \in M} w(l)$
- There is a close correspondence between Bayesian network inference and weighted model counting.

WMC Example

$$(\neg A \vee B) \wedge (\neg B \vee C) \wedge (A \vee C).$$

$$w(A) = w(\neg A) = w(B) = w(\neg B) = w(C) = w(\neg C) = 0.5$$

A	B	C	model ?	weight	count $=$ model x weight
0	0	0	0	0.5^3	0
0	0	1	1	0.5^3	0.5^3
0	1	0	1	0.5^3	0.5^3
0	1	1	1	0.5^3	0.5^3
1	0	0	1	0.5^3	0.5^3
1	0	1	1	0.5^3	0.5^3
1	1	0	1	0.5^3	0.5^3
1	1	1	0	0.5^3	0

weight model count wmc: sum of counts 6 x 0.5^3

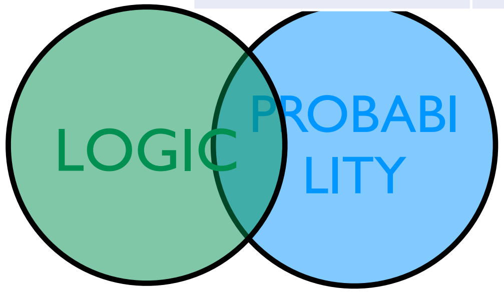
Probabilistic Logic Semantics
Problog

e.g. in ProbLog:

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

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)



Weighted

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

weight of literal

interpretations (truth value assignments) of propositional variables

for p::f,
w(f) = p
w(not f) = 1−p

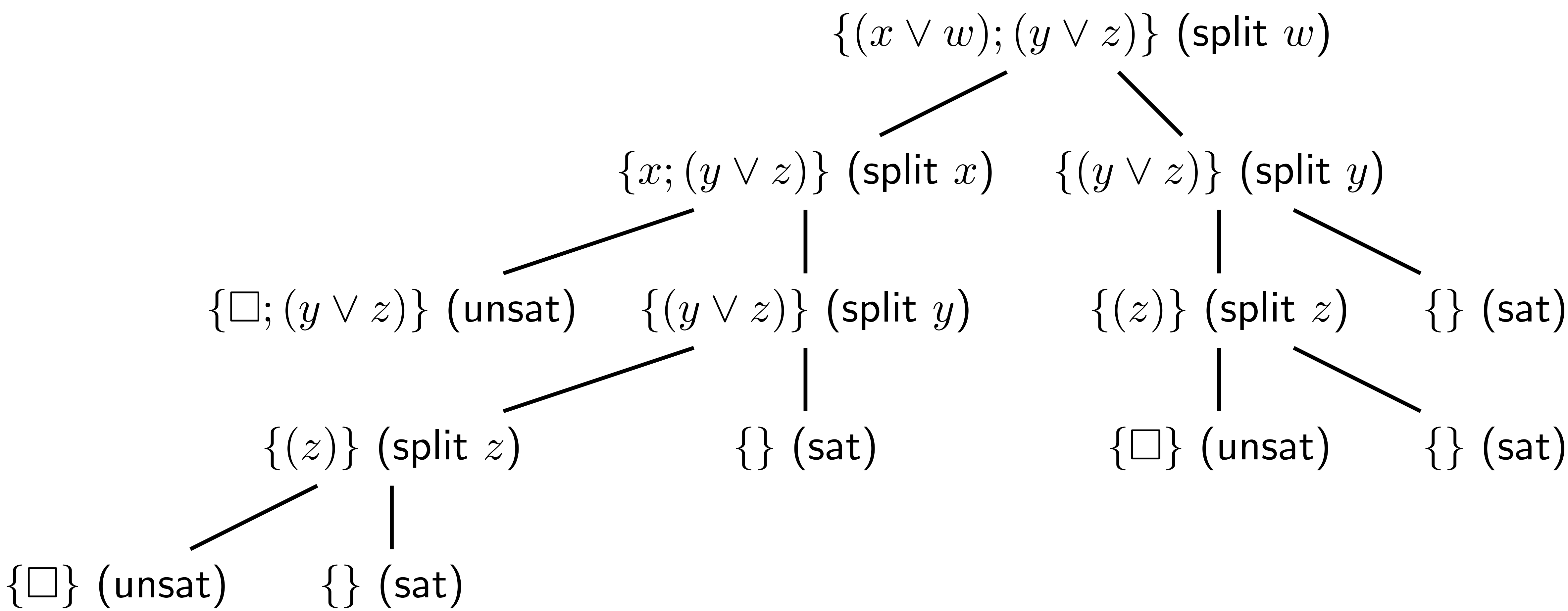
possible worlds

The DPLL Algorithm for SAT

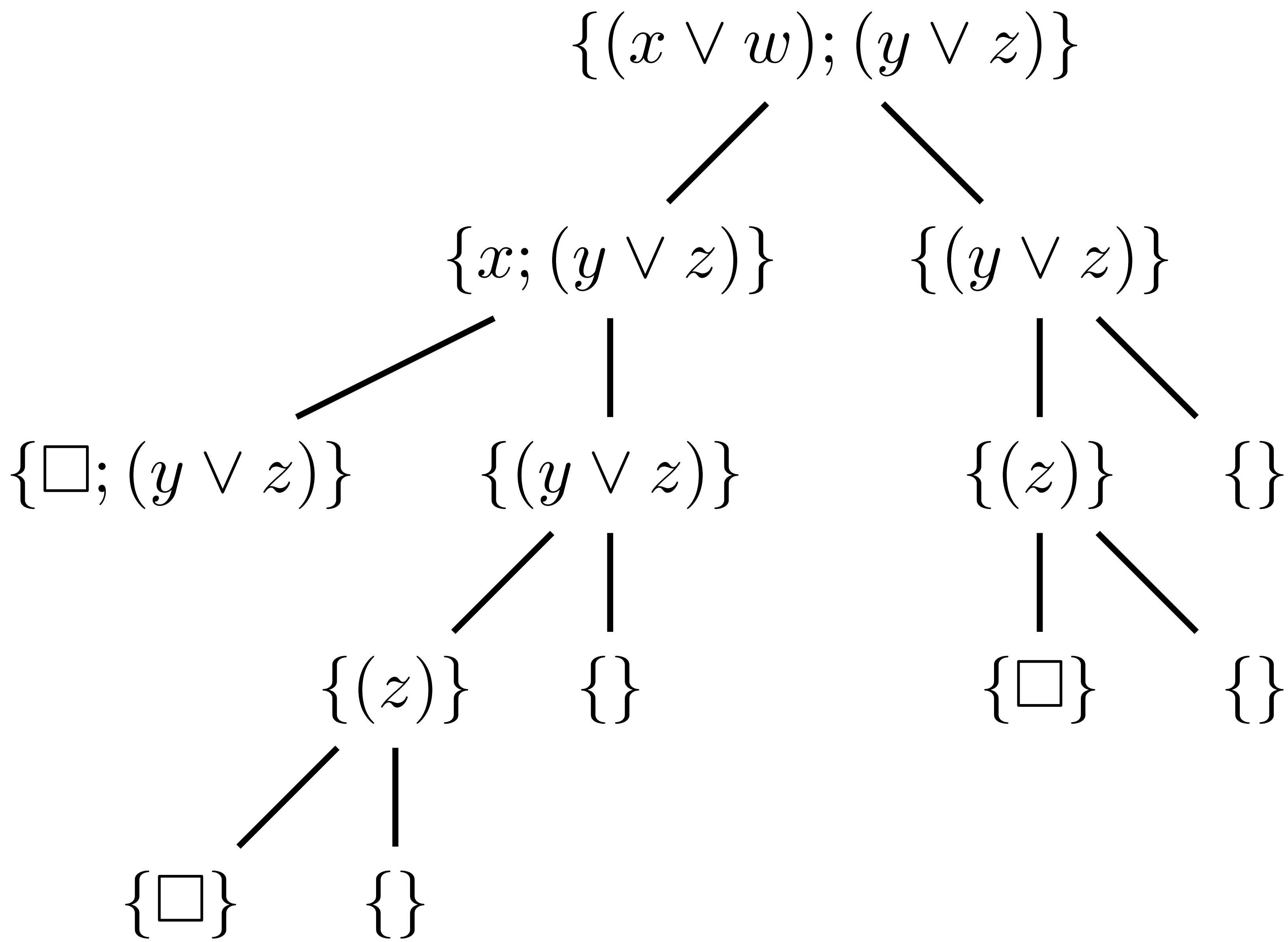
procedure $DPLL(Vars:$ variables$, S:$ set of clauses$)$:
  if S is empty
    return 1
  else if S contains an empty clause
    return 0
  else select $v \in Vars$
    $S_t := S$ where $v = 1$ (making the variable true)
    $S_f := S$ where $v = 0$ (making the variable false)
    return $DPLL(Vars - \{v\}, S_t) + DPLL(Vars - \{v\}, S_f)$

- In a CNF theory $(A \vee \neg B) \wedge (C \vee D)$, the clauses are the disjunctions, that is, $(A \vee \neg B)$ and $(C \vee D)$
- A unit clause contains exactly one literal. E.g., A and $\neg A$ are both unit clauses. (It is possible to make DPLL more efficient by assigning unit clauses the appropriate value)
- An empty clause is a disjunction of 0 literals, at least one of which must be true. Therefore an empty clause is always unsatisfiable.

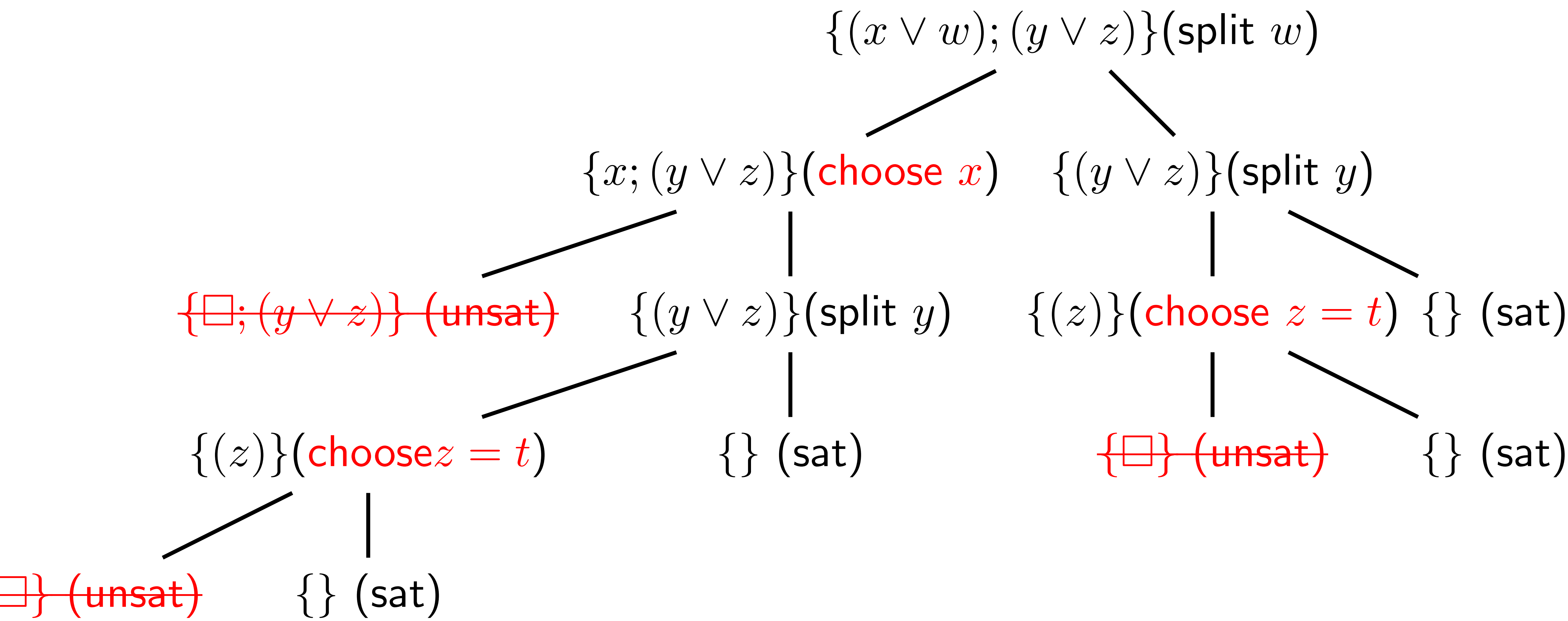
Example for SAT



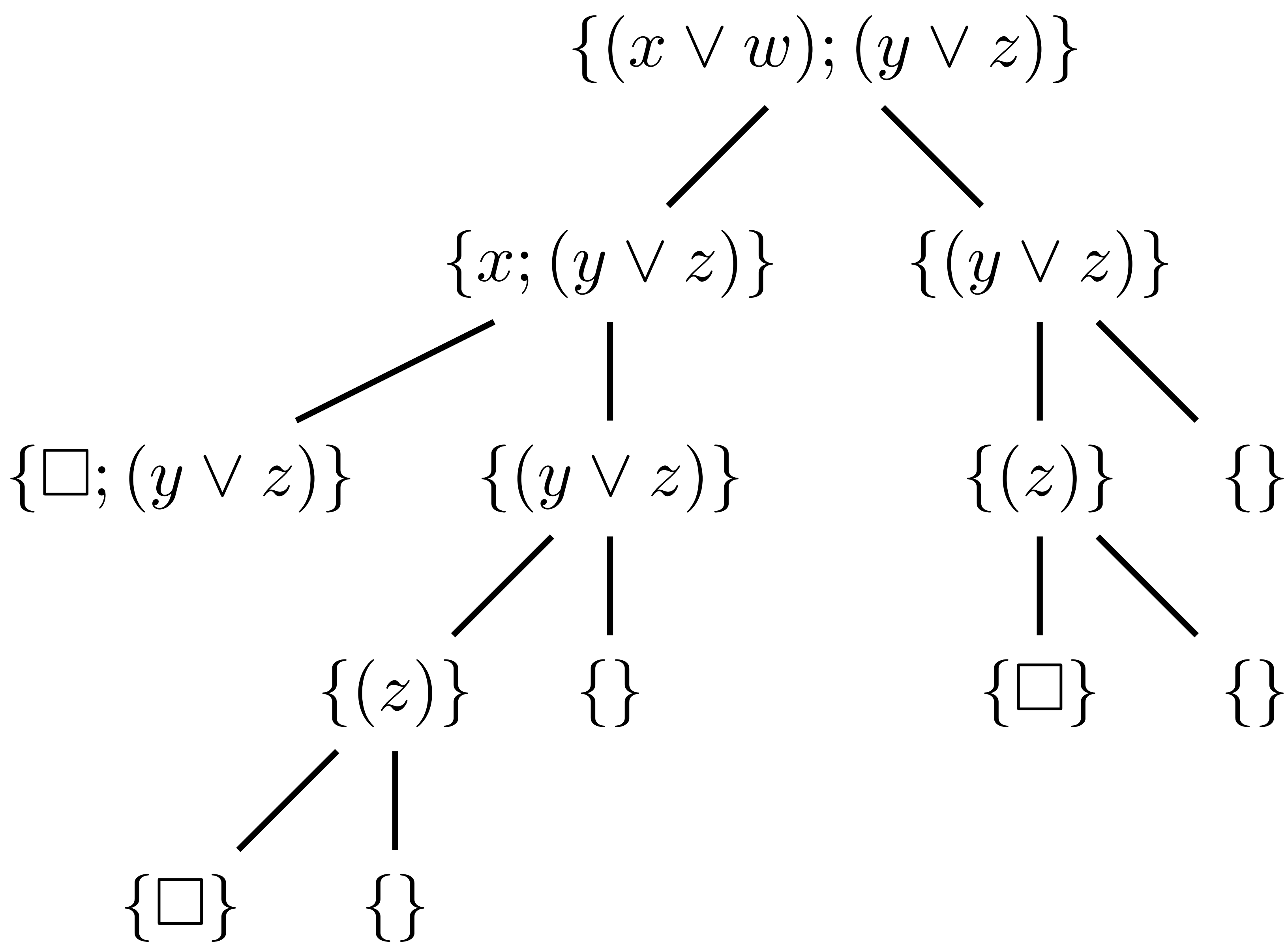
Shorthand Notation



Example - Unit Clause Propagation



Caching

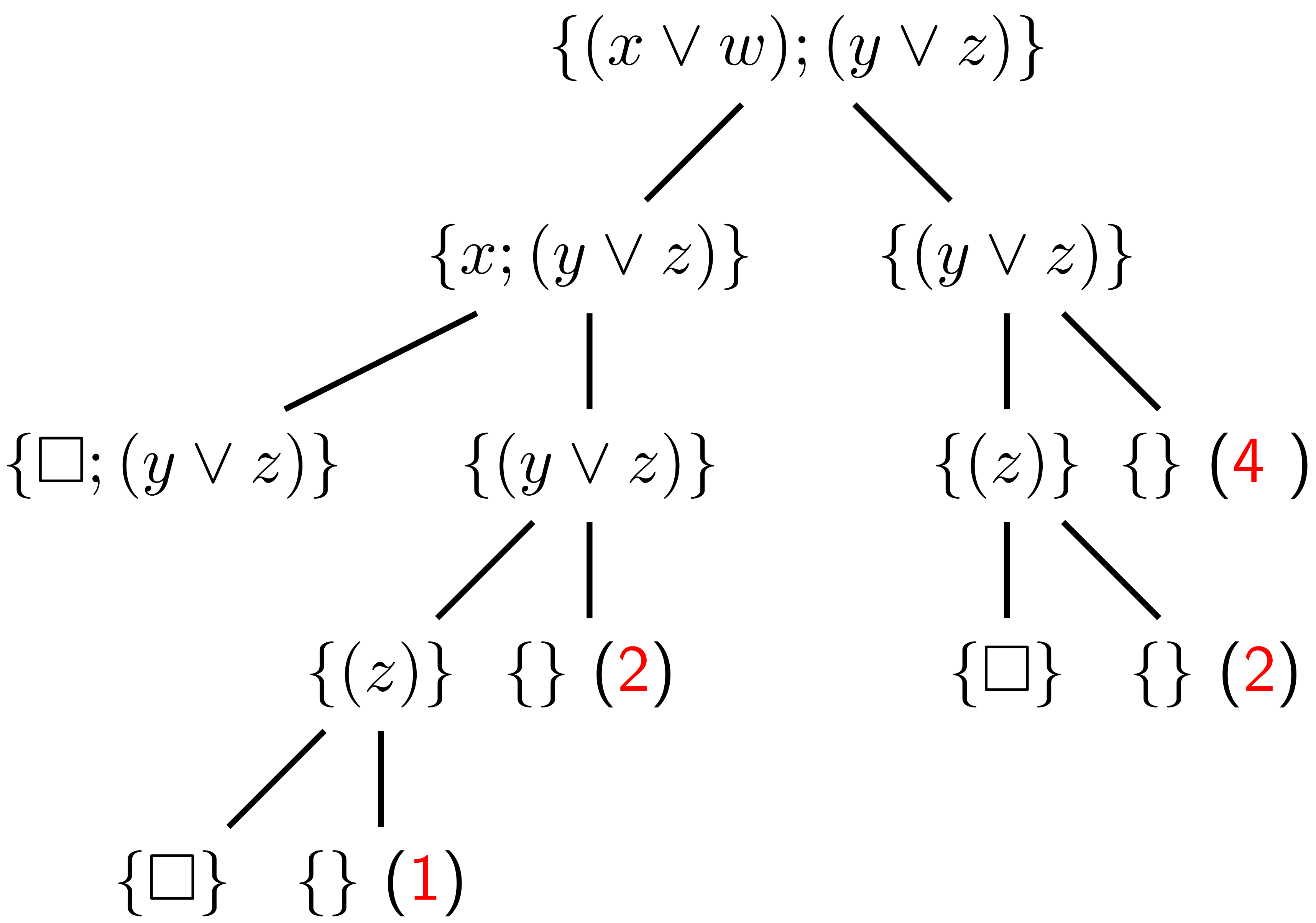


Solution : cache computed answers for sub trees and test whether sub formula was already encountered before.

DPLL Variant for #SAT

procedure $\#SAT(Vars:$ variables, $S:$ set of clauses):
  if S is empty
    return $2^{|Vars|}$
  else if S contains an empty clause
    return 0
  else select $v \in Vars$
    $S_t := S$ where $v = 1$ (making the variable true)
    $S_f := S$ where $v = 0$ (making the variable false)
    return $\#SAT(Vars - \{v\}, S_t) + \#SAT(Vars - \{v\}, S_f)$

Example #SAT



DPLL Variant for WMC

procedure $WMC(Vars:$ variables, $S:$ set of clauses):

  if S is empty

    return $\prod_{v \in Vars} w(v) + w(\neg v)$

  else if S contains an empty clause

    return 0

  else select $v \in Vars$

    $S_t := S$ where $v = 1$ (making the variable true)

    $S_f := S$ where $v = 0$ (making the variable false)

    return $w(v)\ WMC(Vars - \{v\}, S_t) + w(\neg v)\ WMC(Vars - \{v\}, S_f)$

What have we done ?

We have used sum products — semi-rings

A semiring is a structure $(A, \oplus, \otimes, e^{\oplus}, e^{\otimes})$, where addition $\oplus$ and multiplication $\otimes$ are associative binary operations over the set $A, \oplus$ is commutative, $\otimes$ distributes over $\oplus$, $e^{\oplus}$. $e^{\oplus} \in A$ is the neutral element of $\oplus$, $e^{\otimes} \in A$ that of $\otimes$,and for all $a \in A$, $e^{\oplus} \otimes a = a \otimes e^{\oplus} = e^{\oplus}$. In a commutative semiring, $\otimes$ is commutative as well.

Algebraic Model Counting

- commutative semiring $(A,\oplus,\otimes,e^{\oplus},e^{\otimes})$
- algebraic literals
  $L(F) = \{f_1,\ldots,f_n\} \cup \{\neg f_1,\ldots,\neg f_n\}$
- labeling function $\alpha : L(F) \rightarrow A$
- propositional logical theory T

$$AMC(T) = \bigoplus_{T \models w} \bigotimes_{l \in w} \alpha(l)$$

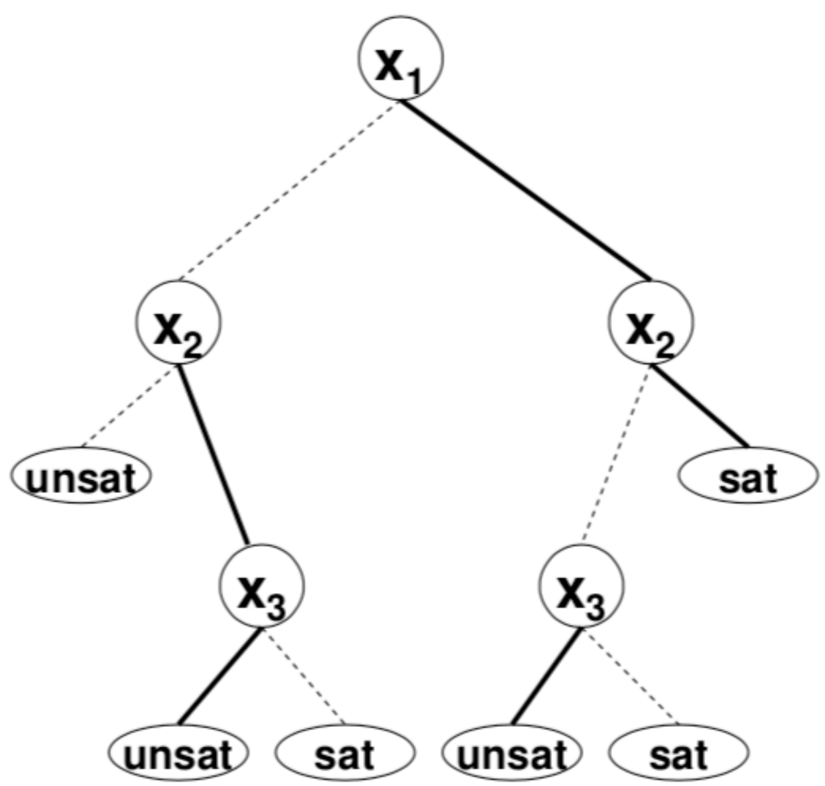
Useful Semirings

task	$\mathcal{A}$	$e^{\oplus}$	$e^{\otimes}$	$\oplus$	$\otimes$	$\alpha(v)$	$\alpha(\neg v)$	ref
SAT	$\{true, false\}$	$false$	$true$	$\vee$	$\wedge$	$true$	$true$	B, BT, G, GK, K, L, M
#SAT	$\mathbb{N}$	0	1	$+$	$\cdot$	1	1	B, G, GK, K, L
WMC	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in \mathbb{R}_{\geq 0}$	$\in \mathbb{R}_{\geq 0}$	
PROB	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in [0,1]$	$1 - \alpha(v)$	B, BT, E, G, K
SENS	$\mathbb{R}[\mathcal{V}]$	0	1	$+$	$\cdot$	v or $\in [0,1]$	$1 - \alpha(v)$	K
GRAD	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	$(0,0)$	$(1,0)$	Eq. (4)	Eq. (5)	Eq. (2)	Eq. (3)	E, K
MPE	$\mathbb{R}_{\geq 0}$	0	1	max	$\cdot$	$\in [0,1]$	$1 - \alpha(v)$	B, BT, G, K, L, M
S-PATH	$\mathbb{N}^{\infty}$	∞	0	min	$+$	$\in \mathbb{N}$	0	BT, GK, K
W-PATH	$\mathbb{N}^{\infty}$	0	∞	max	min	$\in \mathbb{N}$	∞	BT
FUZZY	$[0,1]$	0	1	max	min	$\in [0,1]$	1	GK, M
kWEIGHT	$\{0,\ldots,k\}$	k	0	min	$+^{k}$	$\in \{0,\ldots,k\}$	$\in \{0,\ldots,k\}$	M
$\texttt{OBDD}_{<}$	$\texttt{OBDD}_{<}(\mathcal{V})$	$\texttt{OBDD}_{<}(0)$	$\texttt{OBDD}_{<}(1)$	$\vee$	$\wedge$	$\texttt{OBDD}_{<}(v)$	$\neg\texttt{OBDD}_{<}(v)$	K
WHY	$\mathcal{P}(\mathcal{V})$	$\emptyset$	$\emptyset$	$\cup$	$\cup$	$\{v\}$	n/a	GK
$\mathcal{RA}^{+}$	$\mathbb{N}[\mathcal{V}]$	0	1	$+$	$\cdot$	v	n/a	GK

Table 1: Examples of commutative semirings and labeling functions. The **WHY** and $\mathcal{RA}^{+}$ provenance semirings apply to positive literals only. Reference key: B (Bacchus et al., 2009), BT (Baras and Theodorakopoulos, 2010), E (Eisner, 2002), G (Goodman, 1999), GK (Green et al., 2007), K (Kimmig et al., 2011), L (Larrosa et al., 2010), M (Meseguer et al., 2006); more examples can be found in these references.

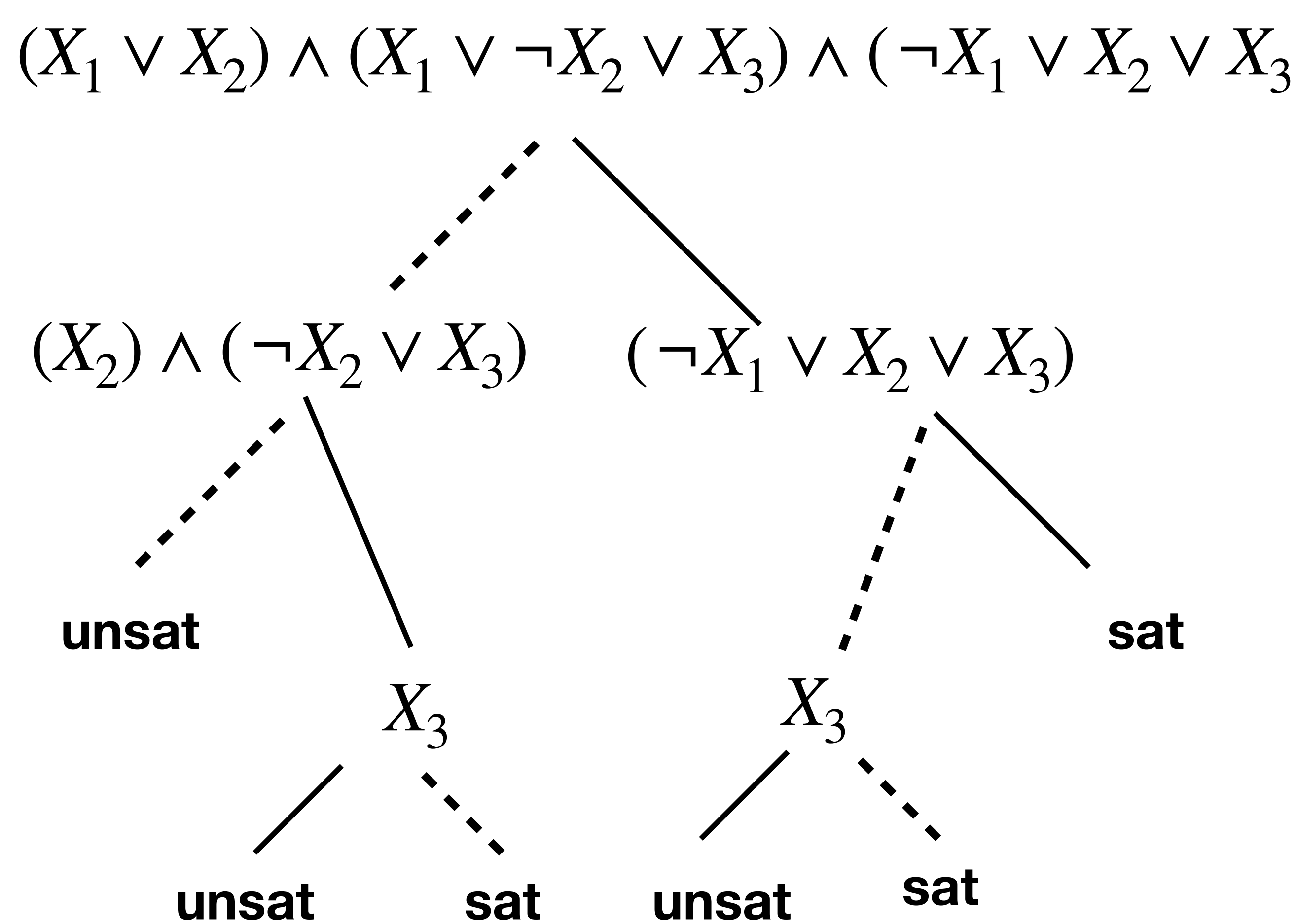
From Kimmig, Vanden Broeck and De Raedt, 2016

NNFs and Decision Nodes



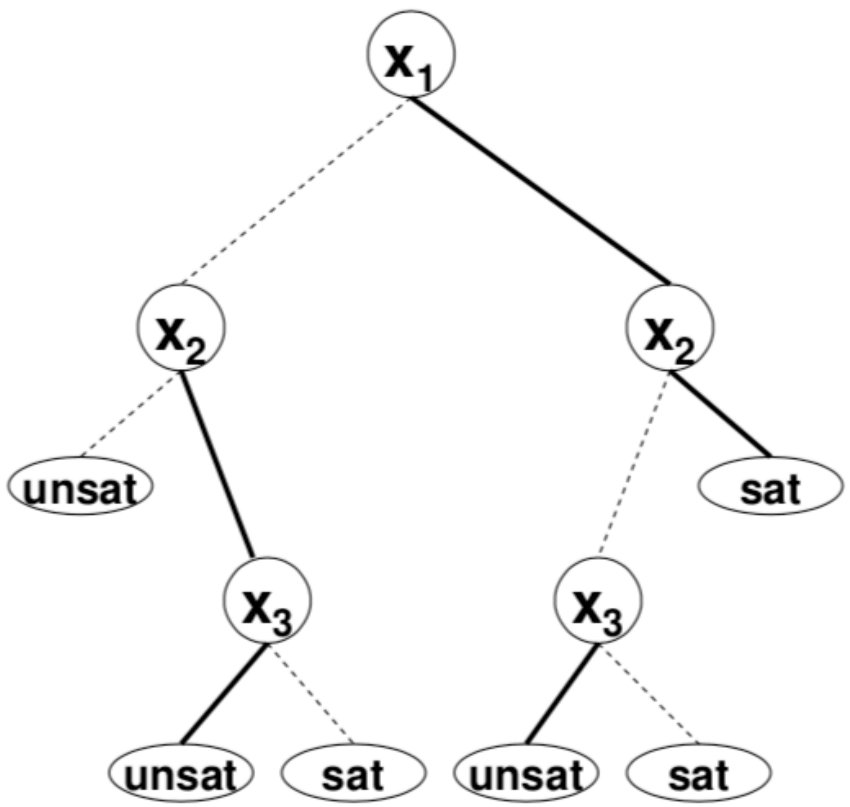
(a) Termination tree

DPLL

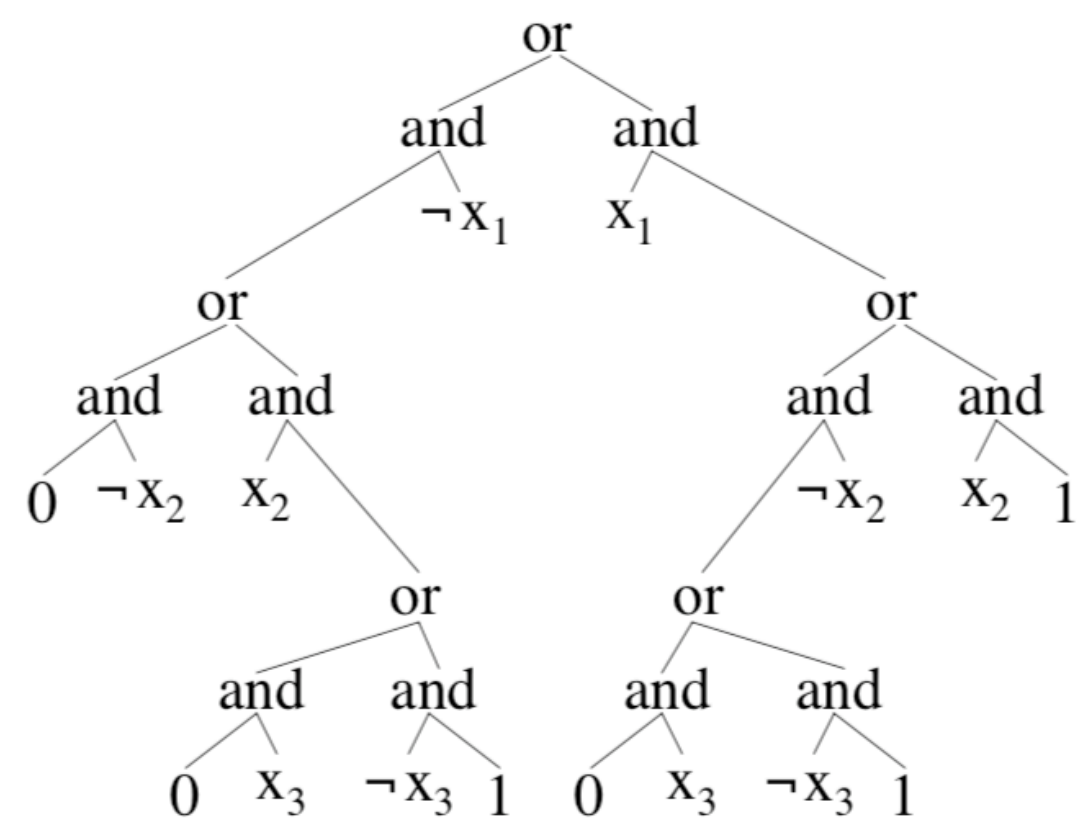


from [Huang & Darwiche, JAIR 2007]

NNFs and Decision Nodes



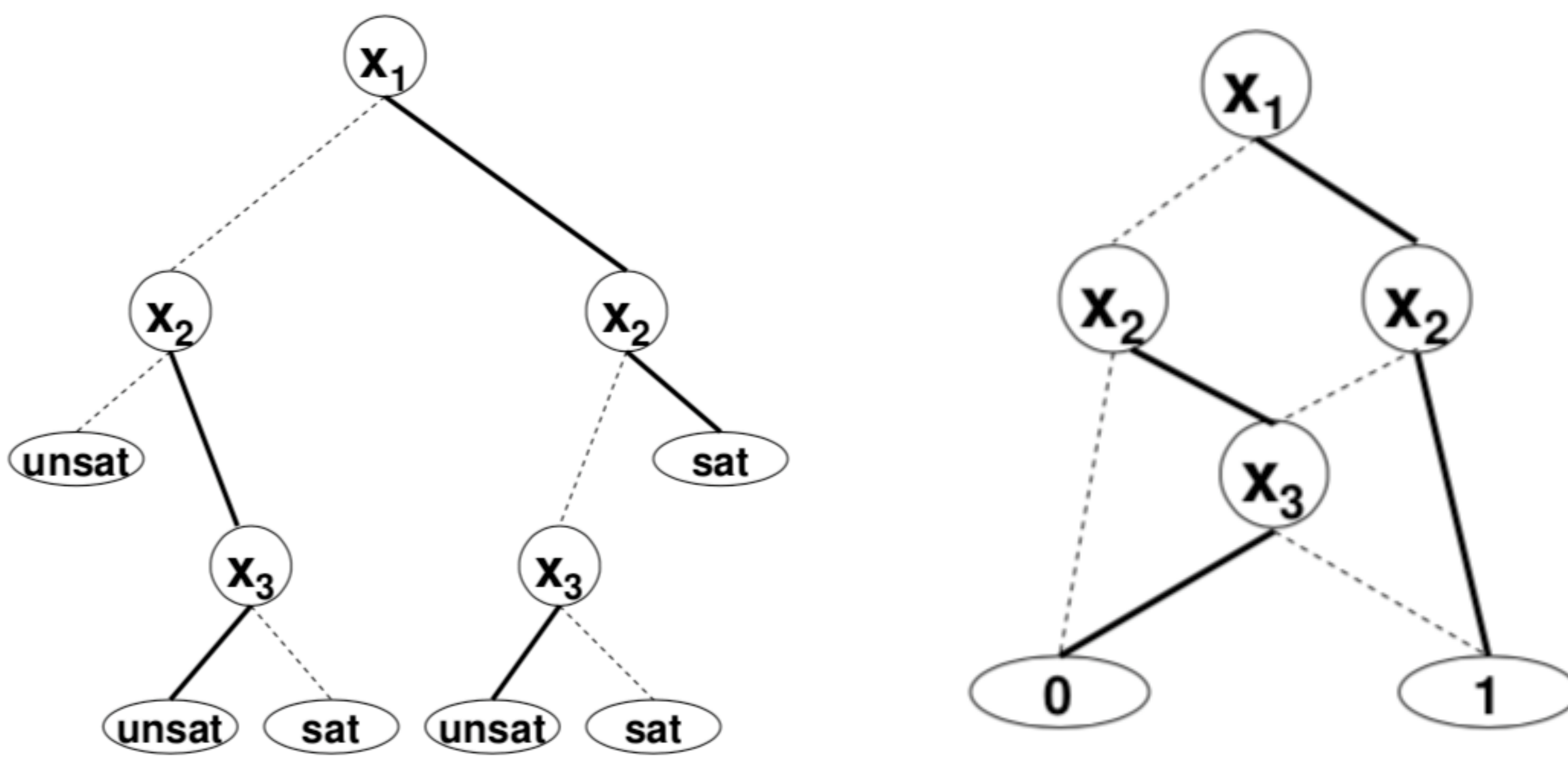
(a) Termination tree



(b) Equivalent NNF circuit

from [Huang & Darwiche, JAIR 2007]

NNFs and Decision Nodes



(a) Termination tree (c) OBDD

DPLL + Caching

Ordered = same order vars along all paths (not really used here)

Identifying and Exploiting Isomorphisms

from [Huang & Darwiche, JAIR 2007]

NNFs : special forms

- Why is this important ?
  - remember that we replace and by x and or by +
- decomposability allows to rewrite $P(A \wedge B) = P(A) \times P(B)$
- determinism allows to rewrite $P(A \vee B) = P(A) + P(B)$
- without smoothness you might not take into account all variables
- these eqs. do not hold for arbitrary formula A and B !

Useful Semirings

task	$\mathcal{A}$	$e^{\oplus}$	$e^{\otimes}$	$\oplus$	$\otimes$	$\alpha(v)$	$\alpha(\neg v)$	ref
SAT	$\{true, false\}$	$false$	$true$	$\vee$	$\wedge$	$true$	$true$	B, BT, G, GK, K, L, M
#SAT	$\mathbb{N}$	0	1	$+$	$\cdot$	1	1	B, G, GK, K, L
WMC	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in \mathbb{R}_{\geq 0}$	$\in \mathbb{R}_{\geq 0}$	
PROB	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in [0,1]$	$1 - \alpha(v)$	B, BT, E, G, K
SENS	$\mathbb{R}[\mathcal{V}]$	0	1	$+$	$\cdot$	v or $\in [0,1]$	$1 - \alpha(v)$	K
GRAD	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	$(0,0)$	$(1,0)$	Eq. (4)	Eq. (5)	Eq. (2)	Eq. (3)	E, K
MPE	$\mathbb{R}_{\geq 0}$	0	1	max	$\cdot$	$\in [0,1]$	$1 - \alpha(v)$	B, BT, G, K, L, M
S-PATH	$\mathbb{N}^{\infty}$	∞	0	min	$+$	$\in \mathbb{N}$	0	BT, GK, K
W-PATH	$\mathbb{N}^{\infty}$	0	∞	max	min	$\in \mathbb{N}$	∞	BT
FUZZY	$[0,1]$	0	1	max	min	$\in [0,1]$	1	GK, M
kWEIGHT	$\{0,\ldots,k\}$	k	0	min	$+^{k}$	$\in \{0,\ldots,k\}$	$\in \{0,\ldots,k\}$	M
$\mathtt{OBDD}_<$	$\mathtt{OBDD}_<(\mathcal{V})$	$\mathtt{OBDD}_<(0)$	$\mathtt{OBDD}_<(1)$	$\vee$	$\wedge$	$\mathtt{OBDD}_<(v)$	$\neg\mathtt{OBDD}_<(v)$	K
WHY	$\mathcal{P}(\mathcal{V})$	$\emptyset$	$\emptyset$	$\cup$	$\cup$	$\{v\}$	n/a	GK
$\mathcal{RA}^+$	$\mathbb{N}[\mathcal{V}]$	0	1	$+$	$\cdot$	v	n/a	GK

Table 1: Examples of commutative semirings and labeling functions. The **WHY** and $\mathcal{RA}^+$ provenance semirings apply to positive literals only. Reference key: B (Bacchus et al., 2009), BT (Baras and Theodorakopoulos, 2010), E (Eisner, 2002), G (Goodman, 1999), GK (Green et al., 2007), K (Kimmig et al., 2011), L (Larrosa et al., 2010), M (Meseguer et al., 2006); more examples can be found in these references.

From Kimmig, Vanden Broeck and De Raedt, 2016

7. Logic vs Probability vs Neural

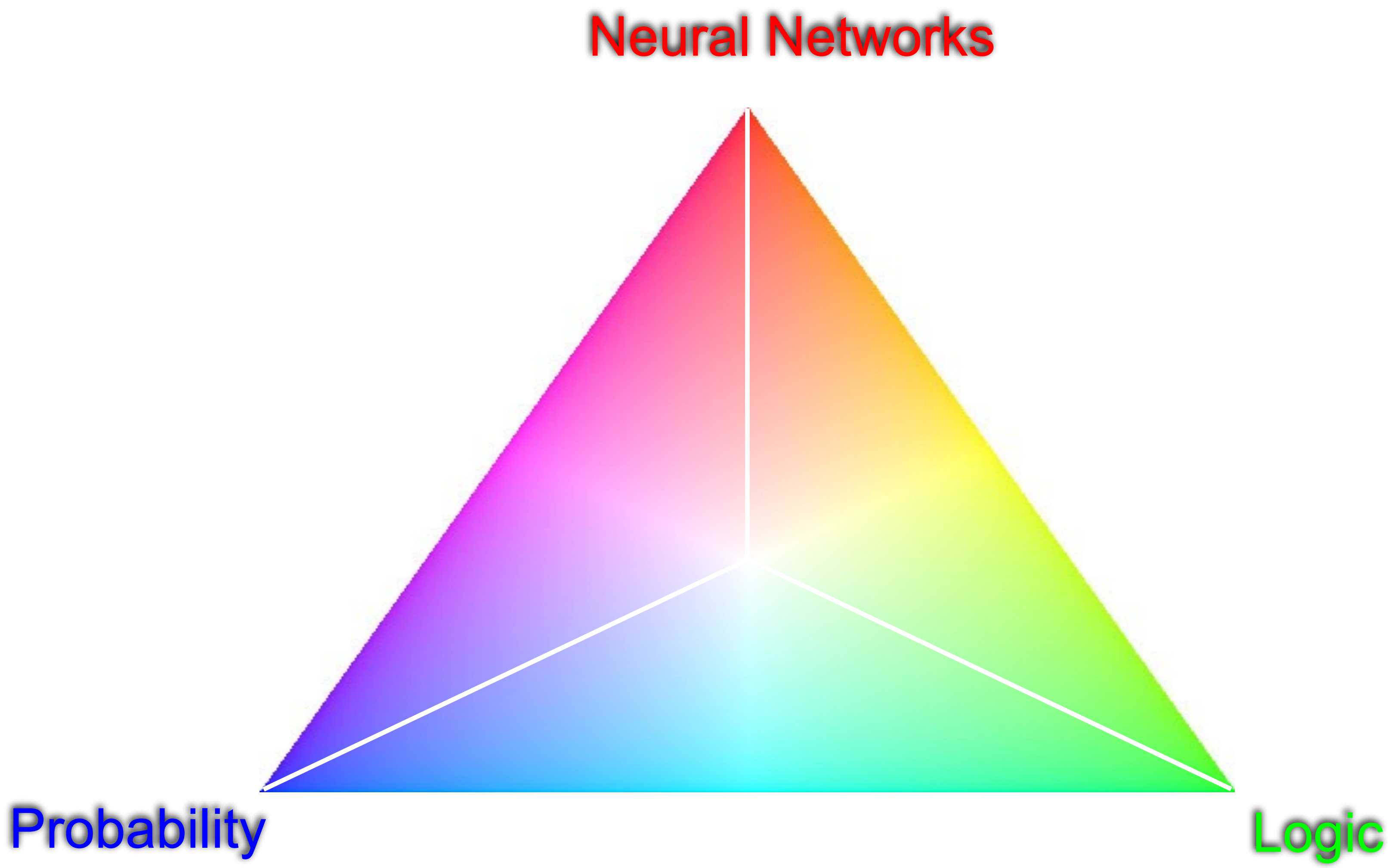
7. Logic vs Probability vs Neural Key Messages

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

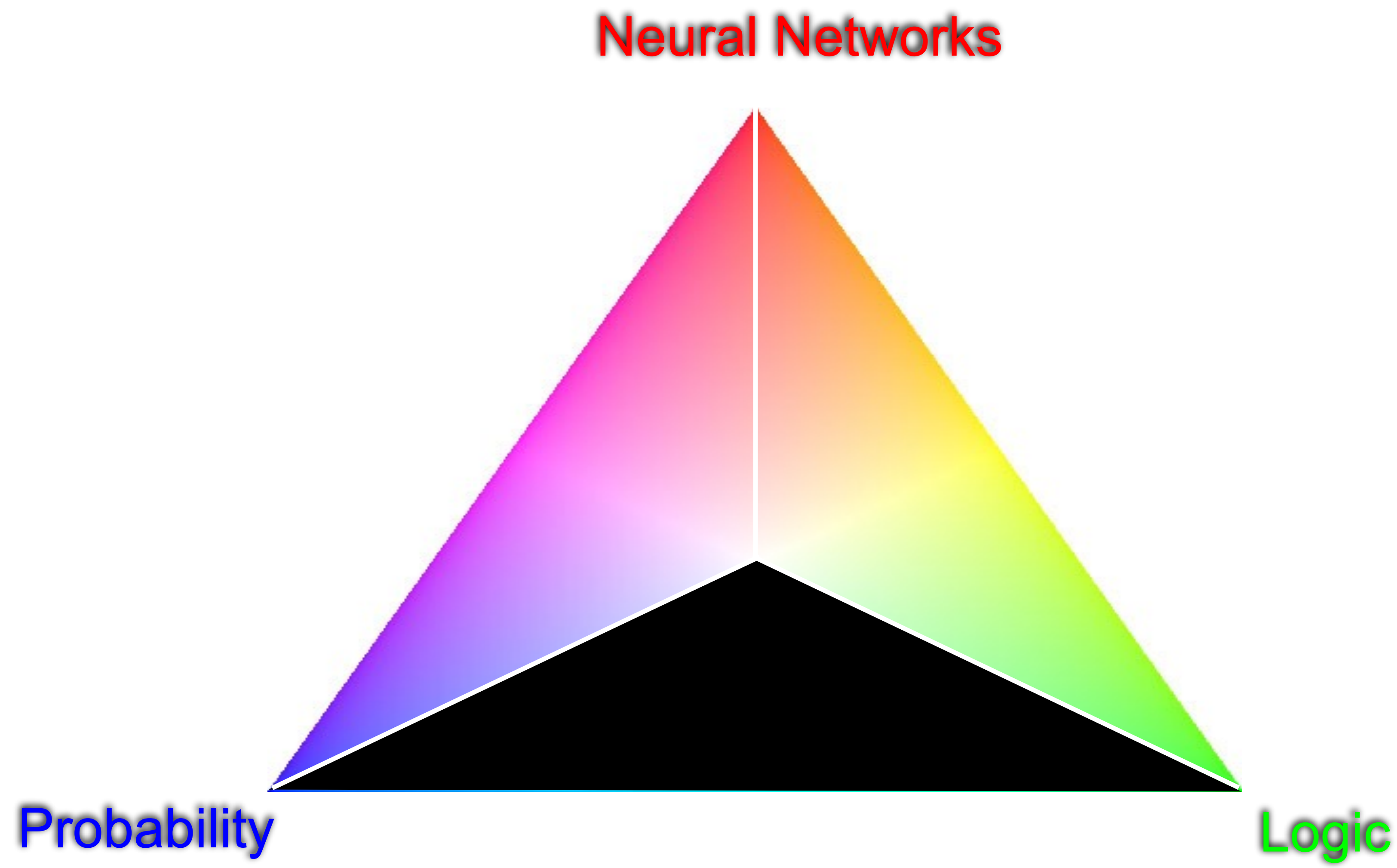
- An integration of the three should have the original paradigms as special cases
  - Computationally complex

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

About integration in neural symbolic

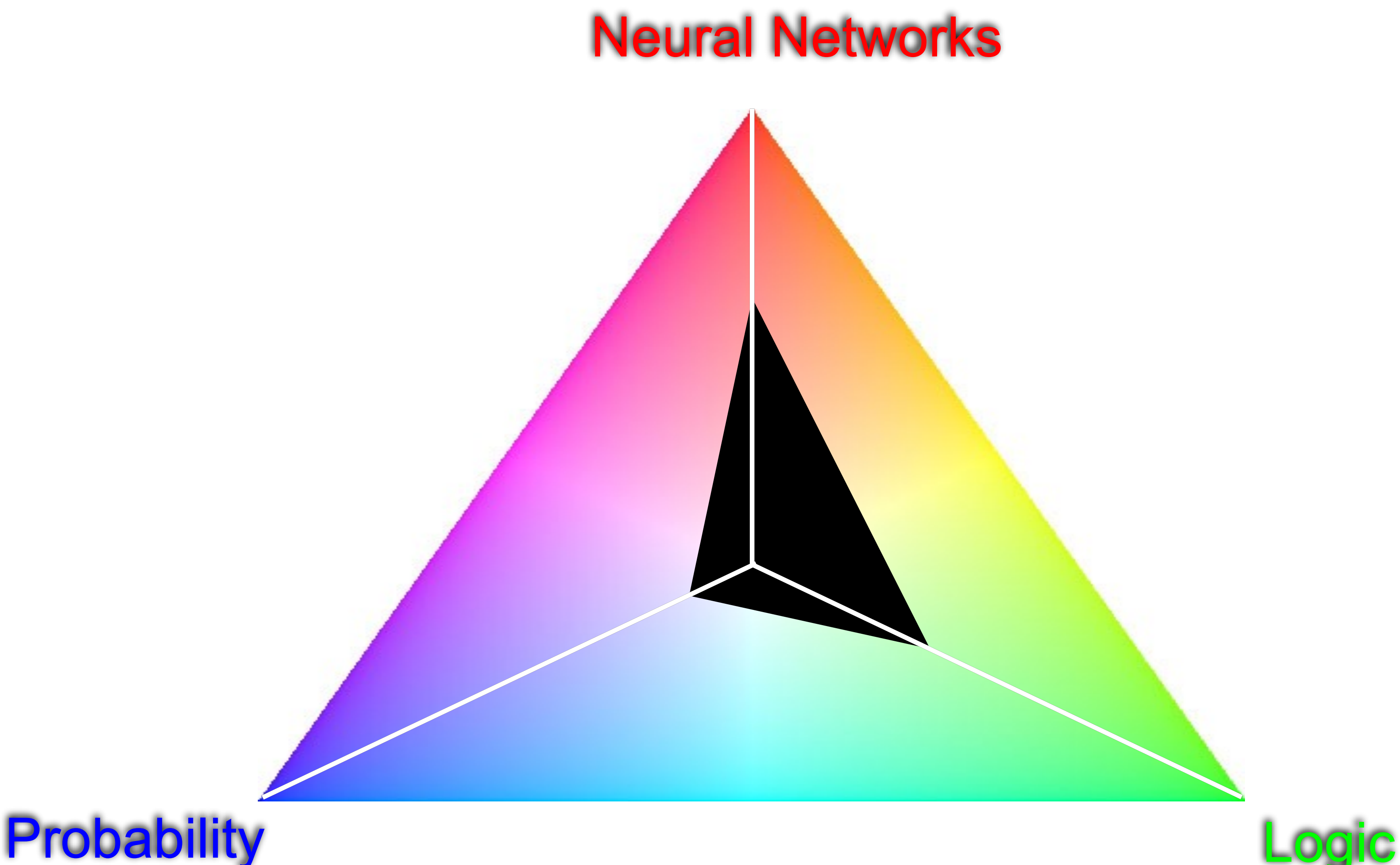


Statistical Relational AI



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

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.

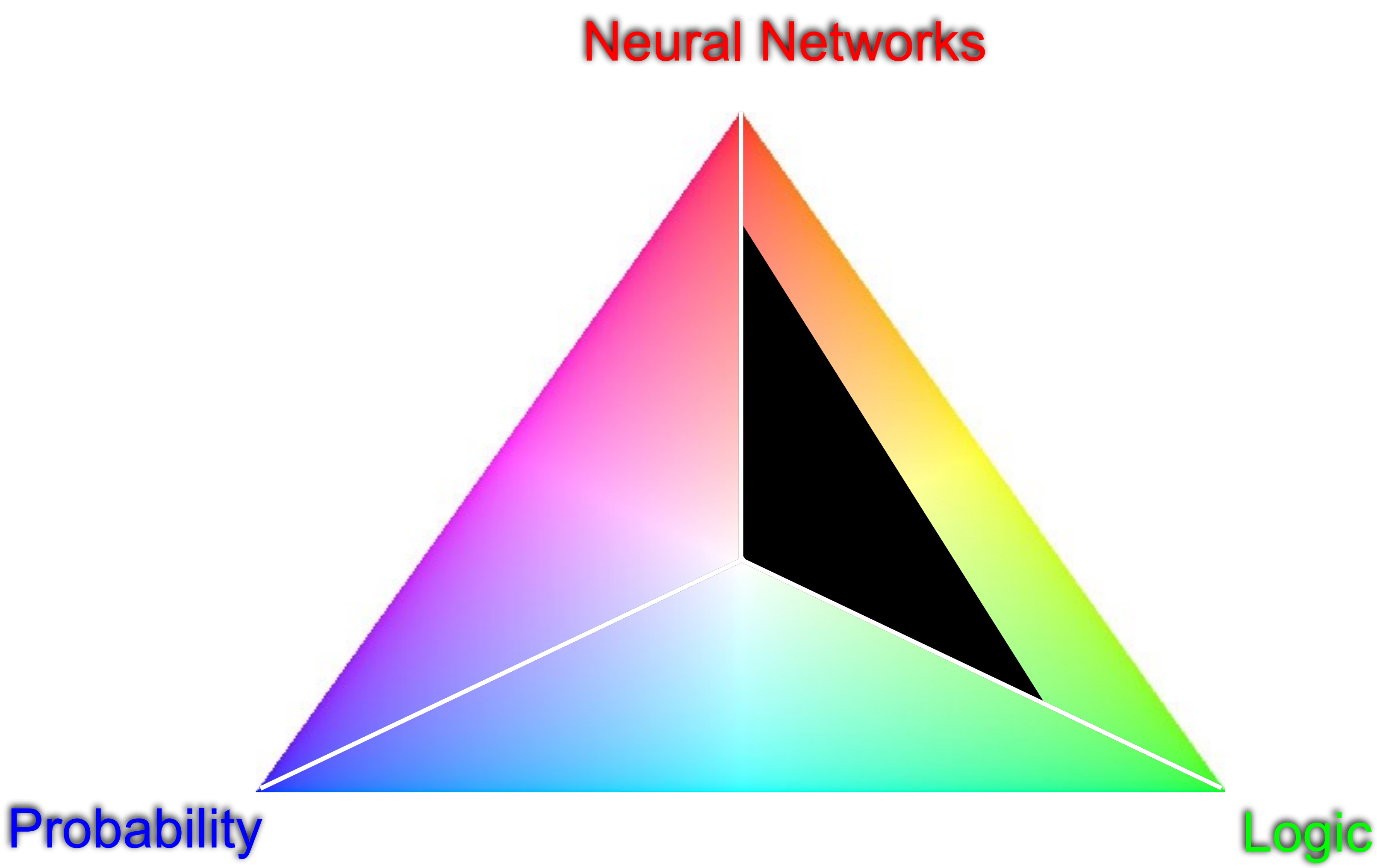
Neural networks cannot be recovered.

Logic is declined to encoding relations

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

Most scalable solutions.

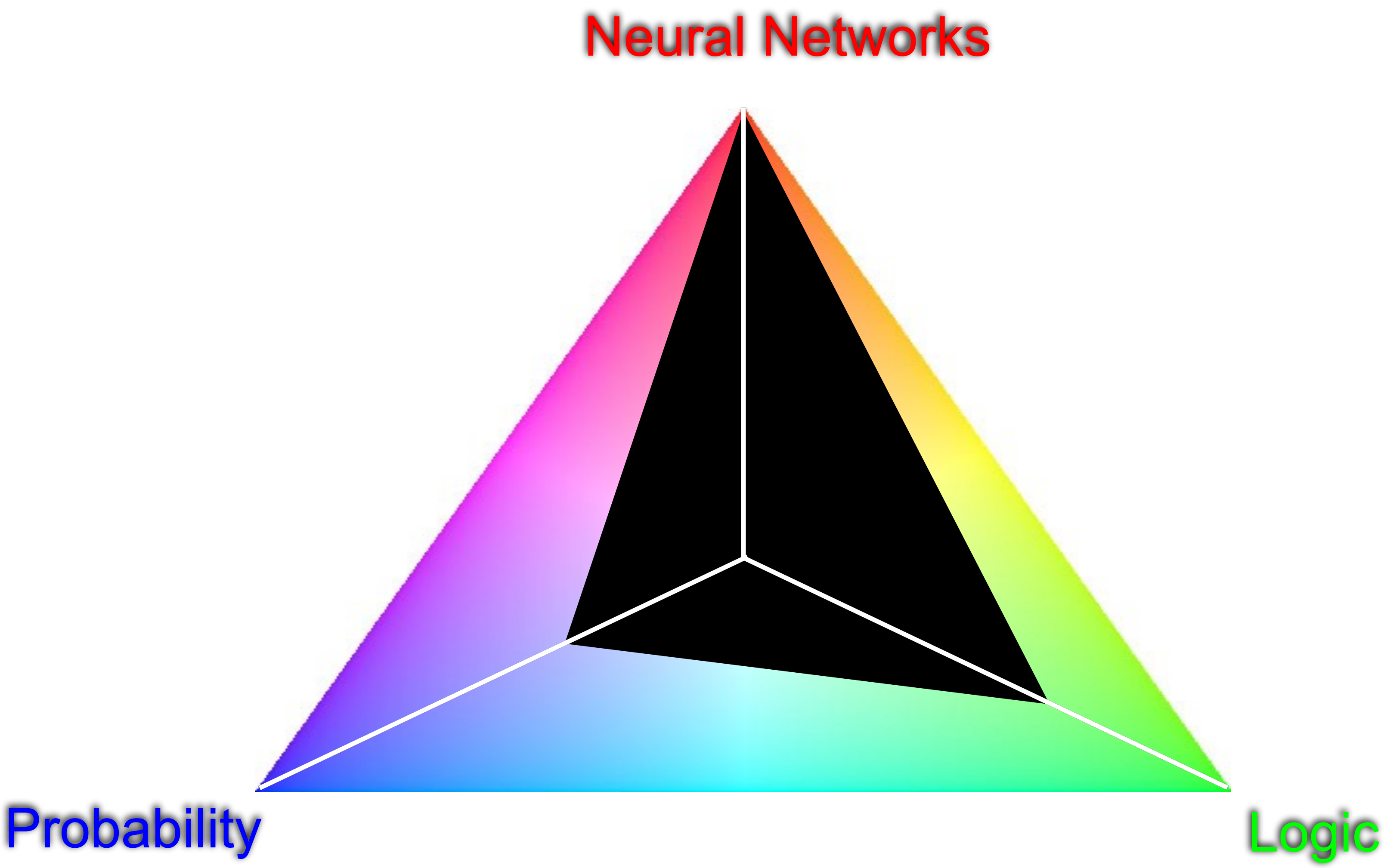
Relaxed theorem provers



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

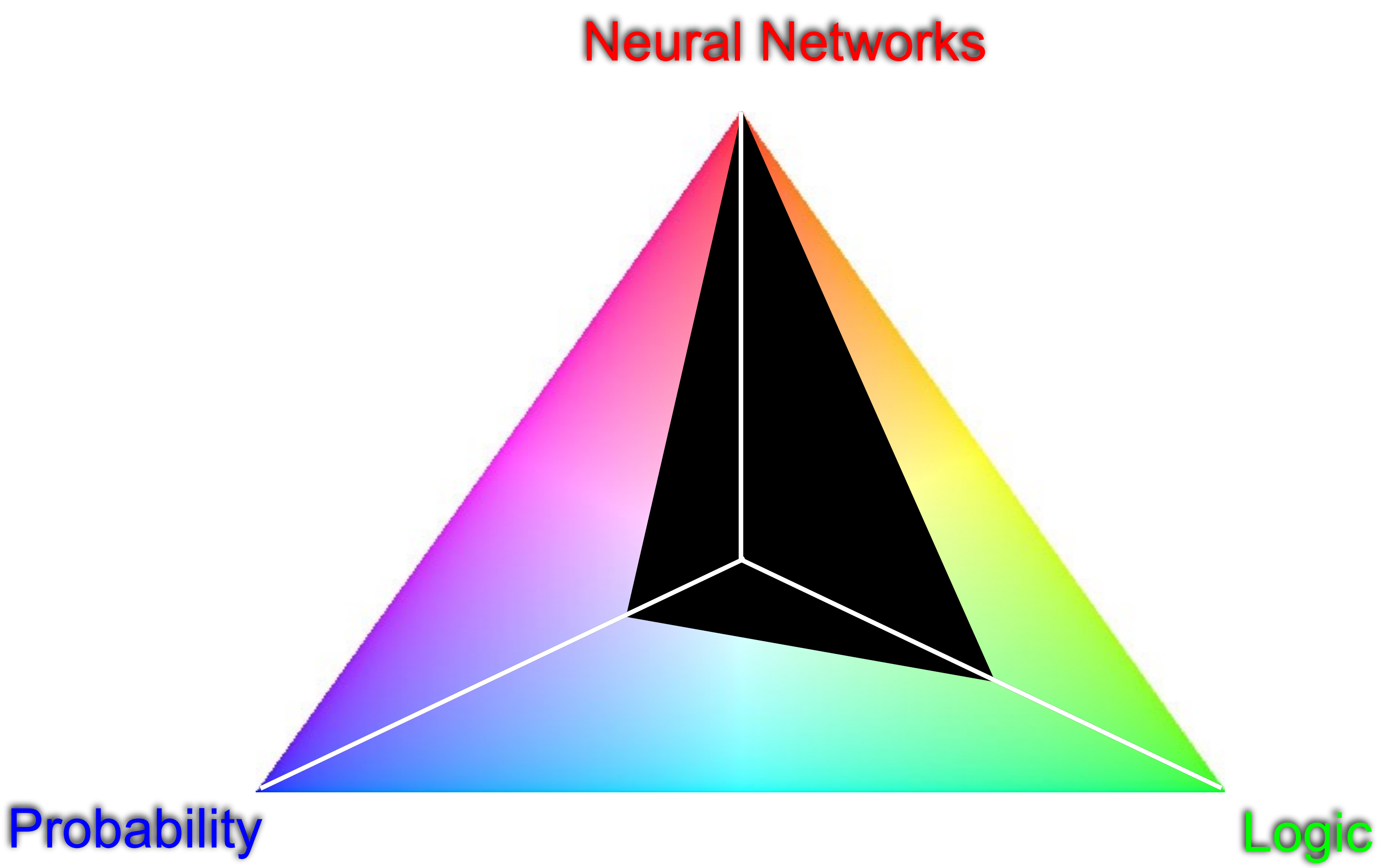


They sacrifice the logic and probability a lot by pushing everything inside the weights of the neural network.

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)

Graph Neural Networks

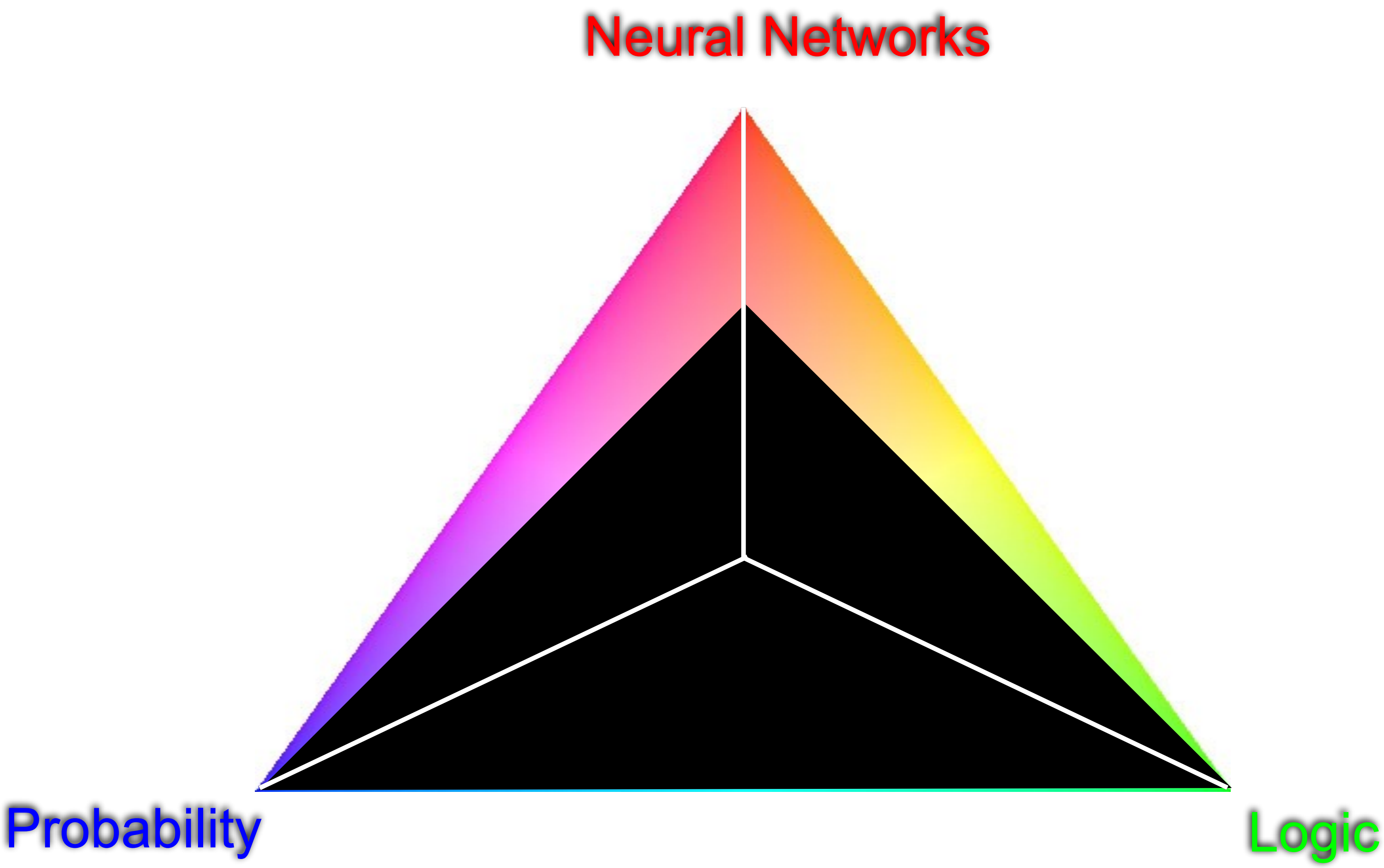


They extend neural network to provide some relational and multihop reasoning.

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

A Recipe for NeSy

One NeSy Recipe

1. Take a symbolic (logic / rule based) representation
2. Turn the 0/1 True/False in Fuzzy or Probabilistic Interpretation
3. Interpret neural networks as logical predicates/functions,
4. (The harder part): inference and learning

For instance:

map an MNIST image to a number

m() = 2

m as a neural network

mp(,2) =0.93 as a neural predicate

(with a fuzzy/prob. interpretation)

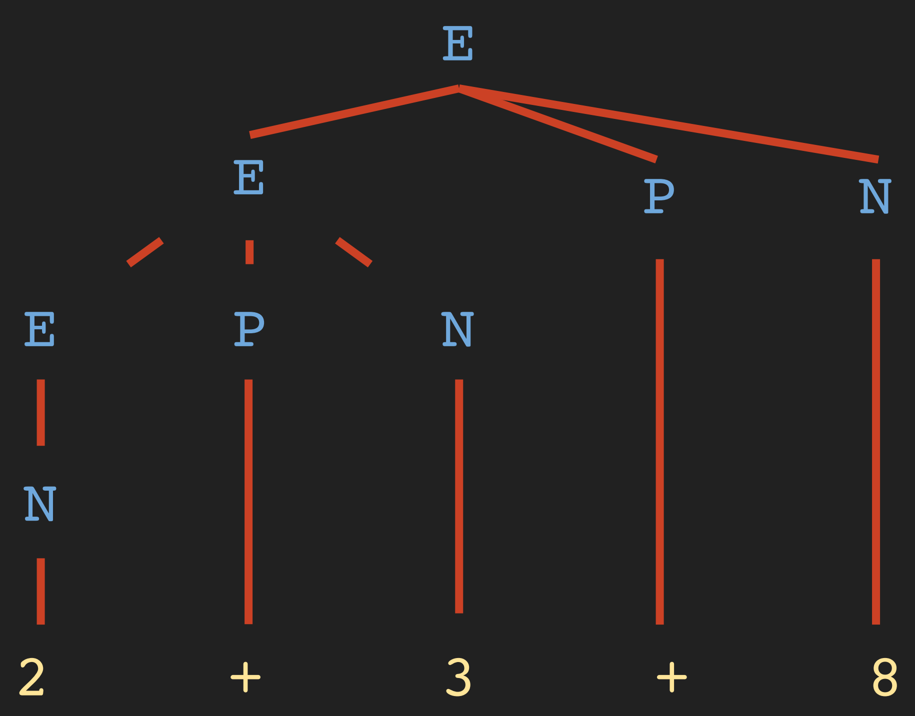
DeepStochLog

- Little sibling of DeepProbLog [Winters, Marra, et al AAAI 22]
- Based on a different semantics
  - probabilistic graphical models vs grammars
  - random graphs vs random walks
- Underlying StarAI representation is Stochastic Logic Programs (Muggleton, Cussens)
  - close to Probabilistic Definite Clause Grammars, ako probabilistic unification based grammar formalism
  - again the idea of neural predicates
- Scales better, is faster than DeepProbLog

Neural Definite Clause Grammar

CFG: Context-Free Grammar

```
E --> N
E --> E, P, N

P --> ["+"]

N --> ["0"]
N --> ["1"]
...
N --> ["9"]
```

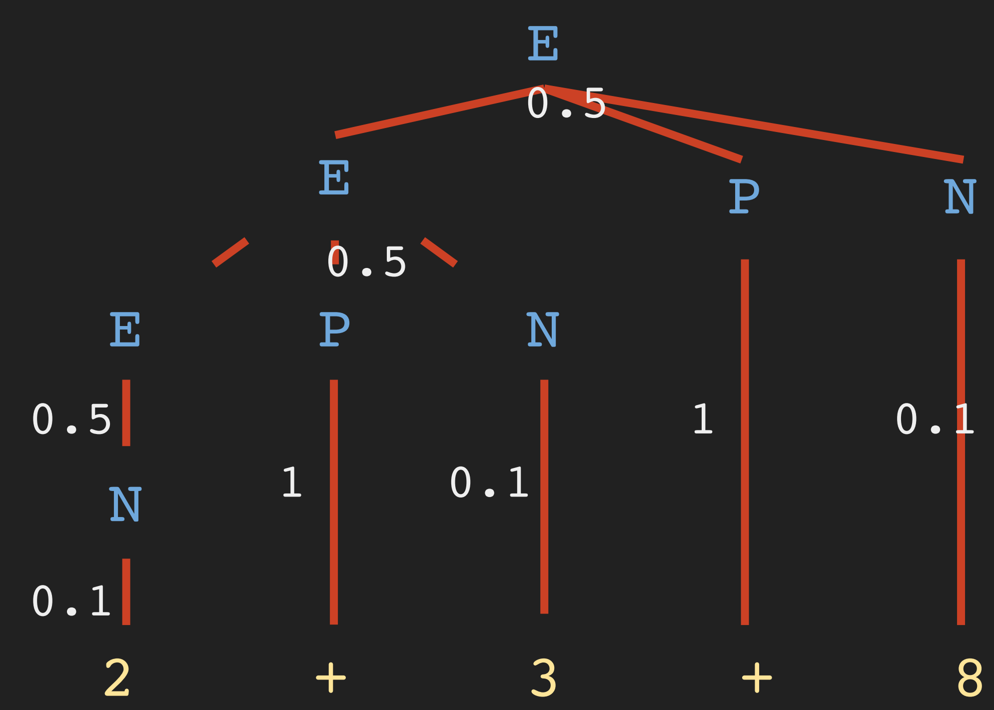


Useful for:
- Is sequence an element of the specified language?
- What is the *"part of speech"*-tag of a terminal
- Generate all elements of language

PCFG: Probabilistic Context-Free Grammar

```
0.5 :: E --> N
0.5 :: E --> E, P, N

1.0 :: P --> ["+"]

0.1 :: N --> ["0"]
0.1 :: N --> ["1"]
         …
0.1 :: N --> ["9"]
```

Always sums to 1 per non-terminal



*Probability of this parse = 0.5*0.5*0.5*0.1*1*0.1*1*0.1*

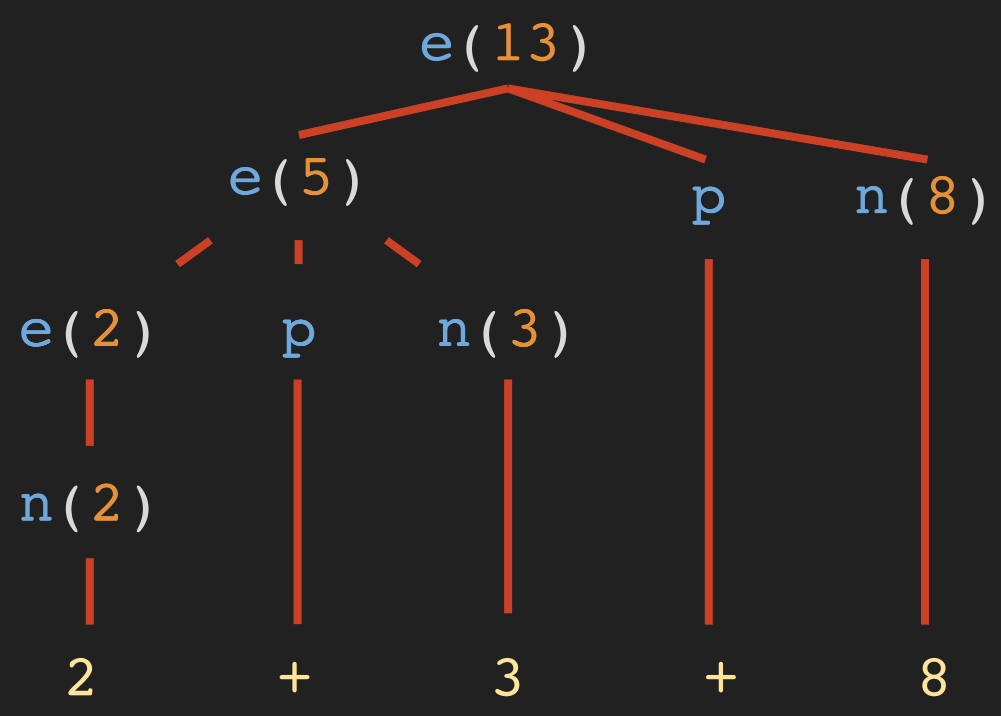
= 0.000125

Useful for:

- What is the most likely parse for this sequence of terminals? *(useful for ambiguous grammars)*
- What is the probability of generating this string?

DCG: Definite Clause Grammar

```
e(N) --> n(N).
e(N) --> e(N1), p, n(N2),
         {N is N1 + N2}.
p    --> ["+"].

n(0) --> ["0"].
n(1) --> ["1"].
…
n(9) --> ["9"].
```

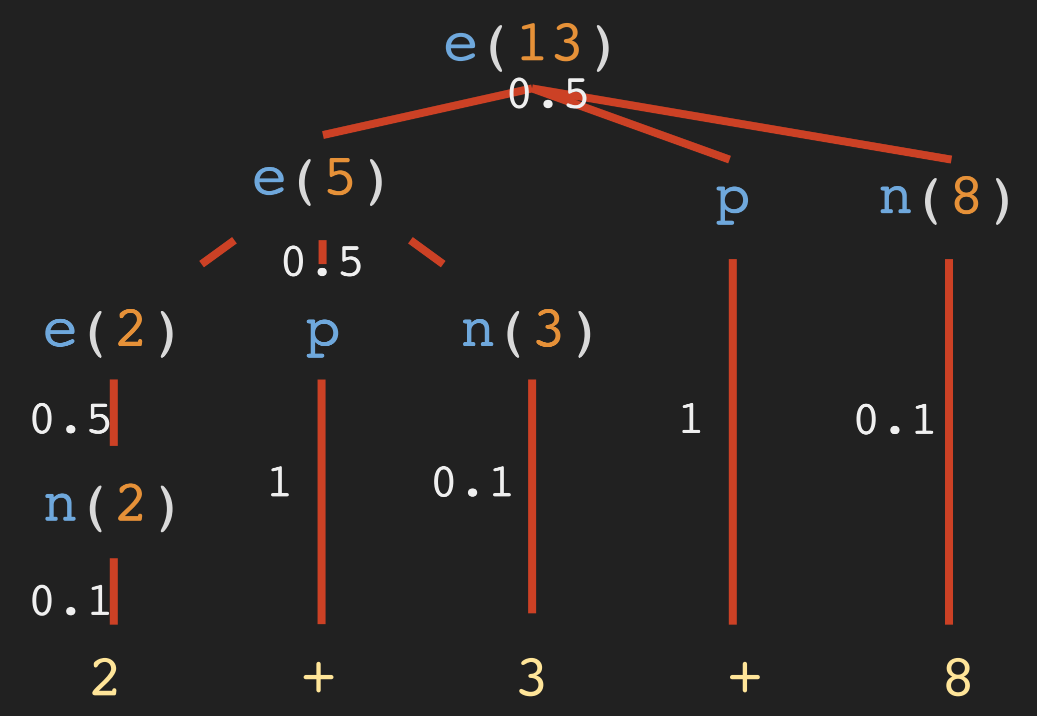


Useful for:

- Modelling more complex languages *(e.g. context-sensitive)*
- Adding constraints between non-terminals thanks to Prolog power *(e.g. through unification)*
- Extra inputs & outputs aside from terminal sequence *(through unification of input variables)*

SDCG: Stochastic Definite Clause Grammar

```
0.5 :: e(N) --> n(N).
0.5 :: e(N) --> e(N1), p, n(N2),
                {N is N1 + N2}.
1.0 :: p    --> ["+"].

0.1 :: n(0) --> ["0"].
0.1 :: n(1) --> ["1"].
       …
0.1 :: n(9) --> ["9"].
```



*Probability of this parse = 0.5*0.5*0.5*0.1*1*0.1*1*0.1*

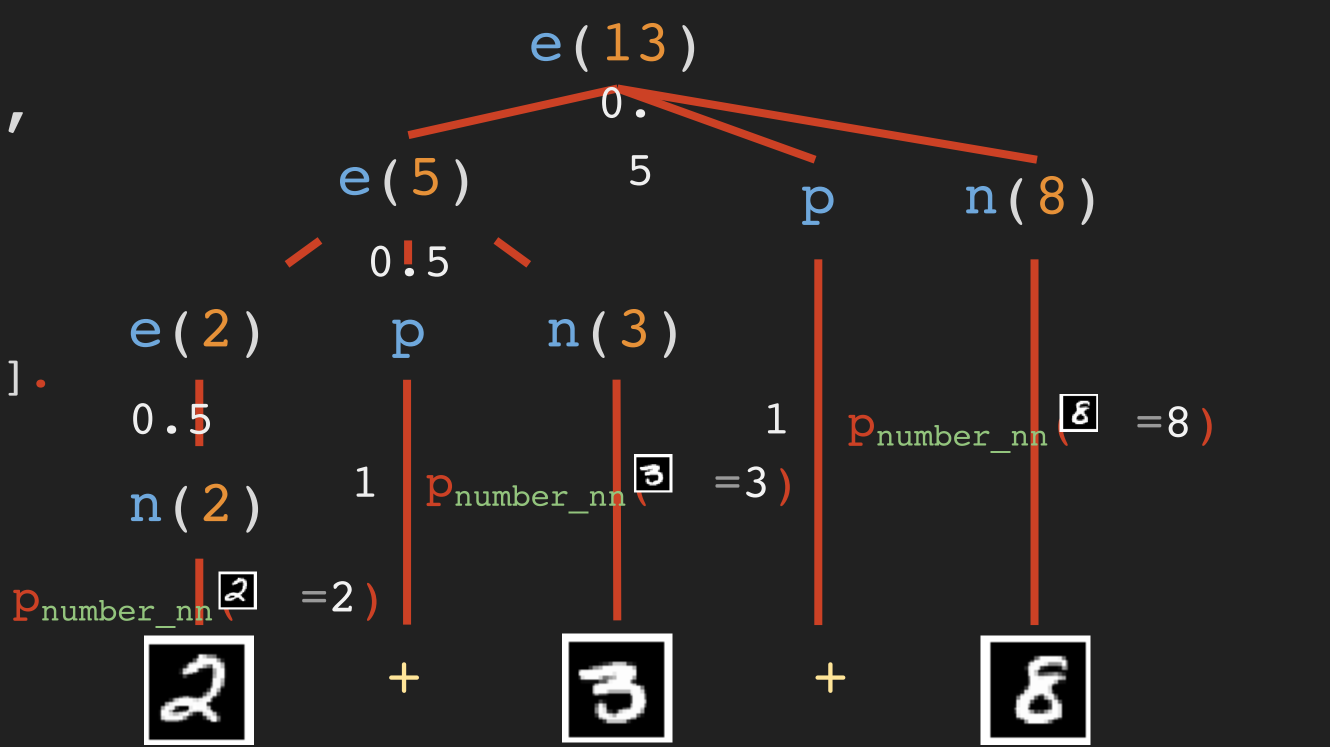
= 0.000125

Useful for:

- Same benefits as PCFGs give to CFG *(e.g. most likely parse)*
- But: loss of probability mass possible due to failing derivations

NDCG: Neural Definite Clause Grammar (= DeepStochLog)

```
0.5 :: e(N) --> n(N).
0.5 :: e(N) --> e(N1), p, n(N2),
                {N is N1 + N2}.
1.0 :: p    --> ["+"].

nn(number_nn,[X],[Y],[digit]) :: n(Y) --> [X].

digit(Y) :- member(Y,[0,1,2,3,4,5,6,7,8,9]).
```



Probability of this parse =

$0.5*0.5*0.5*p_{number\_nn}(\text{[2]}=2)*1*p_{number\_nn}(\text{[3]}=3)*1*p_{number\_nn}(\text{[8]}=8)$

Useful for:

- Subsymbolic processing: e.g. tensors as terminals
- Learning rule probabilities using neural networks

DeepStochLog NDCG definition

$$\texttt{nn(m,[}I_1,\ldots,I_m\texttt{],[}O_1,\ldots,O_L\texttt{],[}D_1,\ldots,D_L\texttt{]) :: nt --> } g_1, \ldots, g_n.$$

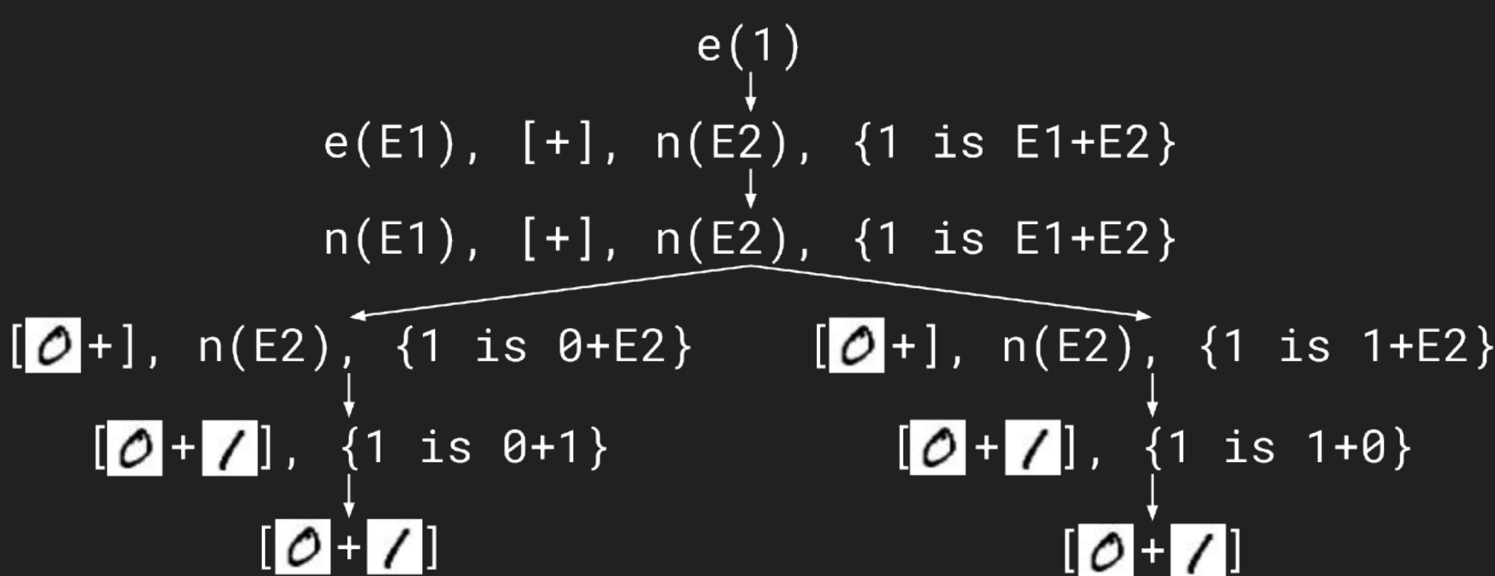
Where:

- `nt` is an atom
- $g_1, \ldots, g_n$ are goals *(goal = atom or list of terminals & variables)*
- $I_1,\ldots,I_m$ and $O_1,\ldots,O_L$ are variables occurring in $g_1, \ldots, g_n$ and are the inputs and outputs of `m`
- $D_1,\ldots,D_L$ are the predicates specifying the domains of $O_1,\ldots,O_L$
- `m` is a neural network mapping $I_1,\ldots,I_m$ to probability distribution over $O_1,\ldots,O_L$ *(= over cross product of $D_1,\ldots,D_L$)*

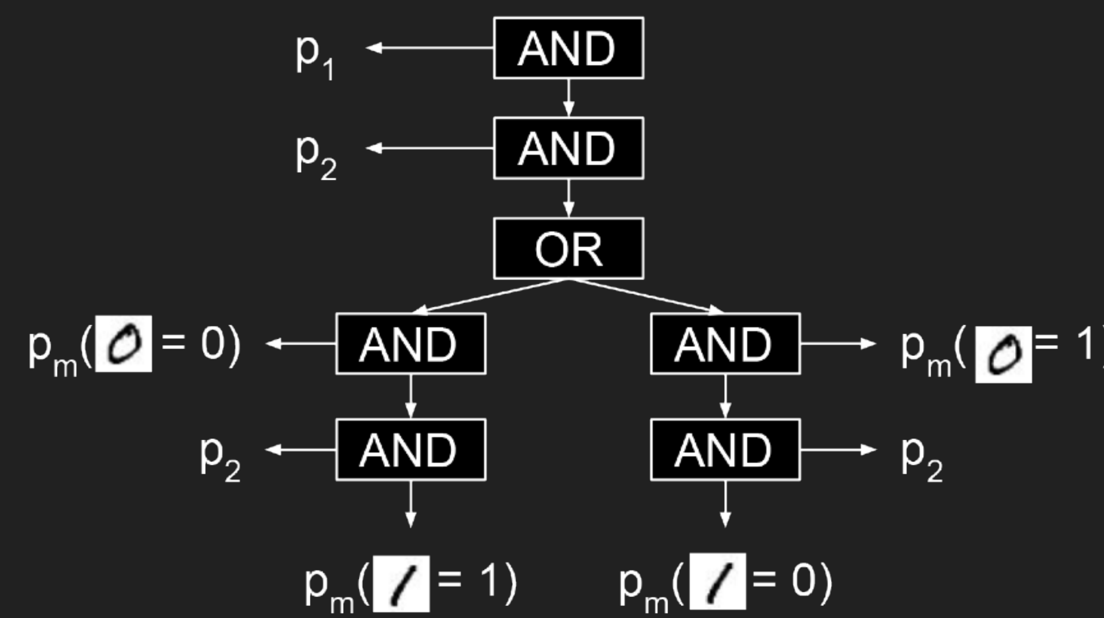
DeepStochLog Inference

Deriving probability of goal for given terminals in NDCG

Proof derivations d(e(1),[+])



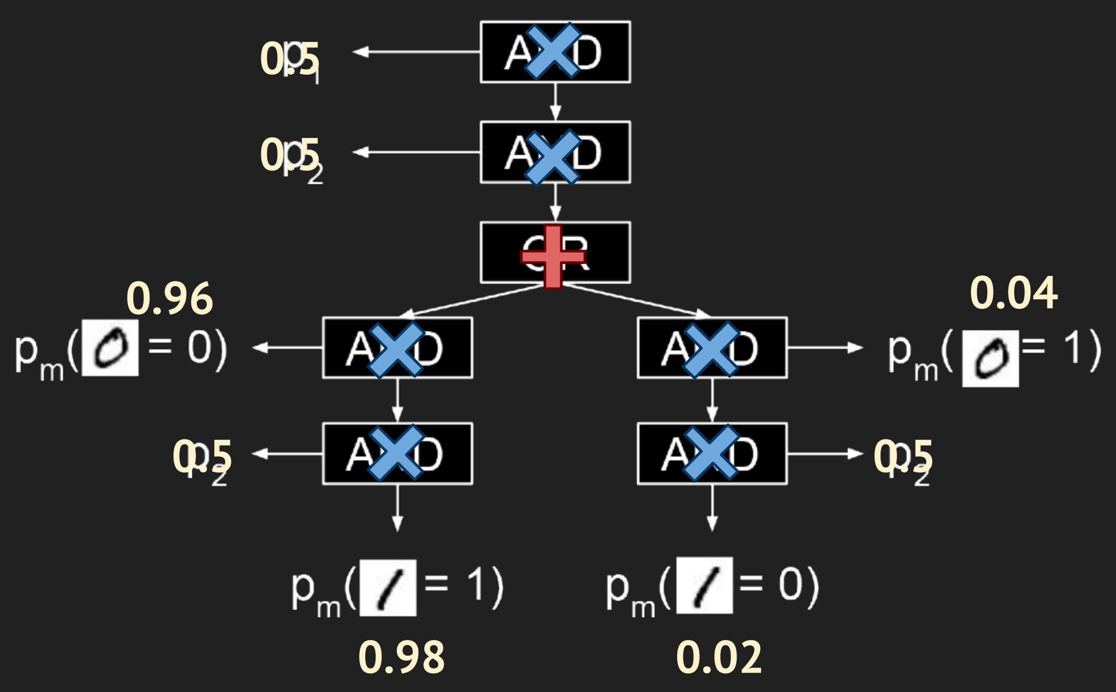
then turn it into and/or tree



And/Or tree + semiring for different inference types

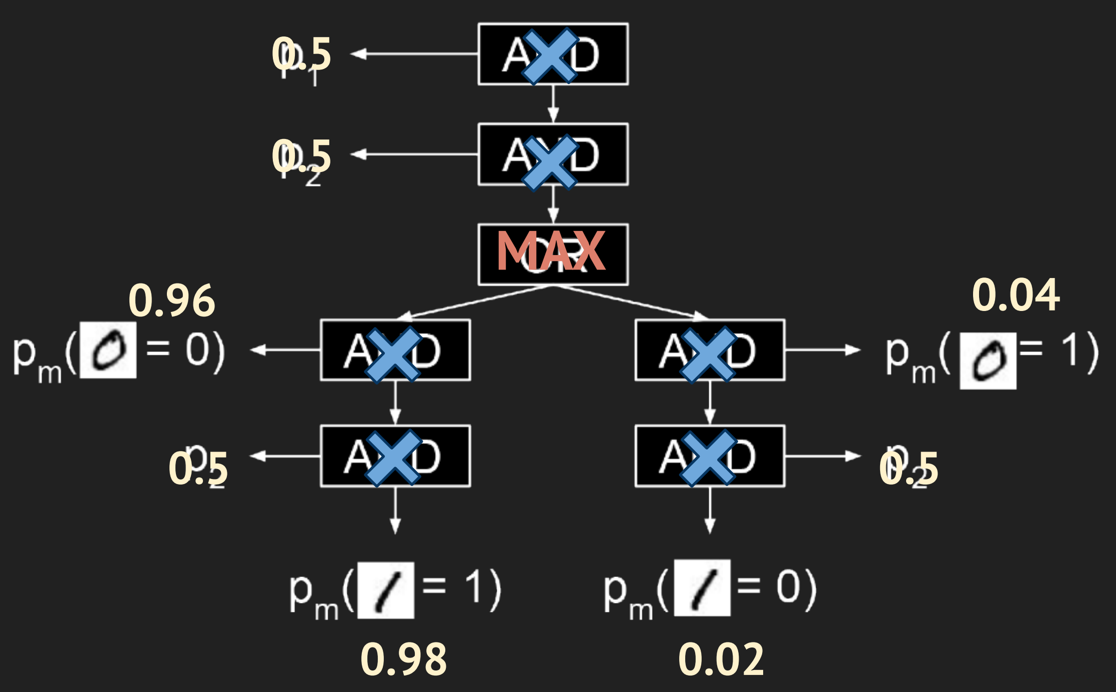
Probability of goal

P_G(derives(e(1), [0, +, 1]) = 0.1141



Most likely derivation

d_{max}(e(1), [0, +, 1]) = $\text{argmax}_{d(e(t))=[0,+,1]} P_G(d(e(1)))$ = [0,+,1]



Inference optimisation

Inference is optimized using

1. **SLG resolution**: Prolog tables the returned proof tree(s), and thus creates forest

   → Allows for reusing probability calculation results from intermediate nodes

Table 6: **Q4** Parsing time in seconds (**T2**). Comparison of the DeepStochLog with and without tabling (SLD vs SLG resolution).

Lengths	# Answers	No Tabling	Tabling
1	10	0.067	0.060
3	95	0.081	0.096
5	1066	3.78	0.95
7	10386	30.42	10.95
9	68298	1494.23	132.26
11	416517	timeout	1996.09

1. **Batched network calls**: Evaluate all the required neural network queries first

   → Very natural for neural networks to evaluate multiple instances at once using batching
   & less overhead in logic & neural network communication

Research questions

Q1: Does DeepStochLog reach state-of-the-art predictive performance on neural-symbolic tasks?

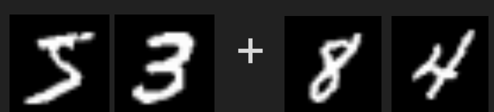
Q2: How does the inference time of DeepStochLog compare to other neural-symbolic frameworks and what is the role of tabling?

Q3: Can DeepStochLog handle larger-scale tasks?

Q4: Can DeepStochLog go beyond grammars and encode more general programs?

Mathematical expression outcome

T1: Summing MNIST numbers with pre-specified # digits



= 137

T2: Expressions with images representing operator or single digit number.

= 19

Table 1: The test accuracy (%) on the MNIST addition (**T1**).

Methods	Number of digits per number (N) 1	2	3	4
NeurASP	97.3 ± 0.3	93.9 ± 0.7	timeout	timeout
DeepProbLog	97.2 ± 0.5	95.2 ± 1.7	timeout	timeout
DeepStochLog	97.9 ± 0.1	96.4 ± 0.1	94.5 ± 1.1	92.7 ± 0.6

Table 2: The accuracy (%) on the HWF dataset (**T2**).

Method	Expression length 1	3	5	7
NGS	90.2 ± 1.6	85.7 ± 1.0	91.7 ± 1.3	20.4 ± 37.2
DeepProbLog	90.8 ± 1.3	85.6 ± 1.1	timeout	timeout
DeepStochLog	90.8 ± 1.0	86.3 ± 1.9	92.1 ± 1.4	94.8 ± 0.9

Performance comparison

Table 7: Inference times in milliseconds for DeepStochLog, DeepProbLog and NeurASP on task **T1** for variable number lengths.

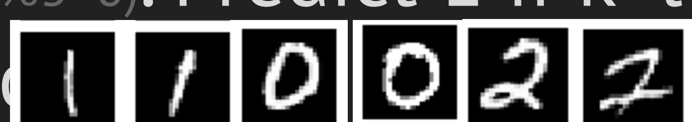
Numbers Length	1	2	3	4
DeepStochLog	1.3 ± 0.9	2.3 ± 0.4	4.0 ± 0.4	5.7 ± 1.8
DeepProbLog	13.5 ± 3.0	36.0 ± 0.5	199.7 ± 14.0	timeout
NeurASP	9.2 ± 1.4	85.7 ± 22.6	158.2 ± 47.7	timeout

Classic grammars, but with MNIST images as terminals

T3: Well-formed brackets as input (without parse). Task: predict parse.

→ parse = ( ) (( ) ( ))

T4: inputs are strings $a^k b^l c^m$ (or permutations of [a,b,c], and (k+l+m) %3=0). Predict 1 if k=l=m,

 = 1

 = 0

Table 3: The parse accuracy (%) on the well-formed parentheses dataset (**T3**).

Method	Maximum expression length 10	14	18
DeepProbLog	100.0 ± 0.0	99.4 ± 0.5	99.2 ± 0.8
DeepStochLog	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0

Table 4: The accuracy (%) on the $a^n b^n c^n$ dataset (**T4**).

Method	Expression length 3-12	3-15	3-18
DeepProbLog	99.8 ± 0.3	timeout	timeout
DeepStochLog	99.4 ± 0.5	99.2 ± 0.4	98.8 ± 0.2

Natural way of expressing this grammar knowledge

```
brackets_dom(X) :- member(X, ["(",")"]).
nn(bracket_nn, [X], Y, brackets_dom) :: bracket(Y) --> [X].


t(_) :: s --> s, s.
t(_) :: s --> bracket("("), s, bracket(")").
t(_) :: s --> bracket("("), bracket(")").
```

All power of Prolog DCGs (here: $a^n b^n c^n$)

```
letter(X) :- member(X, [a,b,c]).

0.5 :: s(0) --> akblcm(K,L,M),
                {K \= L; L \= M; M \= K},
                {K \= 0, L \= 0, M \= 0}.

0.5 :: s(1) --> akblcm(N,N,N).

akblcm(K,L,M) --> rep(K,A),
                  rep(L,B),
                  rep(M,C),
                  {A \= B, B \= C, C \= A}.

rep(0, _) --> [].
nn(mnist, [X], C, letter) :: rep(s(N), C) --> [X],
                                               rep(N,C).
```

Citation networks

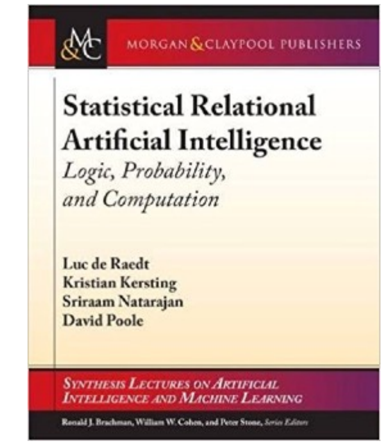
T5: Given scientific paper set with only few labels & citation network, find all labels

Table 5: **Q3** Accuracy (%) of the classification on the test nodes on task **T5**

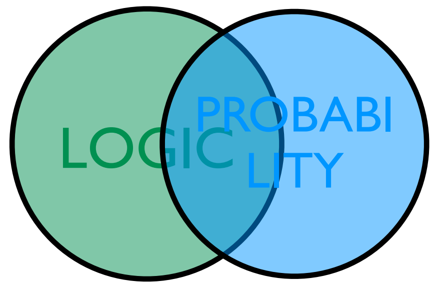
Method	Citeseer	Cora
ManiReg	60.1	59.5
SemiEmb	59.6	59.0
LP	45.3	68.0
DeepWalk	43.2	67.2
ICA	69.1	75.1
GCN	70.3	81.5
DeepProbLog	timeout	timeout
DeepStochLog	65.0	69.4

Conclusions

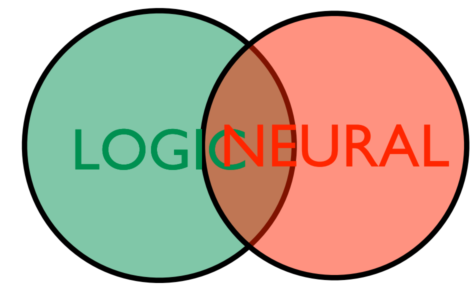
Key Message



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

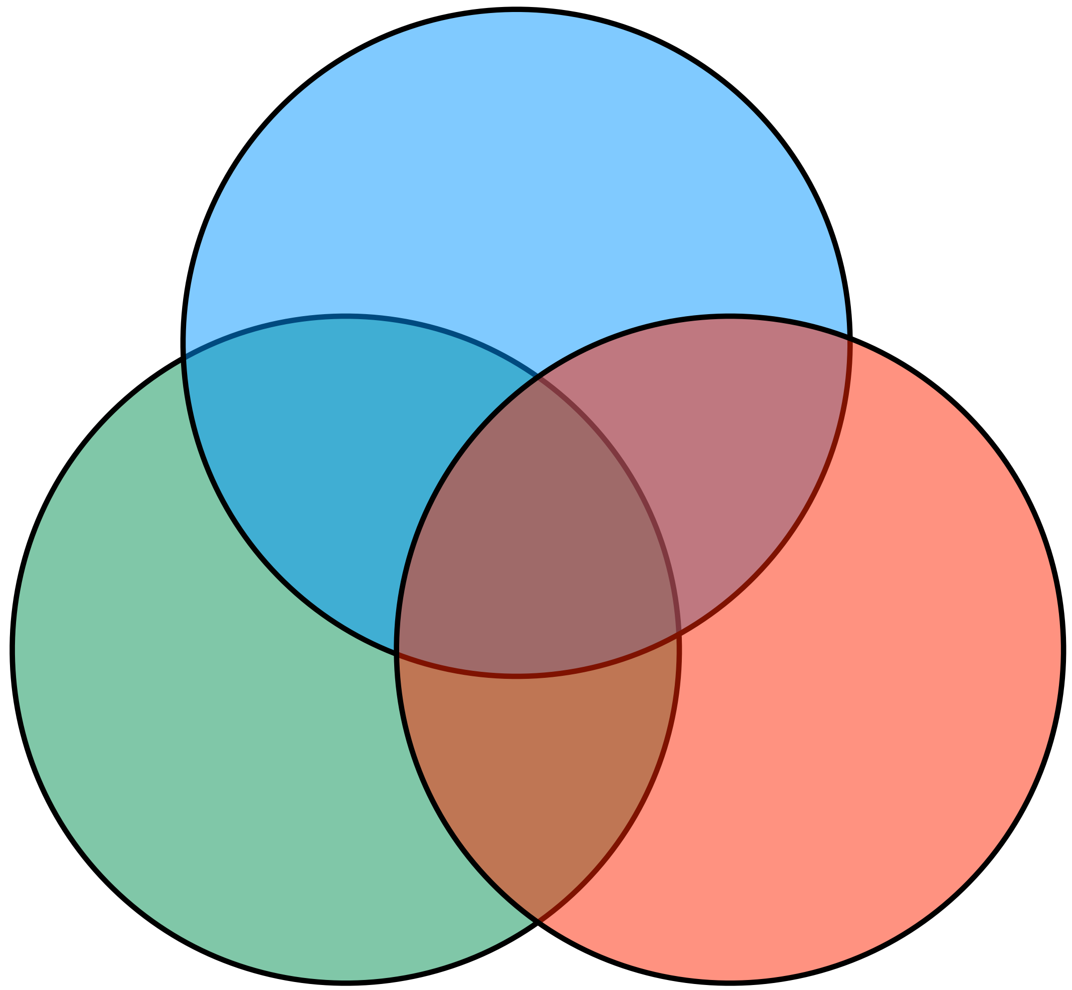
- What properties should integrated representations satisfy ?
  - Should one representation take over ? (As in most approaches to NeSy — push the logic inside and forget about it afterwards)
  - Should one build a pipeline or an interface between the integrated representations ?
  - Should one have the 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˘ci´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˘ci´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