From Statistical Relational AI to Neural Symbolic Computation

Giuseppe Marra
giuseppe.marra@kuleuven.be

reusing some slides from previous tutorials with
Nesy : Luc De Raedt, Sebastijan Dumancin, Robin Manhaeve
StarAI : Angelika Kimmig, Kristian Kersting, David Poole, and Sriraam Natarajan

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

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

DTAI Machine Learning
KU LEUVEN
LEUVEN.AI INSTITUTE
fwo

Introduction

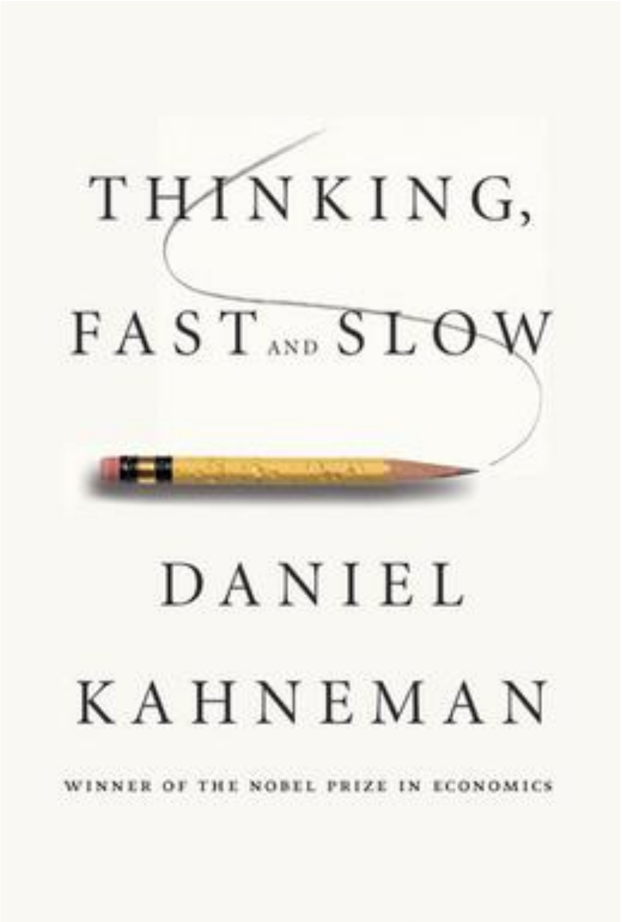
How much effort do you need to solve these tasks?

Is she smiling?

The result of …

147 x 13

Thinking fast and slow



Real-life problems involve both aspects.

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

Who can go first ?

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

Real-life problems involve both aspects.

Who can go first ?

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

Thinking fast

Thinking slow

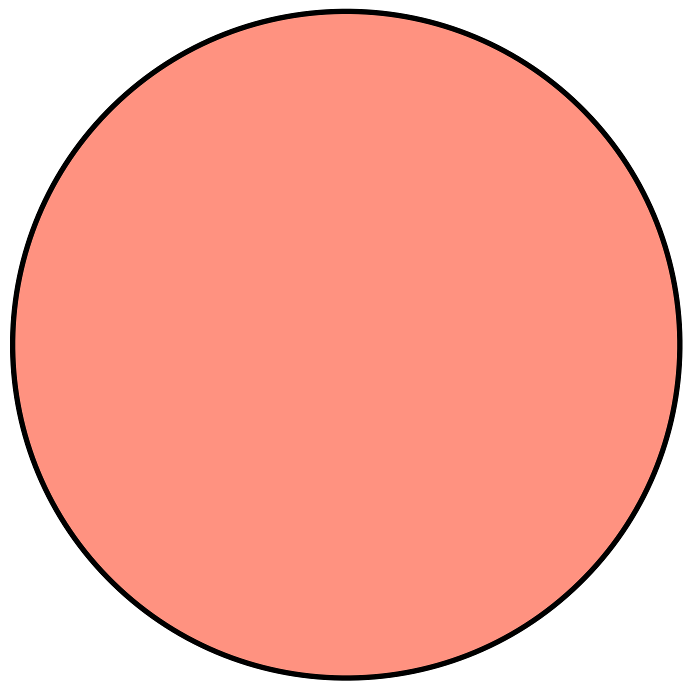
Thinking fast and slow in AI

Subsymbolic (Thinking fast)	Symbolic (Thinking slow)
associative	logical
data	knowledge
learning	reasoning/planning
noisy input	precise input

Thinking fast

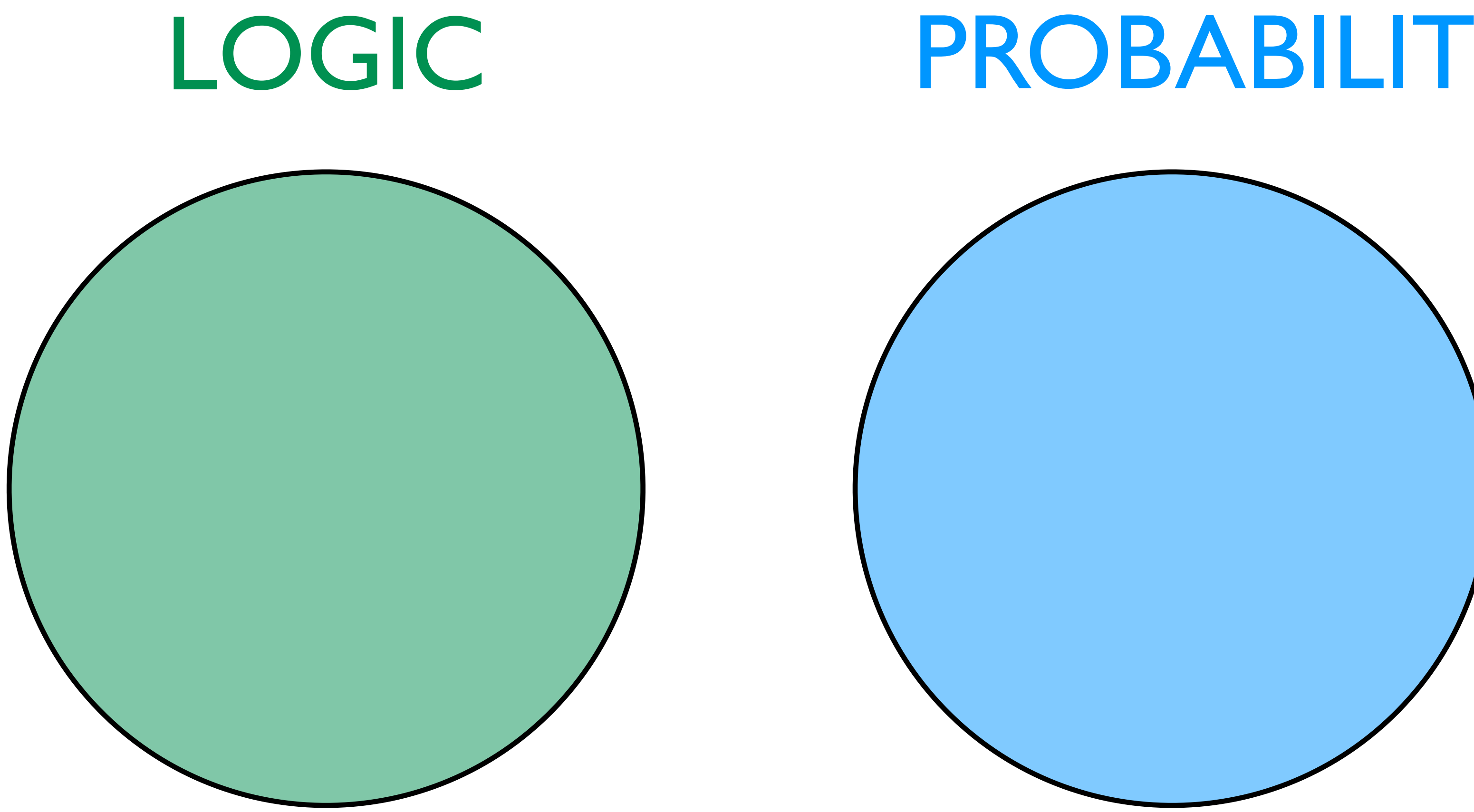
MAIN PARADIGM in AI

NEURAL

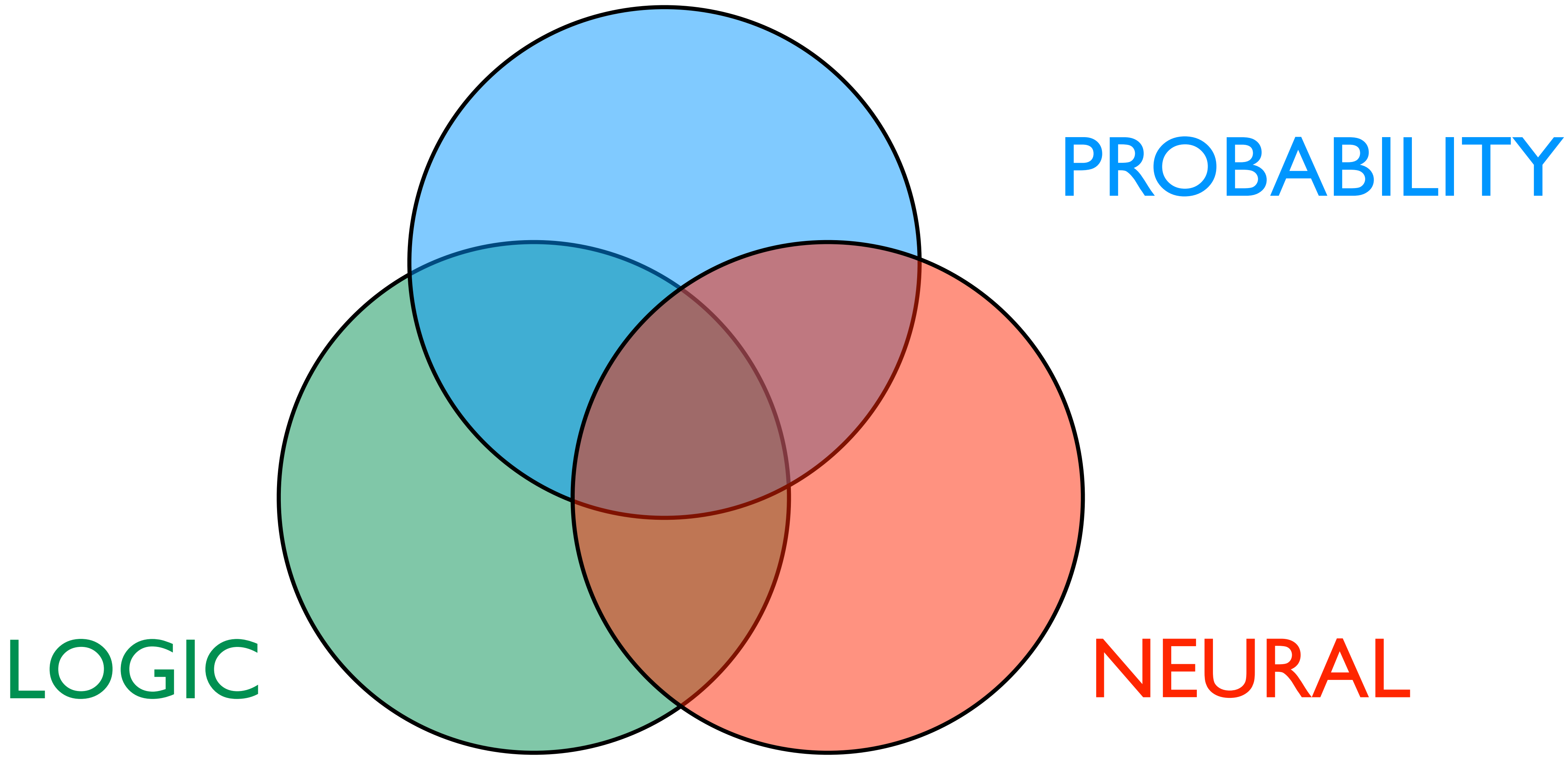


Thinking slow = reasoning

TWO MAIN PARADIGMS in AI

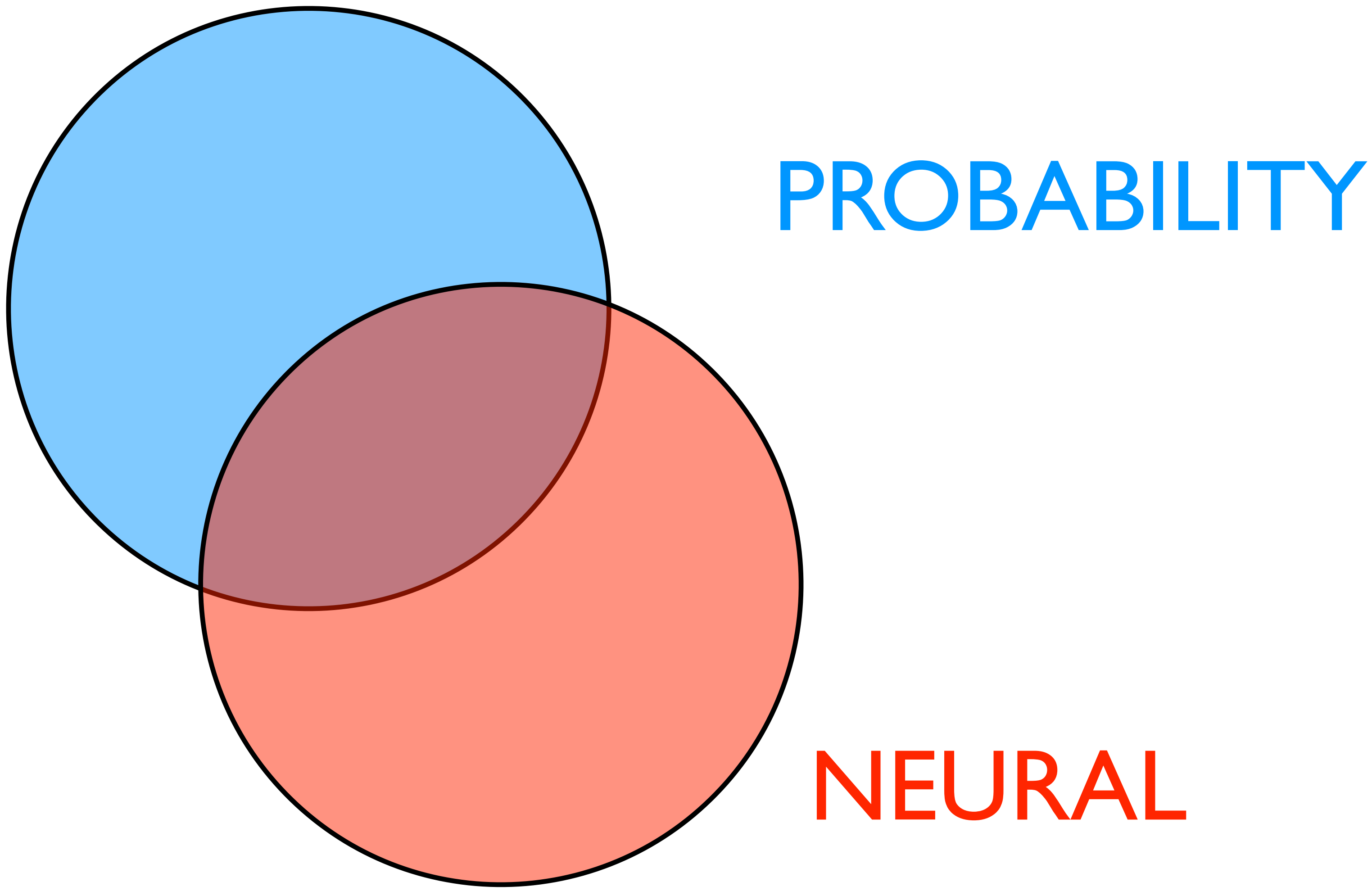


Integration



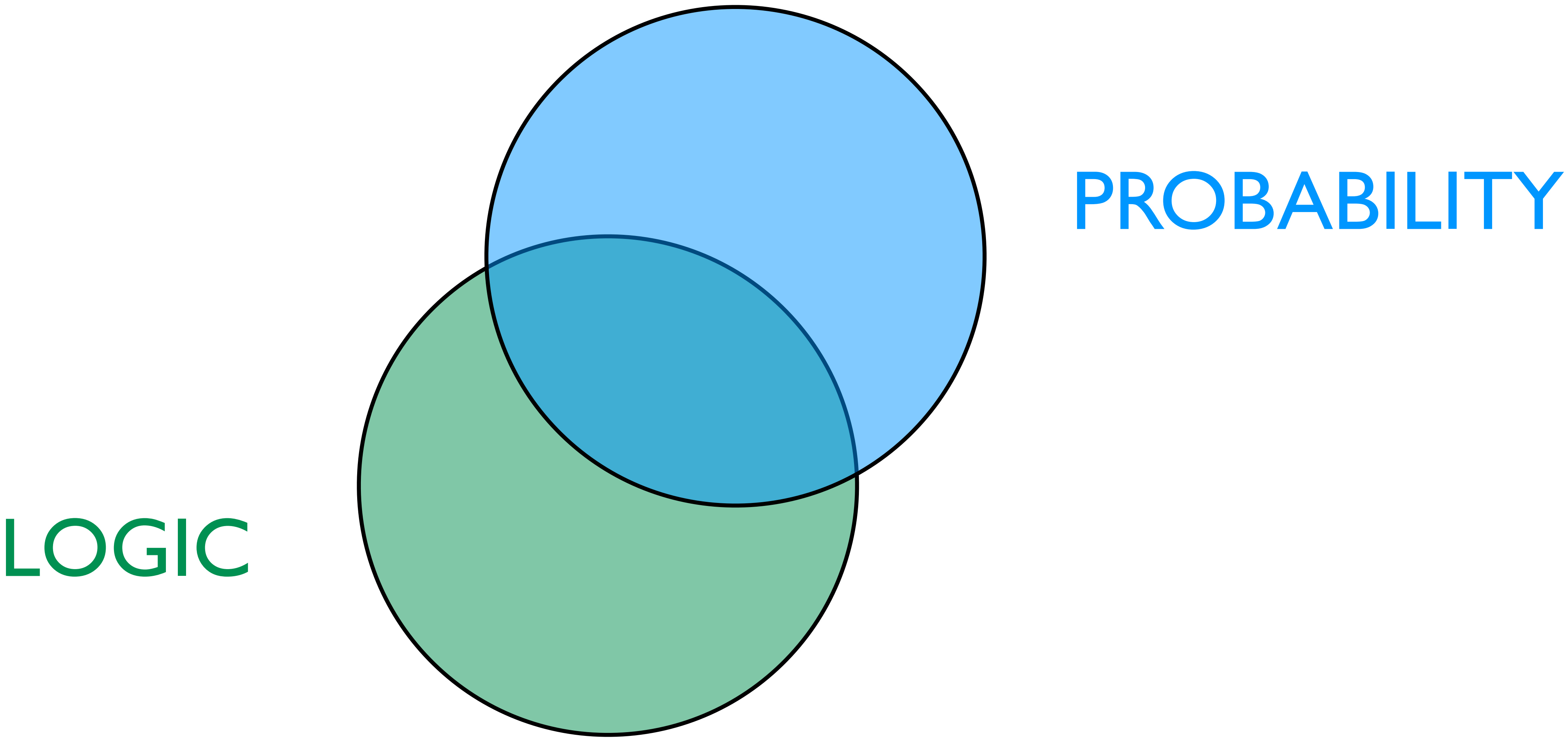
How to integrate these three paradigms in AI ?

Deep Learning

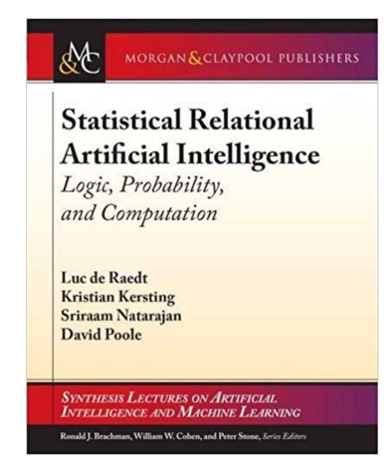


Well studied from a LEARNING perspective

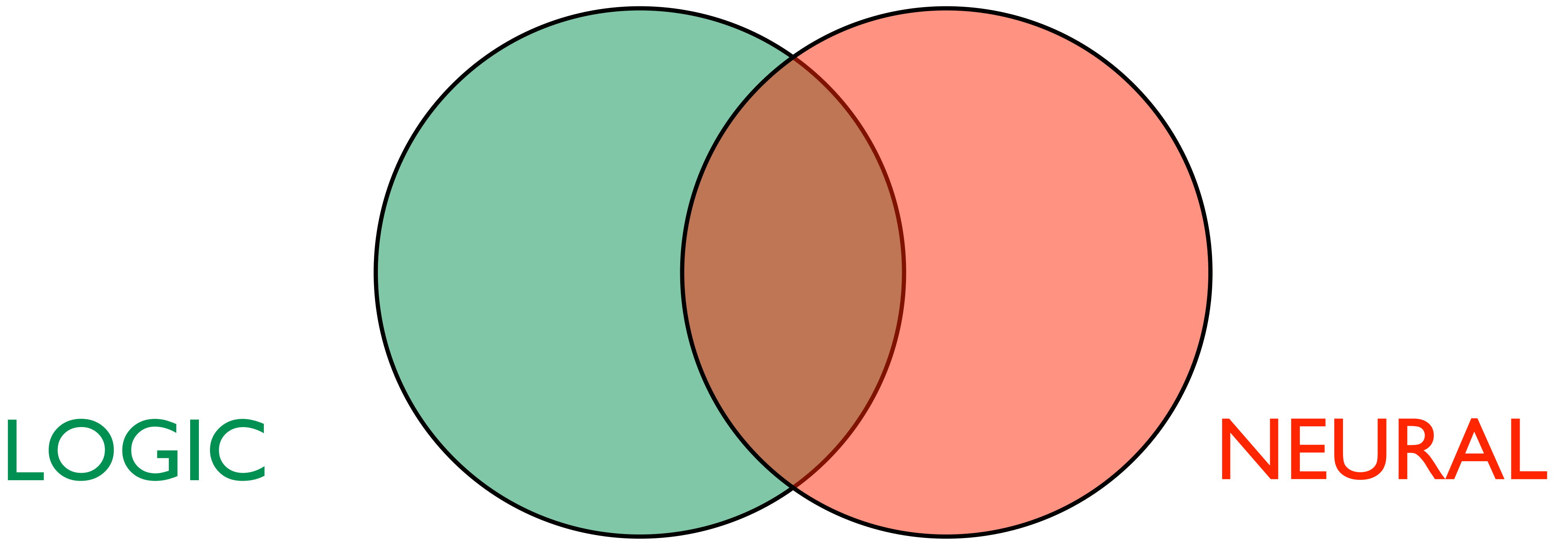
Statistical Relational AI



Their integration has been well studied in Statistical Relational AI (StarAI)



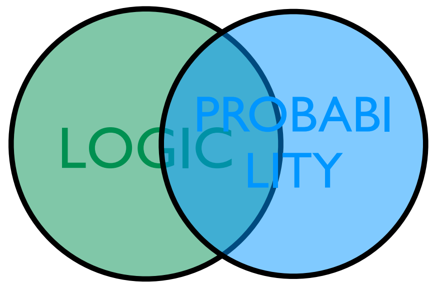
Neural Symbolic



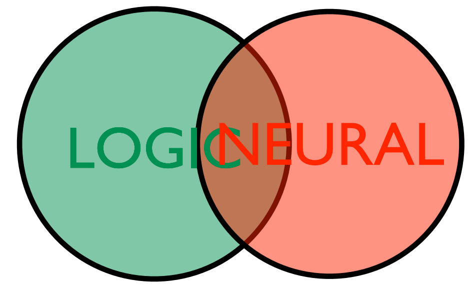
Being studied from a LEARNING perspective in Neuro Symbolic Computation

Key Message

FROM



TO

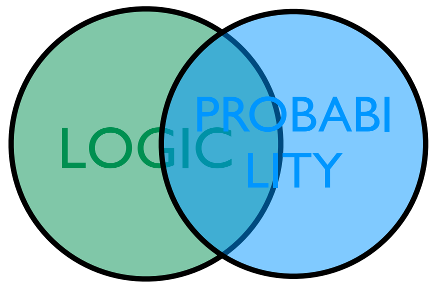


StarAI and NeSy share similar problems and thus similar solutions apply

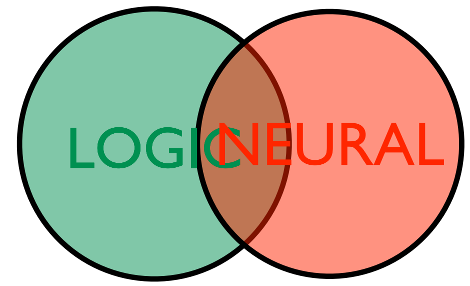
See also
De Raedt, Dumancic, Marra, Manhaeve
From Statistical Relational to Neuro-Symbolic Artificial Intelligence
IJCAI 20

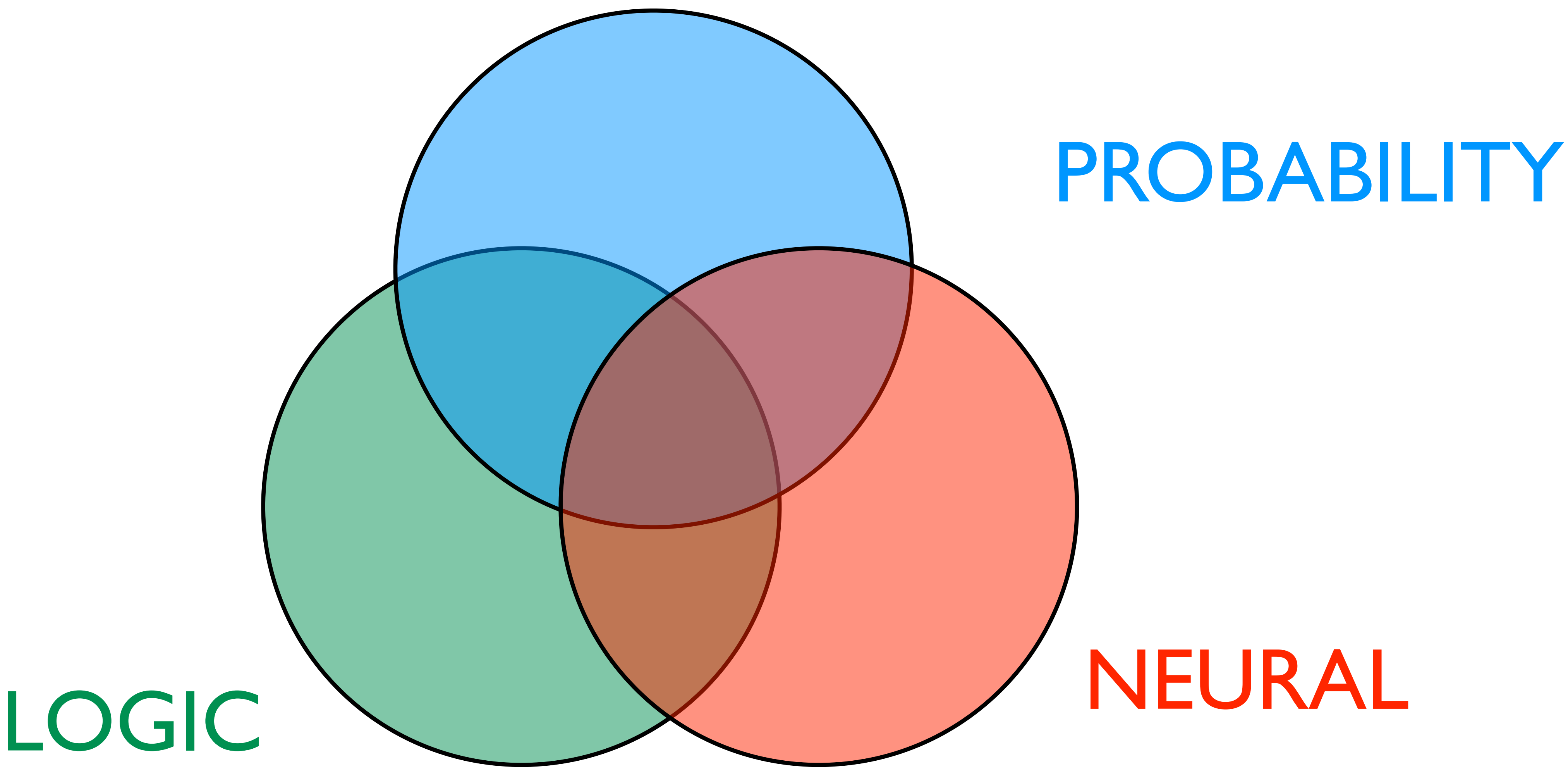
Goal

FROM



TO

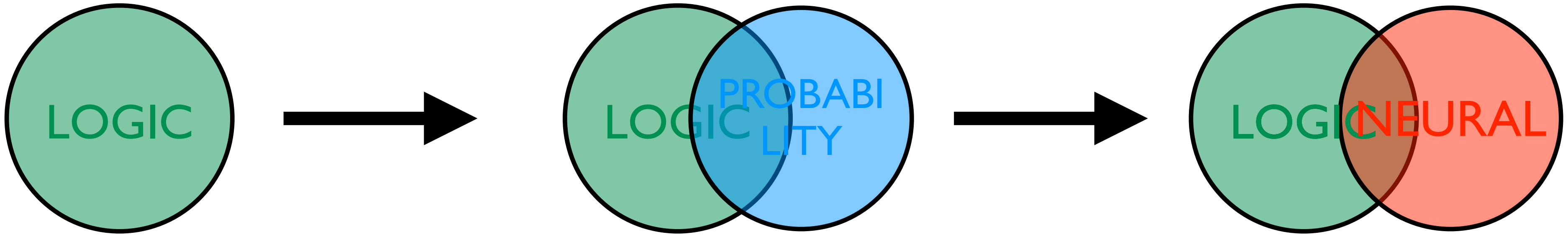




The Seven Dimensions

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

Logic Programs

as in the programming language Prolog

Propositional logic program

```
burglary.
hears_alarm_mary.

earthquake.
hears_alarm_john.
```

facts :
burglary = true

```
alarm :– earthquake.

alarm :– burglary.

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.
```

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

LOGIC

Logic Programs

as in the programming language Prolog

Propositional logic program

burglary.
hears_alarm_mary.

earthquake.
hears_alarm_john.

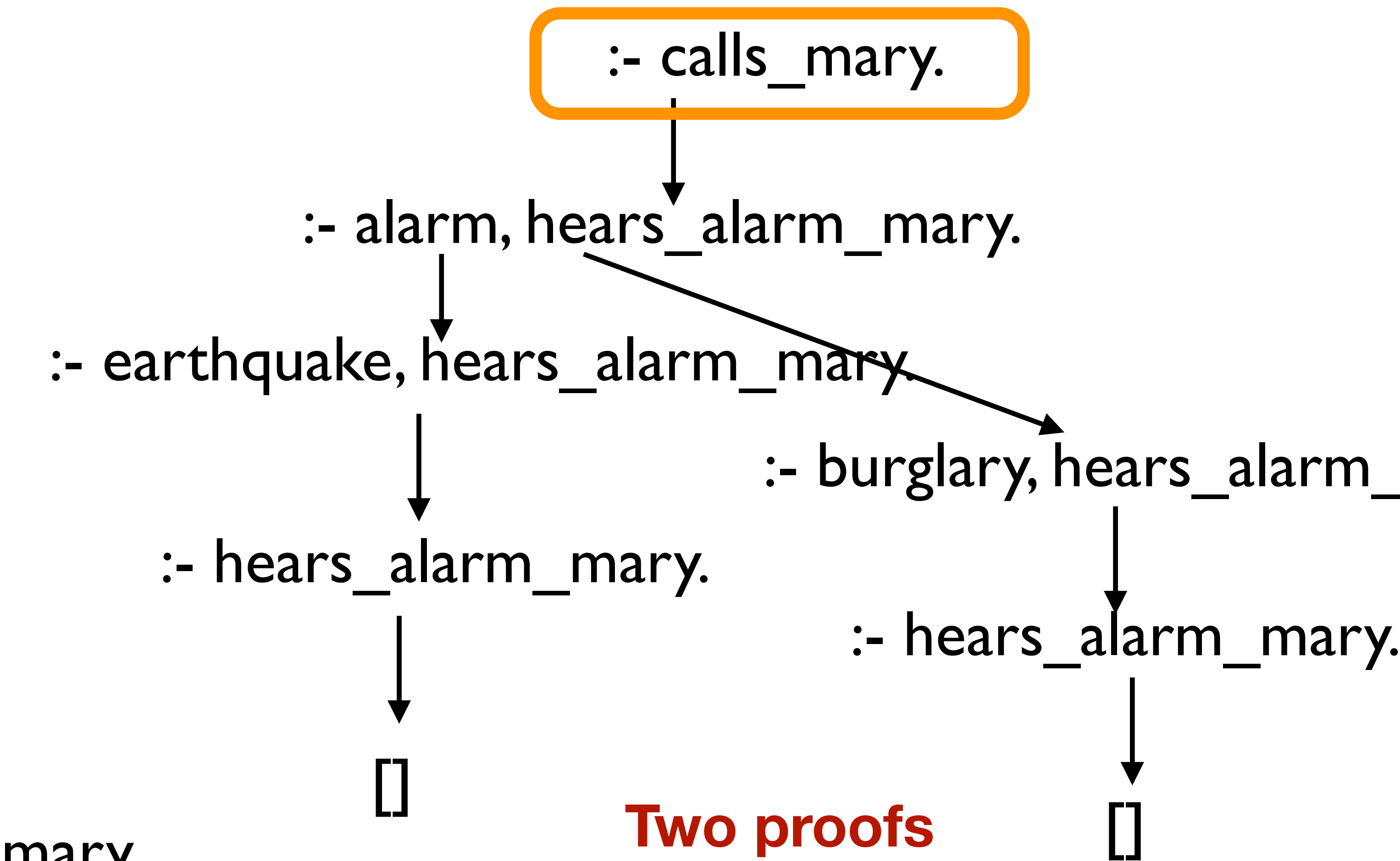
alarm :– earthquake.

alarm :– burglary.

calls_mary :– alarm, hears_alarm_mary.

calls_john :– alarm, hears_alarm_john.

Query



A proof-theoretic view
backward chaining

LOGIC

Logic as constraints

as in SAT solvers

Propositional logic

calls_mary←hears_alarm_mary ∧ alarm

calls_john ← hears_alarm_john ∧ alarm

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

Propositional Logic

```
burglary.
hears_alarm_mary.

earthquake.
hears_alarm_john.

alarm :– earthquake.
alarm :– burglary.

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

LOGIC

Relational/First Order Logic

Introduce Variables and Domains

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}

LOGIC

BOTH for model and proof-based appraoch

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).

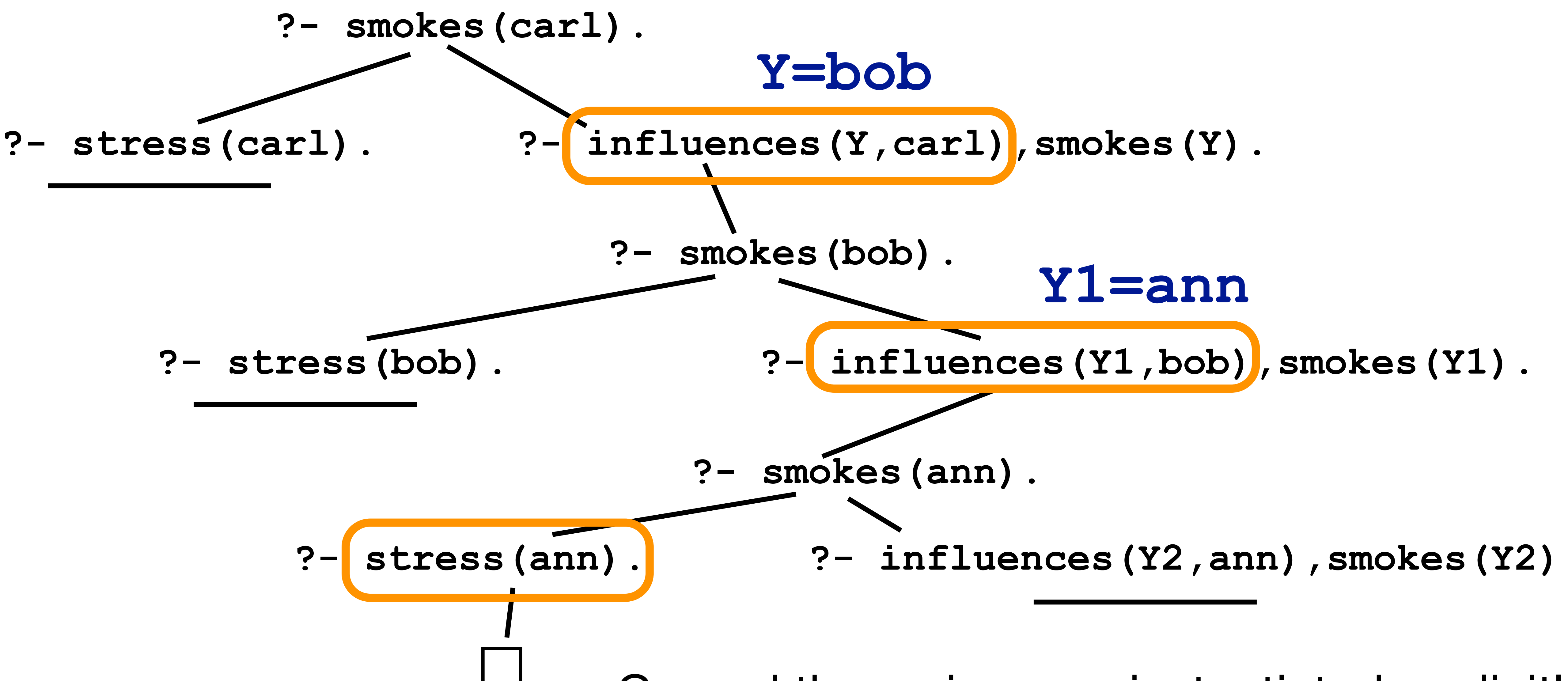
Grounded Theory

BOTH for model and proof-based appraoch

LOGIC

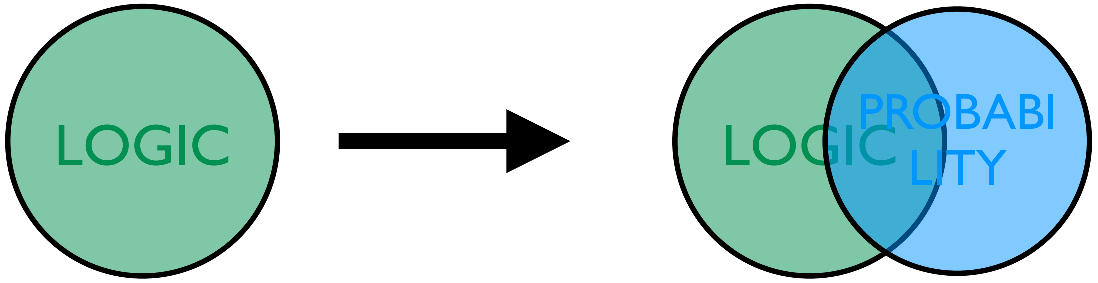
Logical Reasoning Proofs

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

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

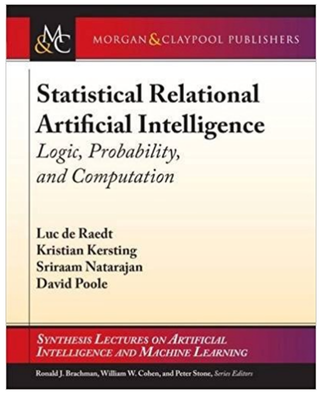


Ground theory is never instantiated explicitly

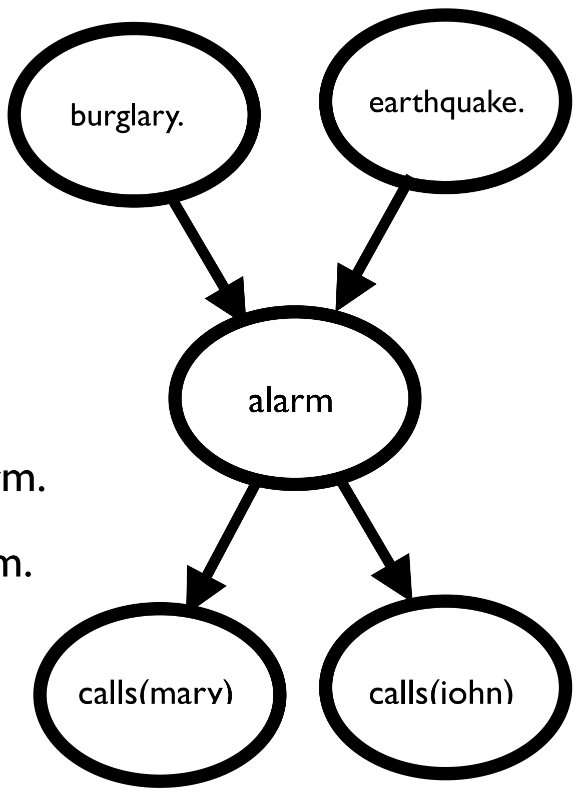
1. Proof vs Model based
2. Directed vs Undirected

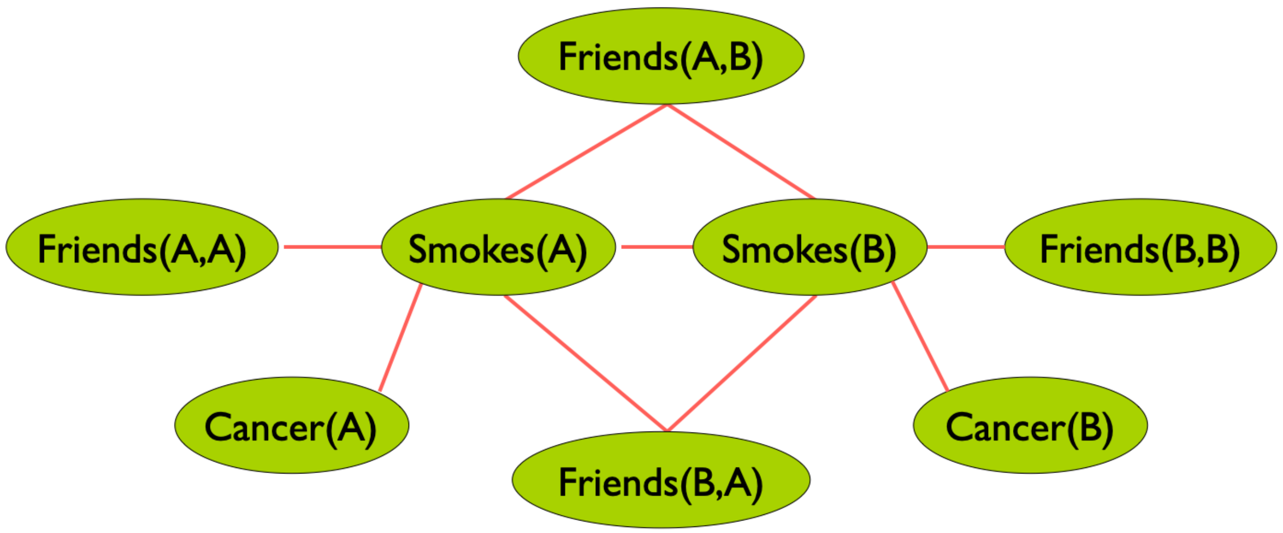


2. Directed vs Undirected the PGM / StarAI dimension




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





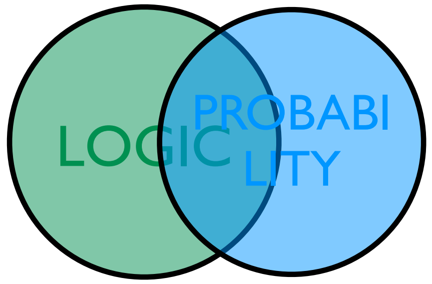
$$1.5 \quad \forall x\ Smokes(x) \Rightarrow Cancer(x)$$
$$1.1 \quad \forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Probabilistic Logic Programs
ProbLog

directed
Bayesian Net

Markov Logic

undirected
Markov Net



key representatives

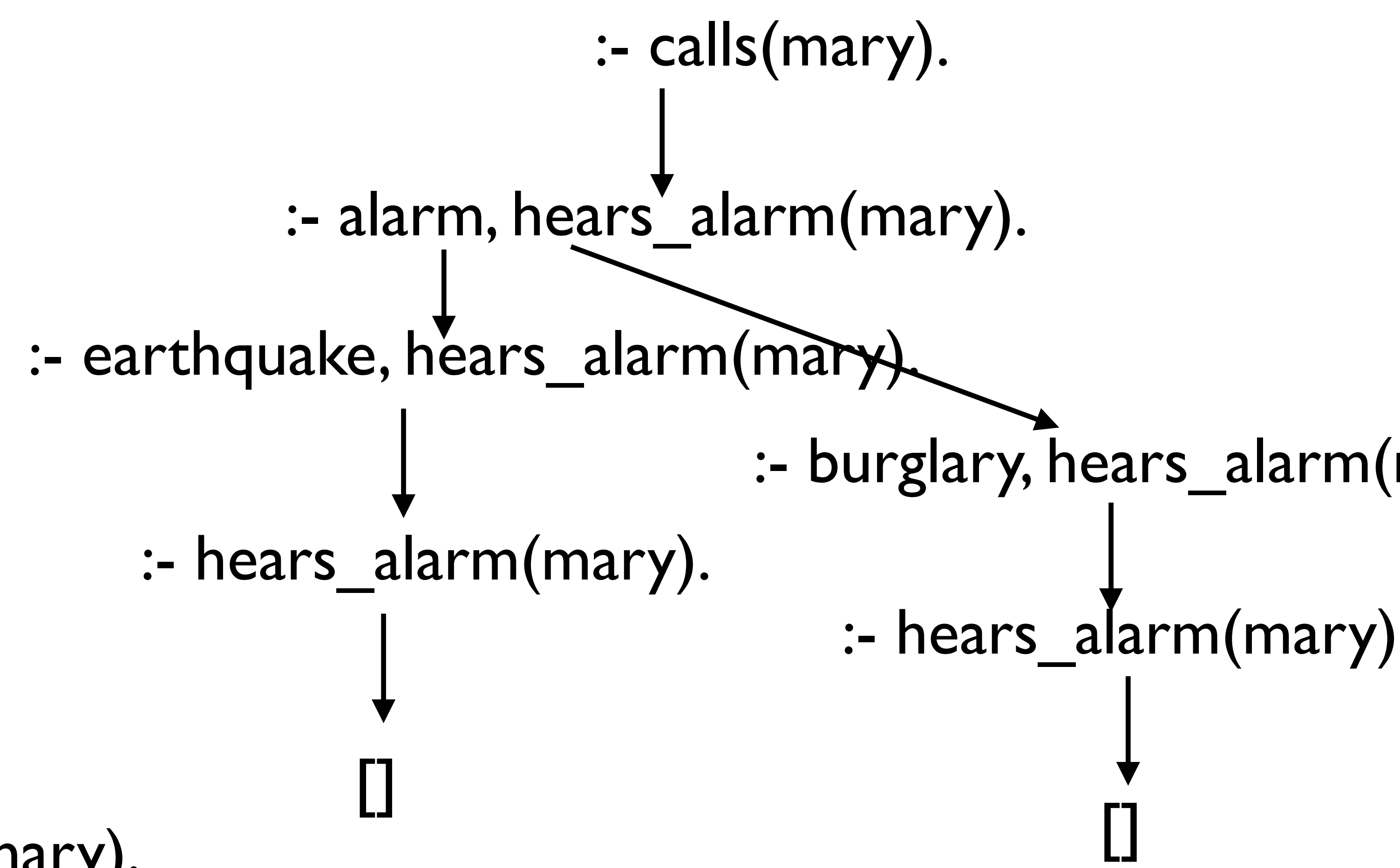
Logic Programs

as in the programming language Prolog

Propositional logic program

```
burglary.
hears_alarm(mary).

earthquake.
hears_alarm(john).

alarm :– earthquake.
alarm :– burglary.

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

Two proofs (by refutation)



A proof-theoretic view

LOGIC

Probabilistic Logic Programs

as in the probabilistic programming language ProbLog

Probabilistic 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).

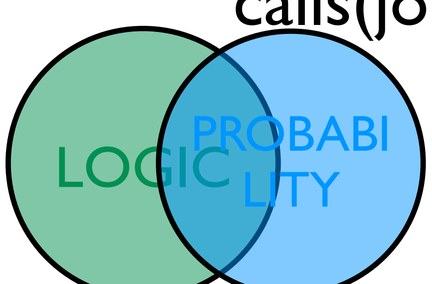
calls(john) :– alarm, hears_alarm(john).

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).

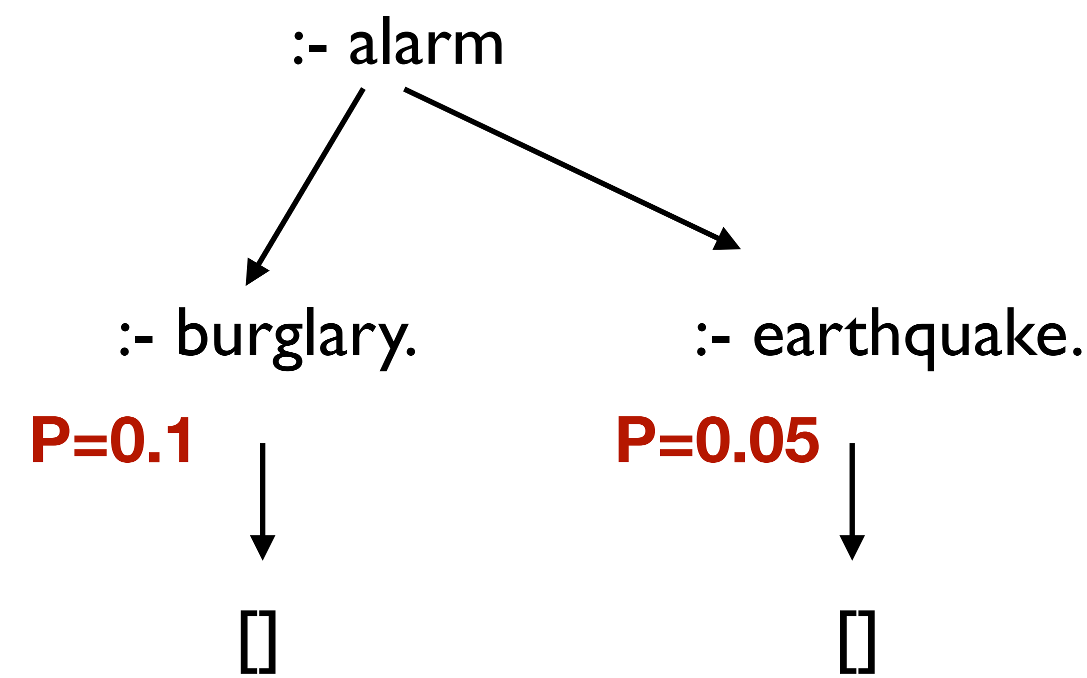
alarm :– earthquake.

alarm :– burglary.

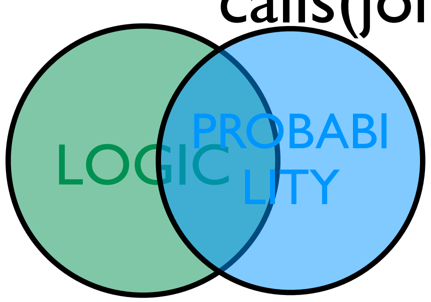
calls(mary) :– alarm, hears_alarm(mary).

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).

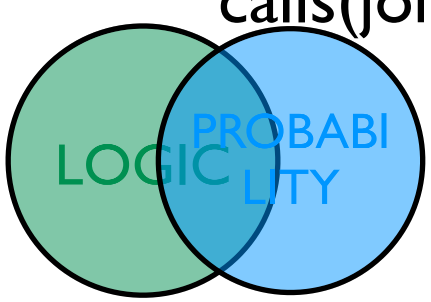
0.05 ::earthquake.
0.6 ::hears_alarm(john).

alarm :– earthquake.

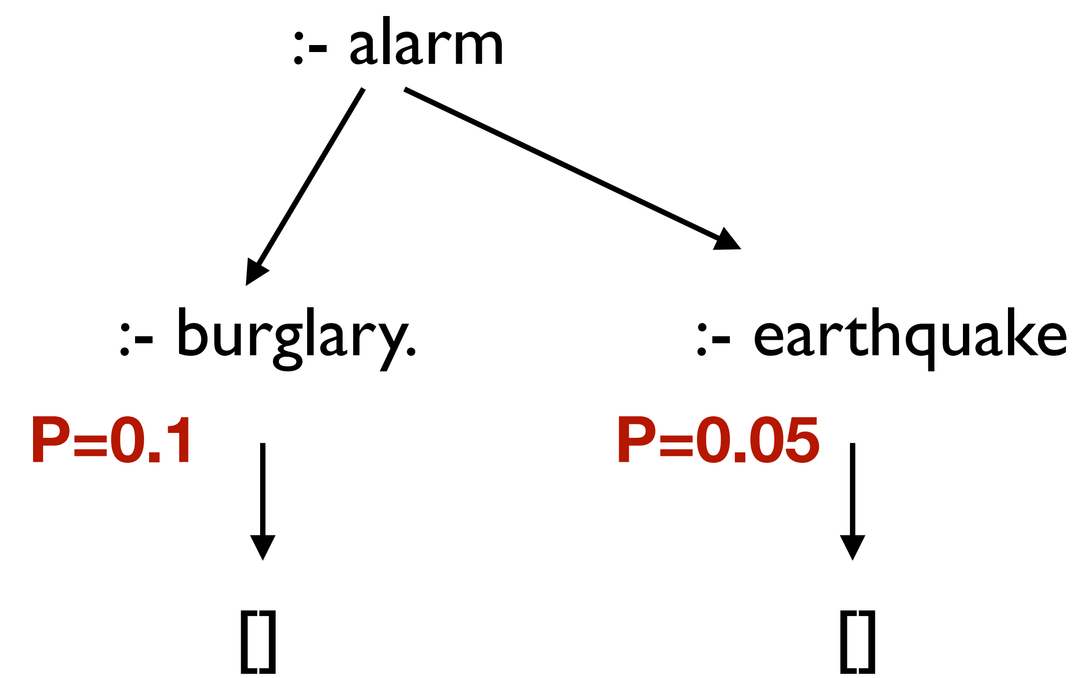
alarm :– burglary.

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

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



Disjoint sum problem



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

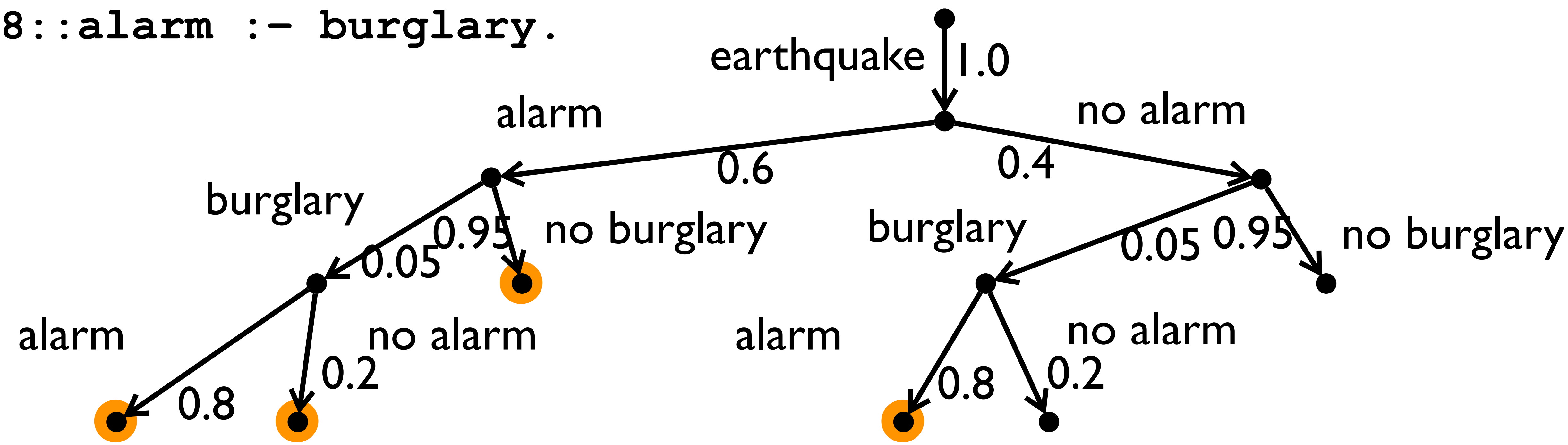
P(alarm) = P(burg OR earth)
= P(burg) + P(earth) - P(burg AND earth)
=/= P(burg) + P(earth)

Probabilistic Logic Program Semantics

[Vennekens et al, ICLP 04]

```
earthquake.
0.05::burglary.

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

probabilistic causal laws



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

Probabilistic Logic Program Semantics

Propositional logic program

0.1 :: burglary.

0.05 :: earthquake.

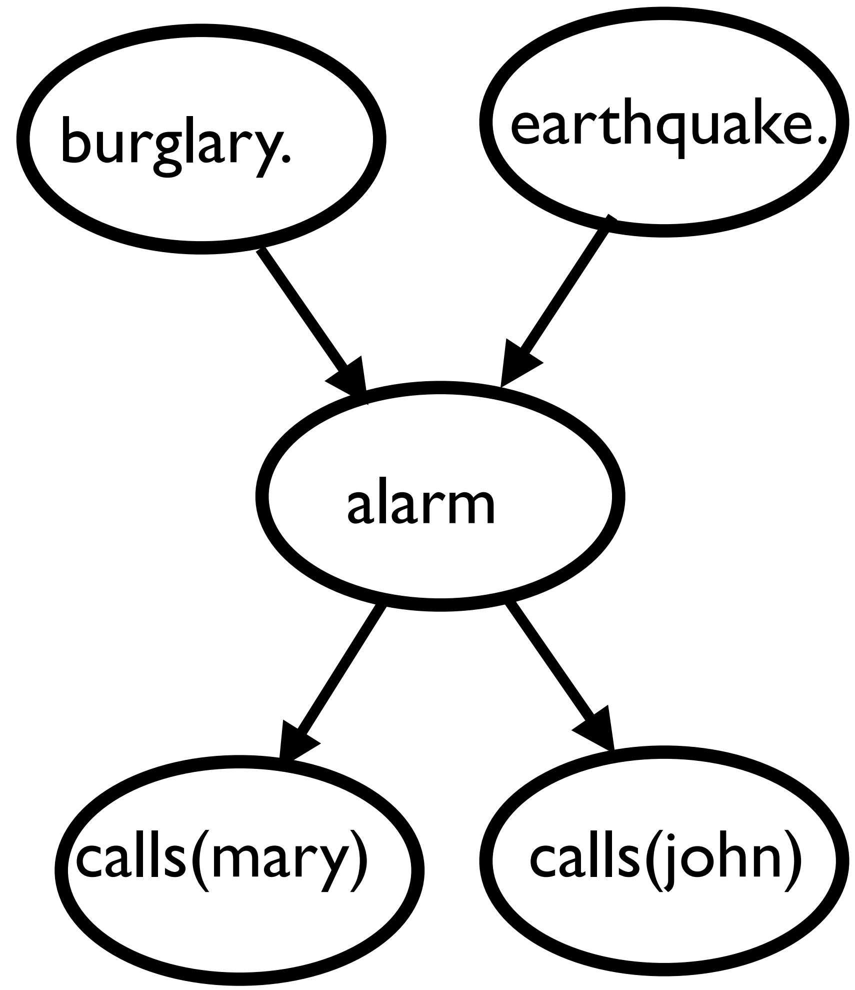
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.

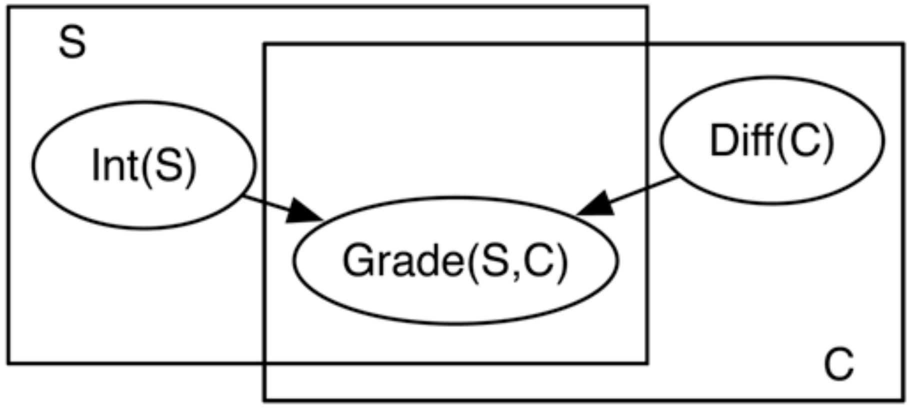
Bayesian Network



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

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

Flexible and Compact Relational Model for Predicting Grades



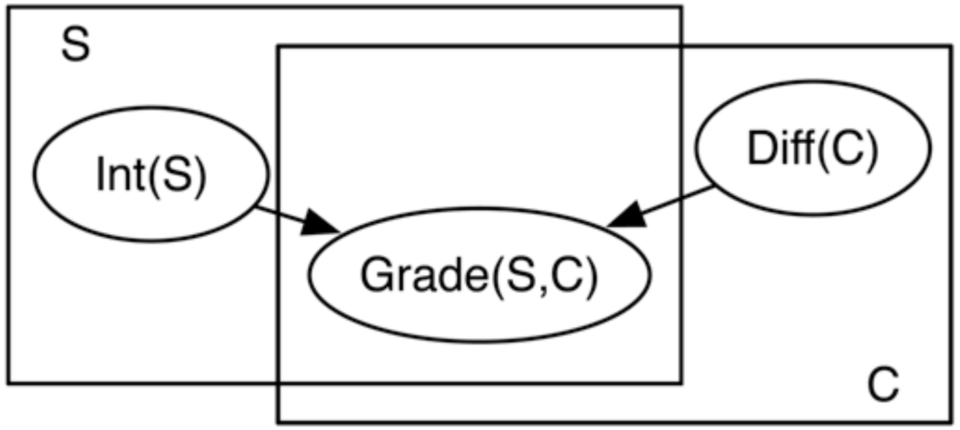
"Program" Abstraction:

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

Grounding:

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

ProbLog by example: Grading

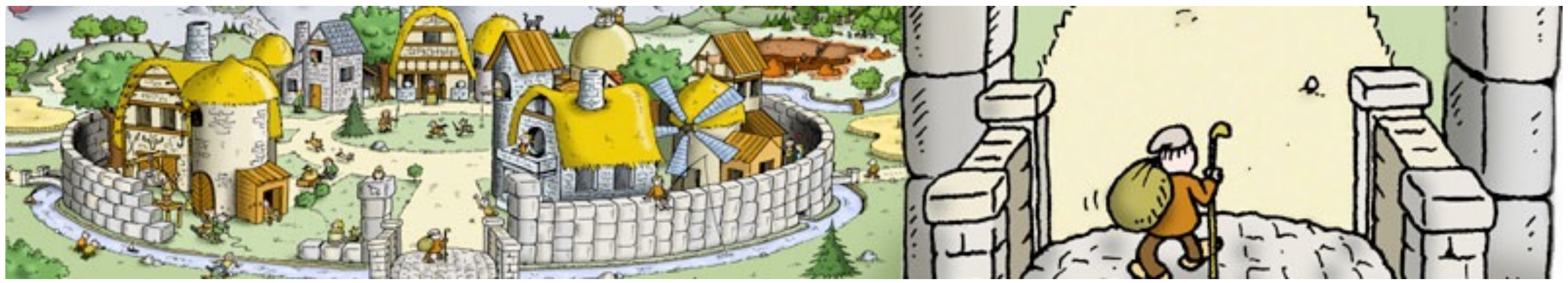



```
0.4 :: int(S) :- student(S).
0.5 :: diff(C):- course(C).

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

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

ProbLog by example: Grading

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

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

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

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

Dynamic networks



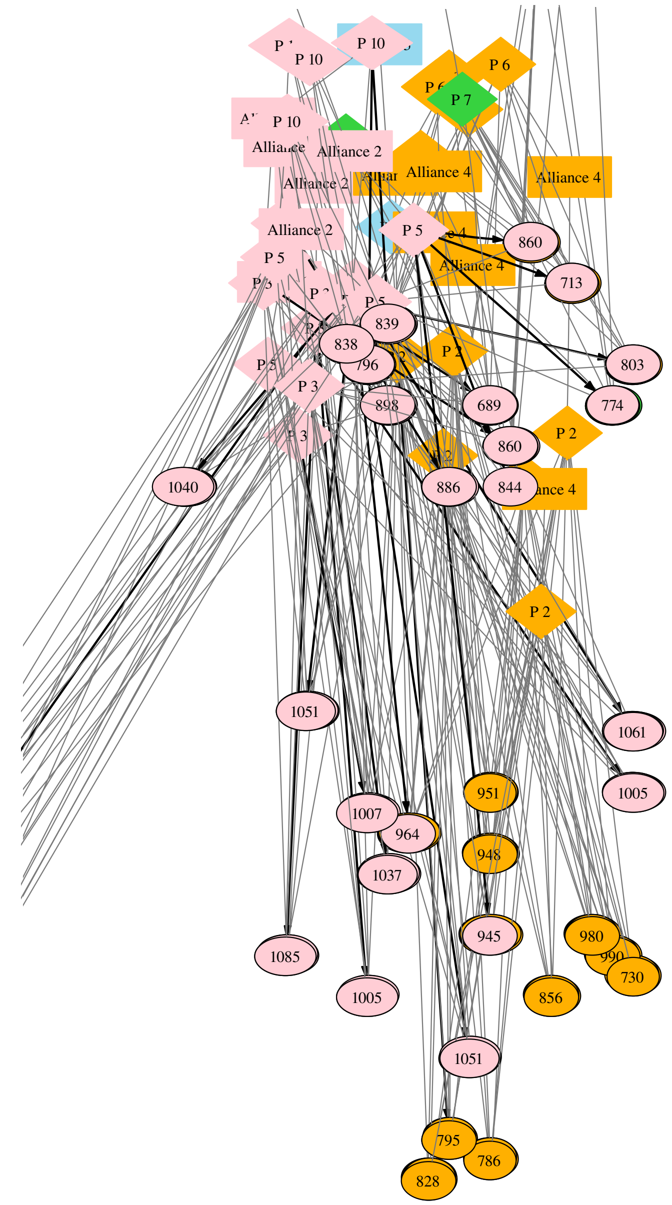
Travian: A massively multiplayer real-time strategy game

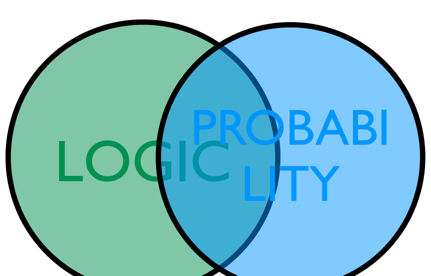
Can we build a model

of this world ?

Can we use it for playing

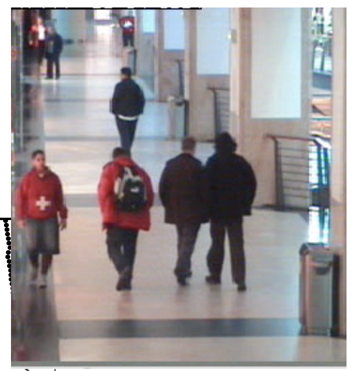
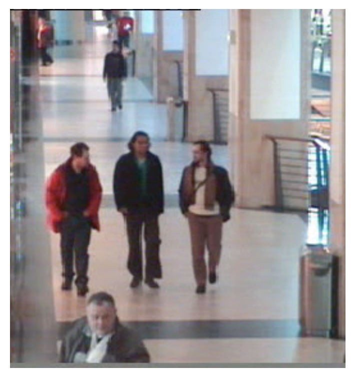
better ?

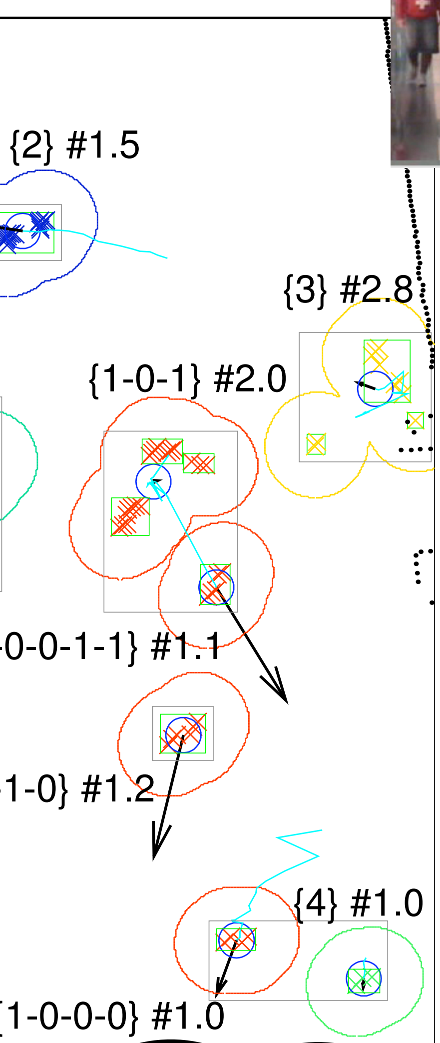




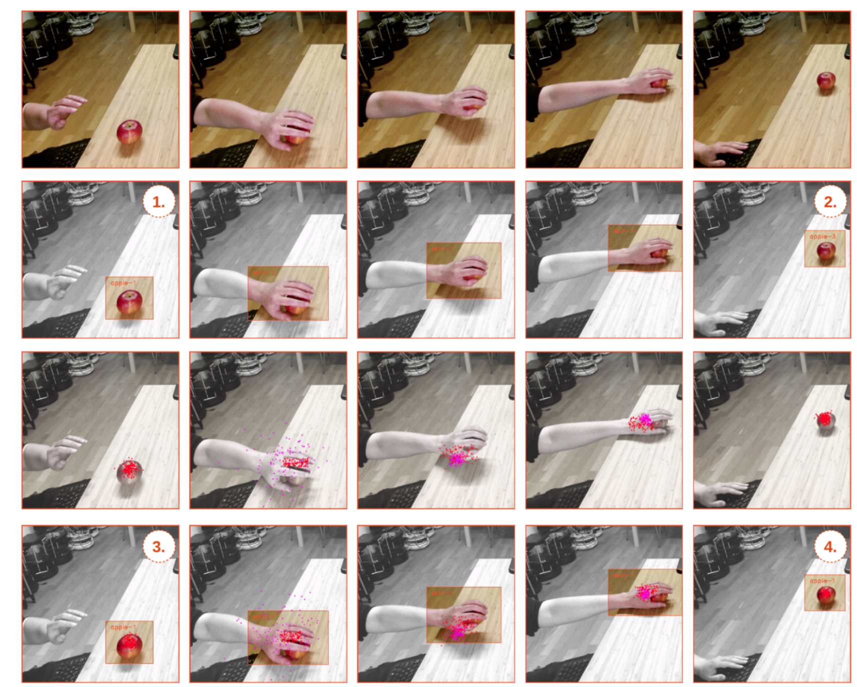
[Thon et al, MLJ 11]

Activity analysis and tracking video analysis

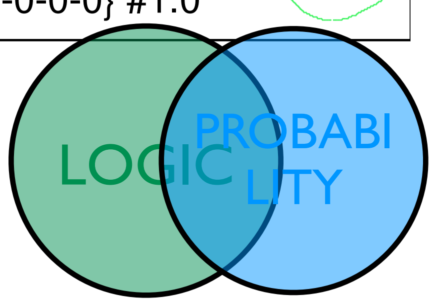



- Track people or objects over time? Even if temporarily hidden?
- Recognize activities?
- Infer object properties?

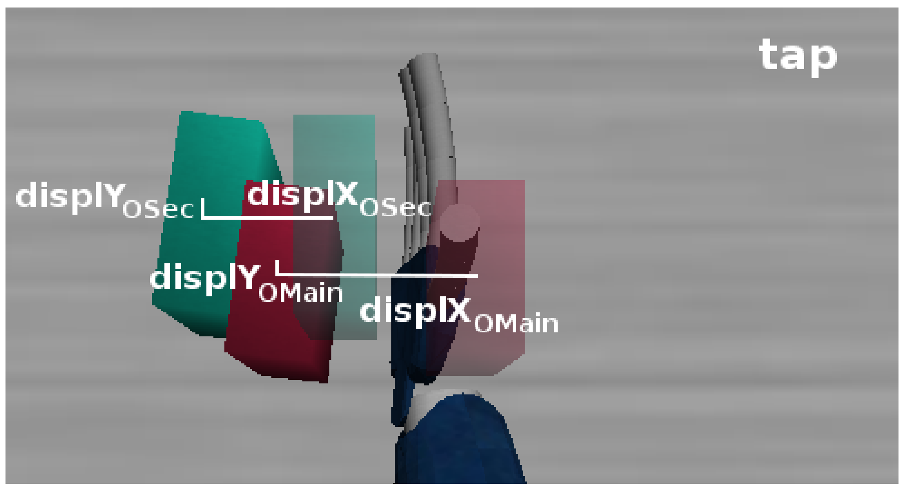


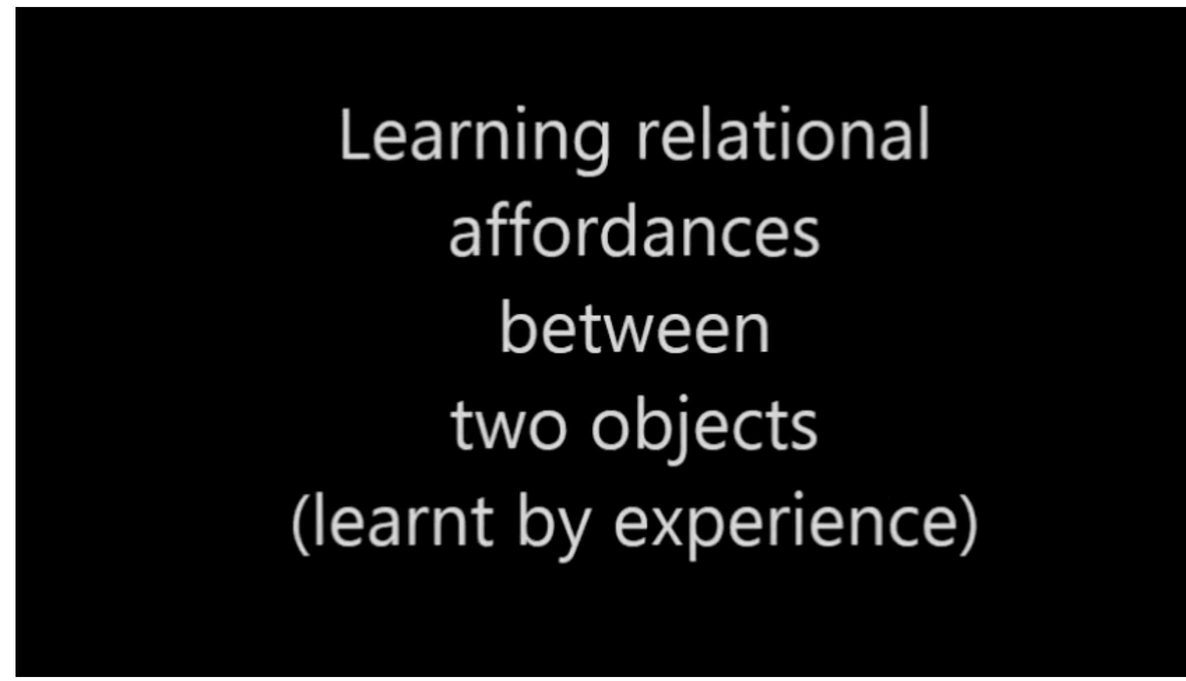
[Skarlatidis et al, TPLP 14; Nitti et al, IROS 13, ICRA 14, MLJ 16]

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



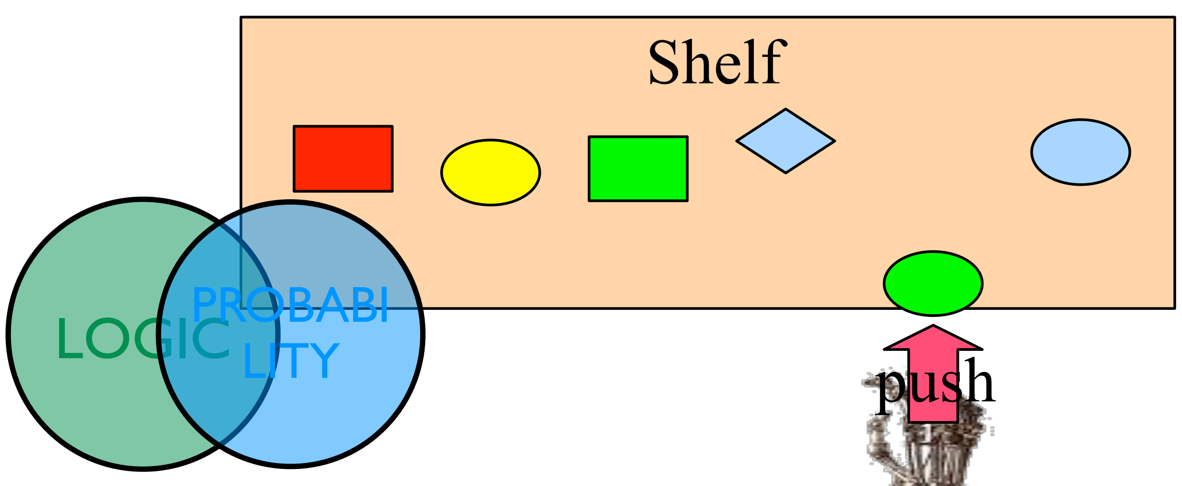
Learning relational affordances

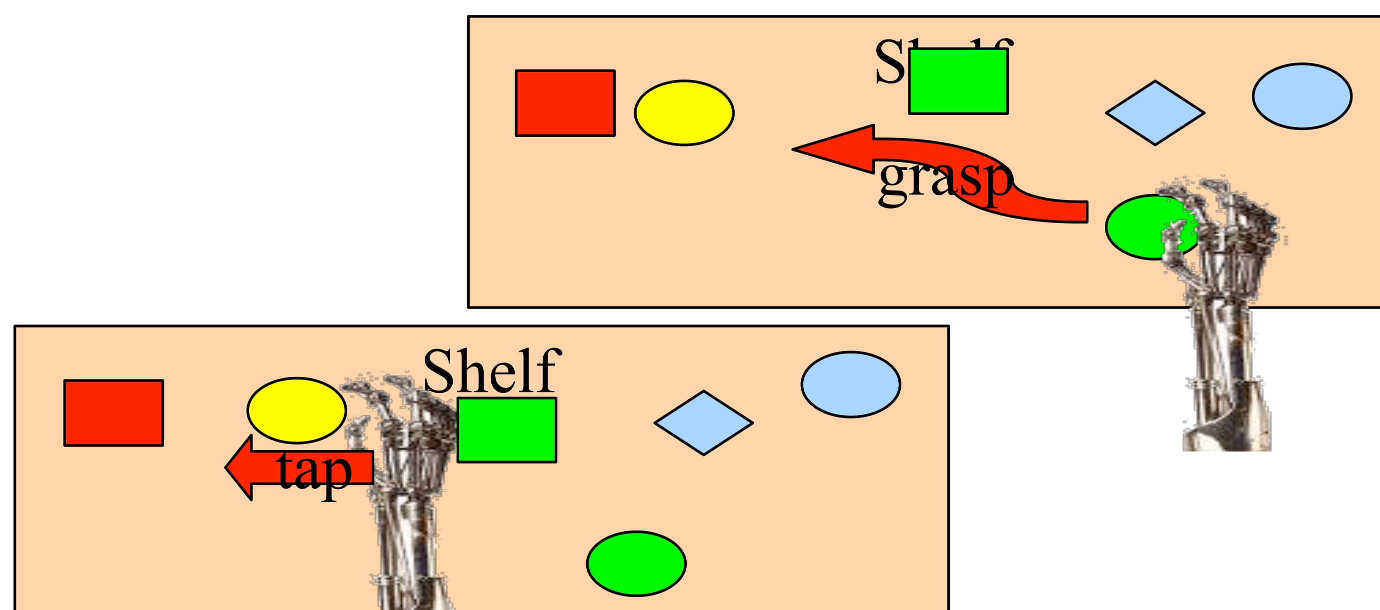




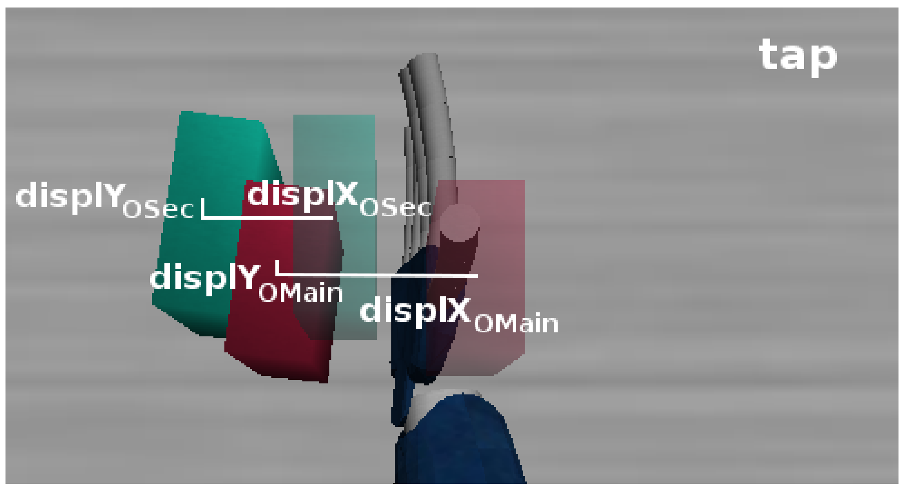
1), an **similar to probabilistic Strips (with continuous distributions)**

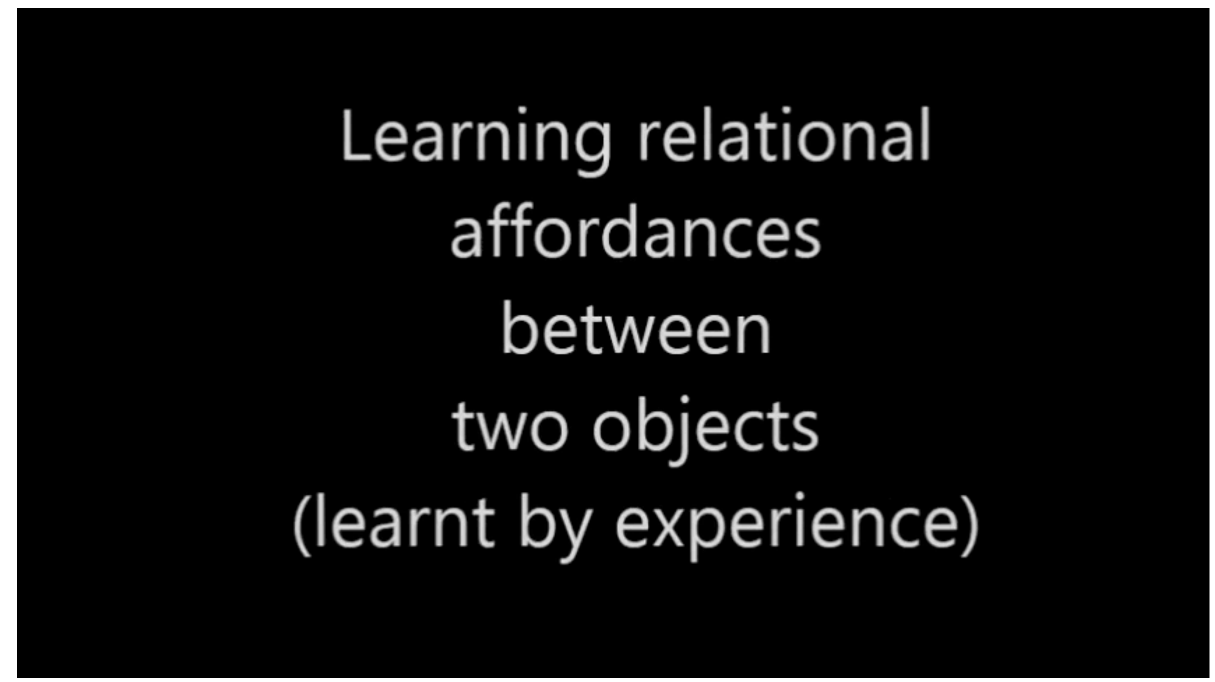
Moldovan et al. ICRA 12, 13, 14; Auton. Robots 18





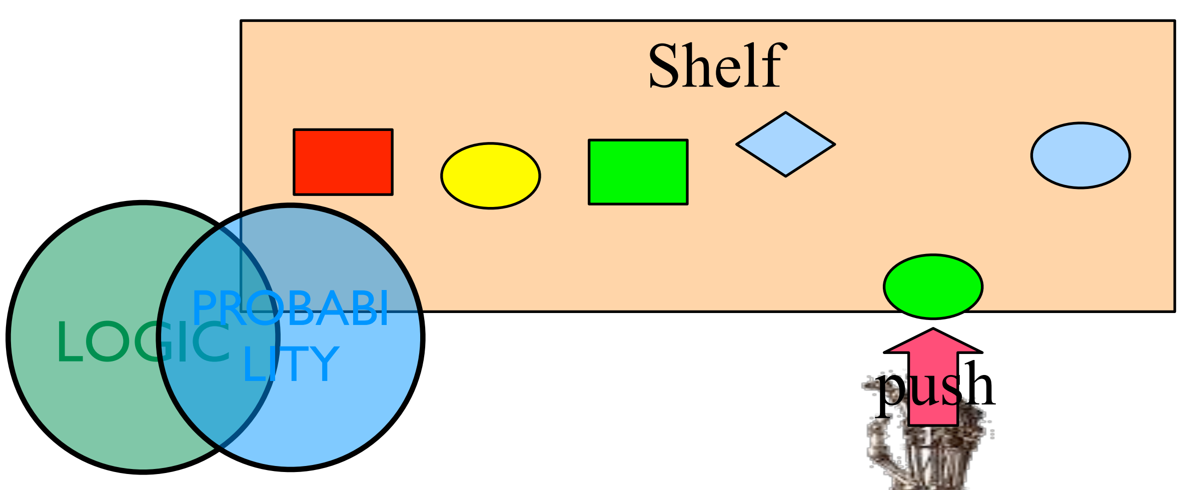
Learning relational affordances

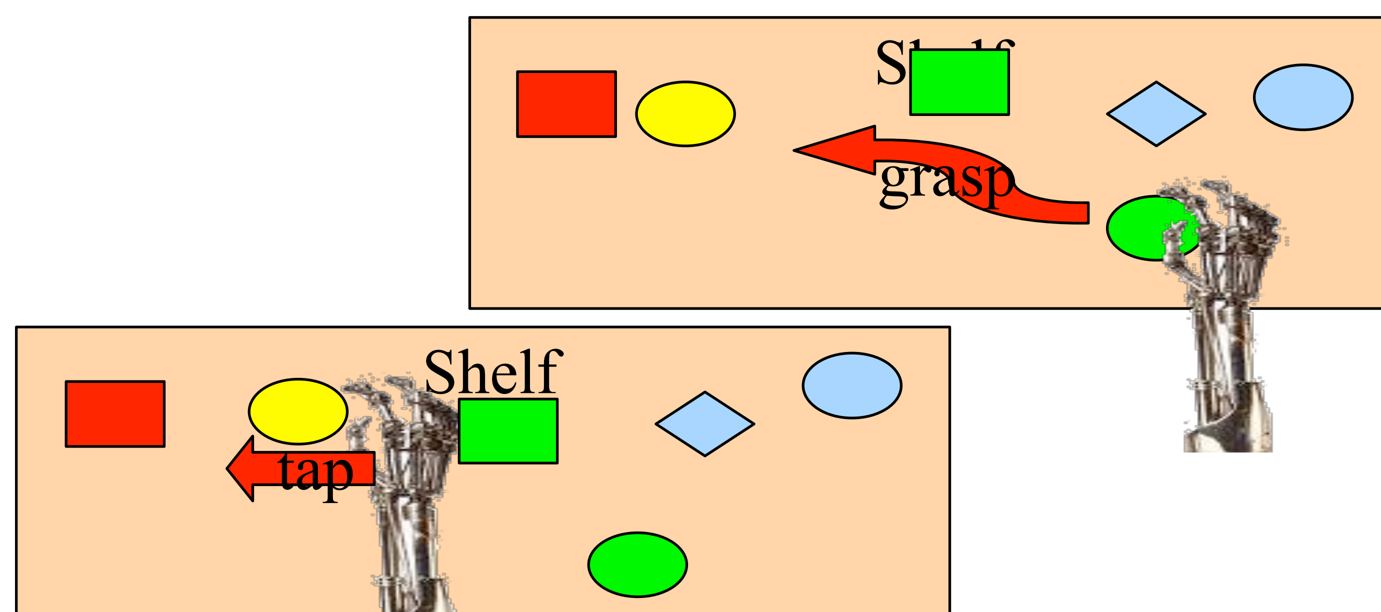




1), an **similar to probabilistic Strips (with continuous distributions)**

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





Biology

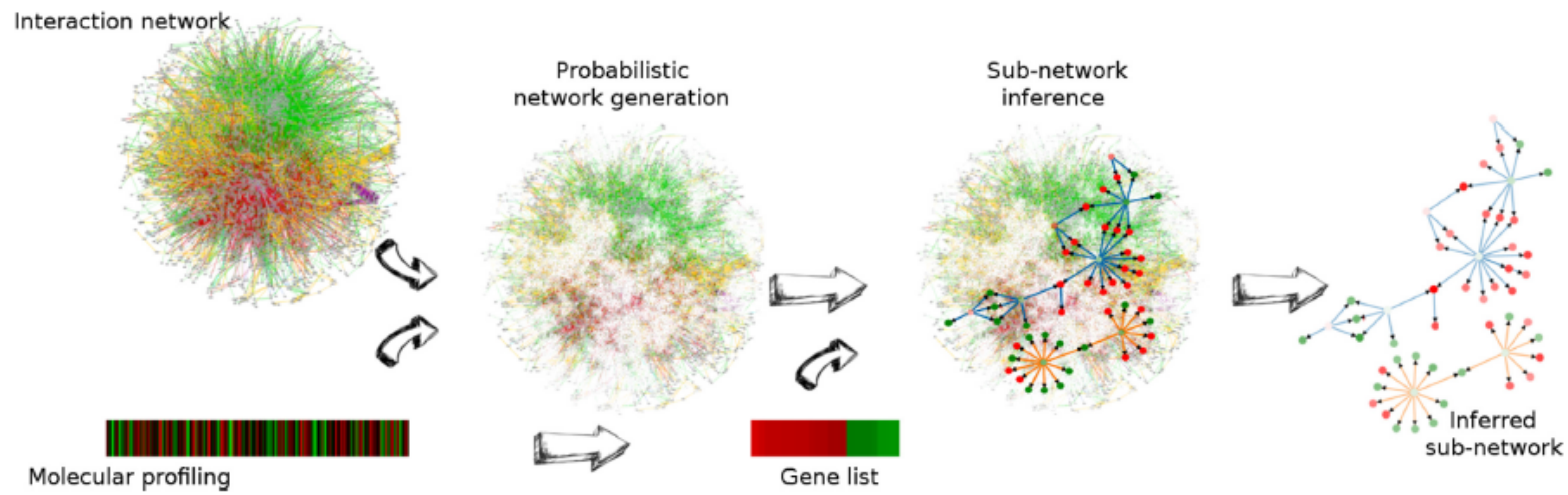


Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for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

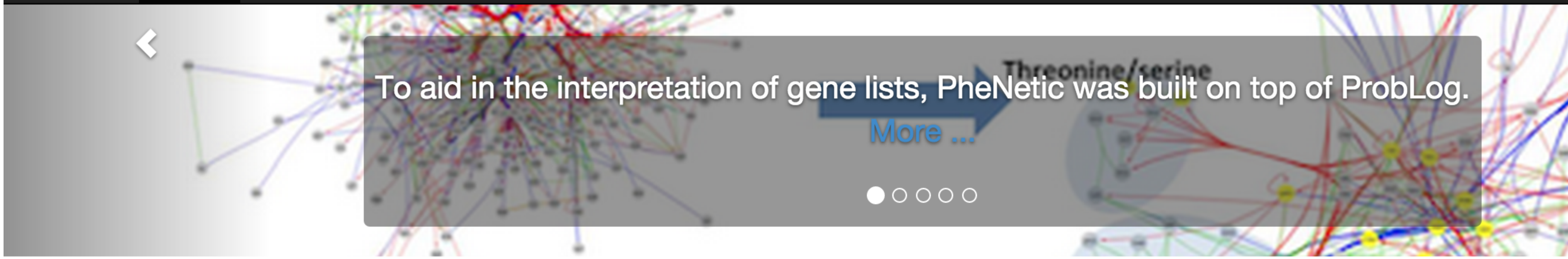
- Causes: Mutations
  - All related to similar phenotype
- Effects: Differentially expressed genes
- 27 000 cause effect pairs

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

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

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

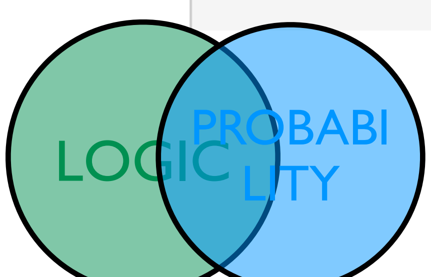
ProbLog is a tool that allows you to intuitively build programs that do not only encode **complex interactions** between a large sets of **heterogenous components** bu **uncertainties** that are present in real-life situations.

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).
```



Constraints

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

What about constraints? Do they have a probabilistic interpretation?

Markov Logic: Intuition

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

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

A possible worlds view

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

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

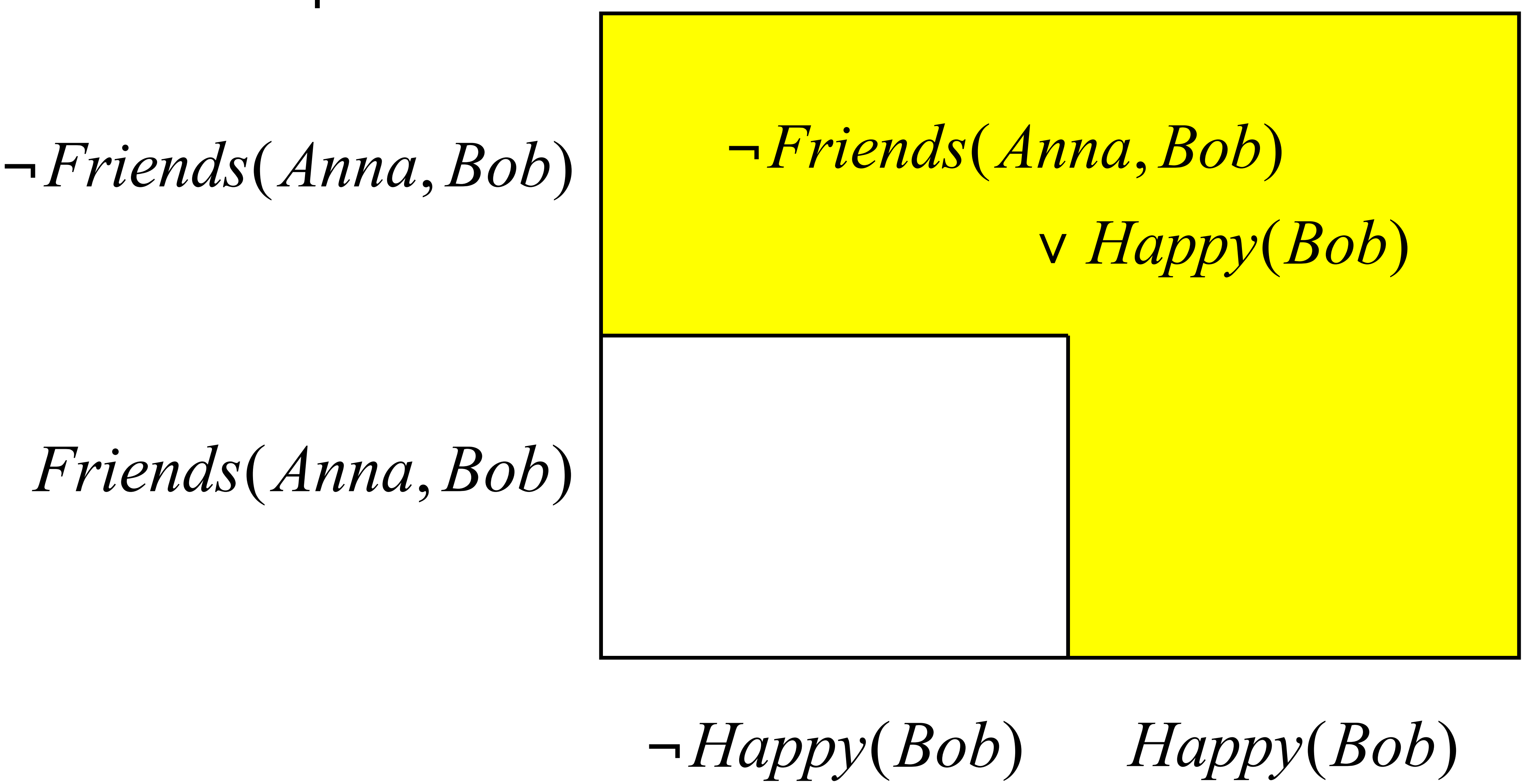
slides by Pedro Domingos

A possible worlds view

Logical formulas such as

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

exclude possible worlds



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

A possible worlds view

Or as log-linear model this is:

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

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

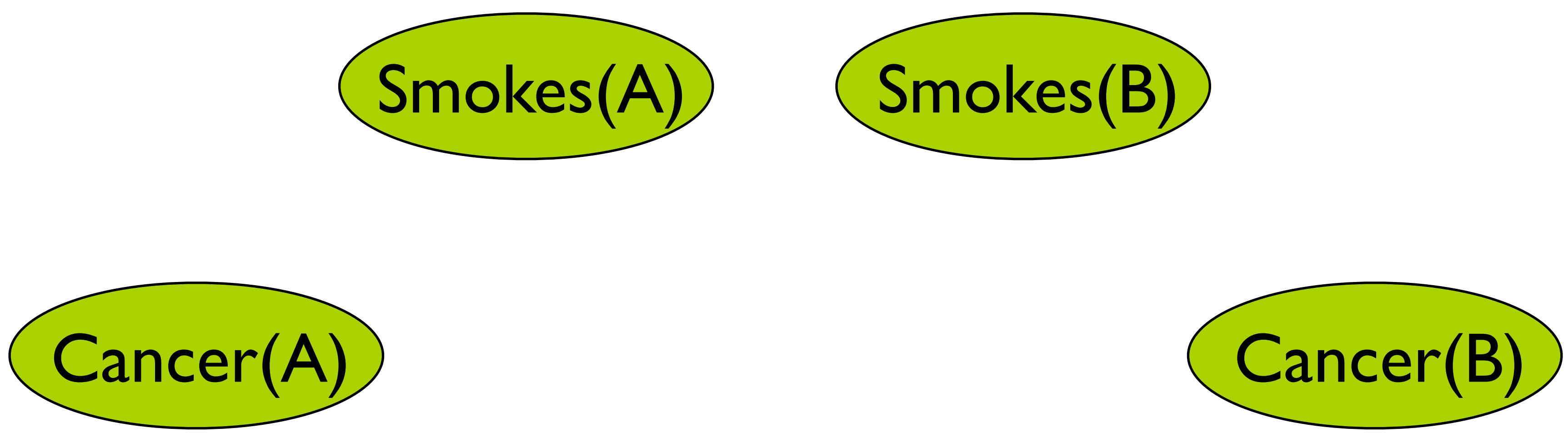
	$\neg Happy(Bob)$	$Happy(Bob)$
$\neg Friends(Anna, Bob)$	1	1
$Friends(Anna, Bob)$	0.75	1

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)

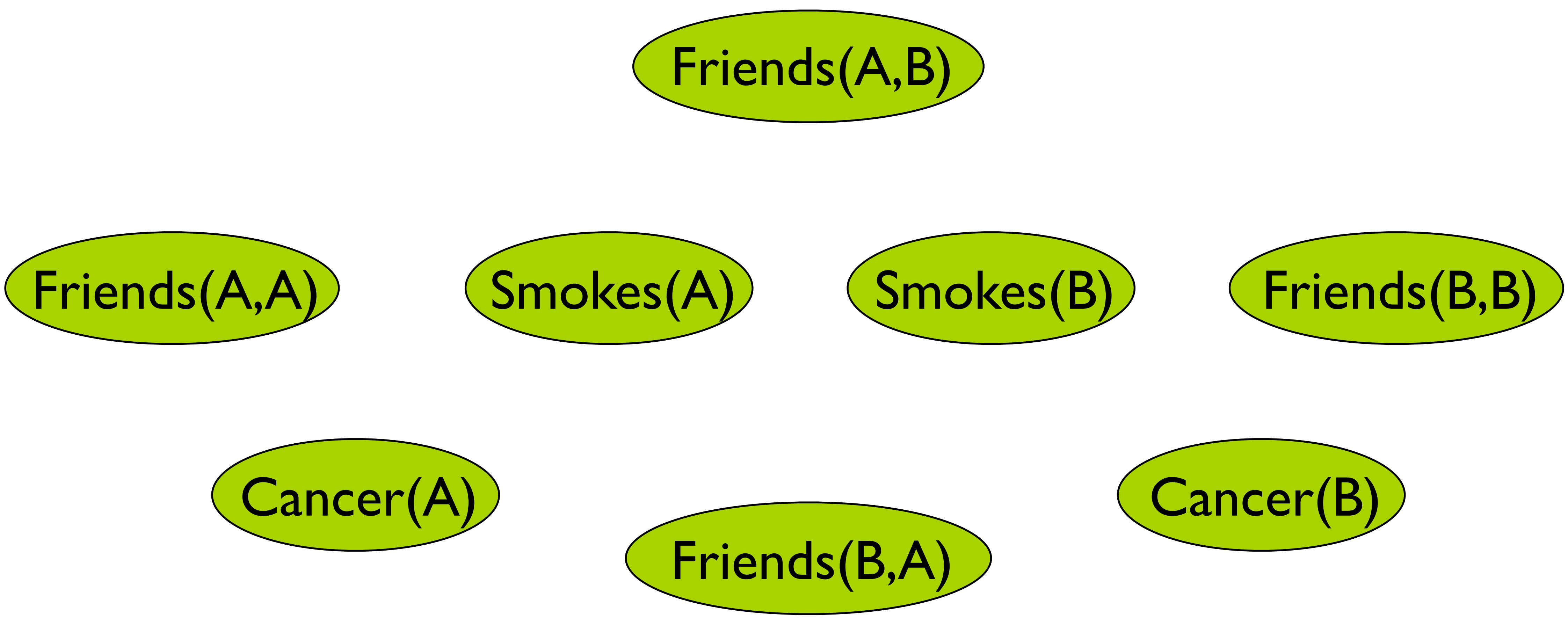


slides by Pedro Domingos

Markov Logic

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

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

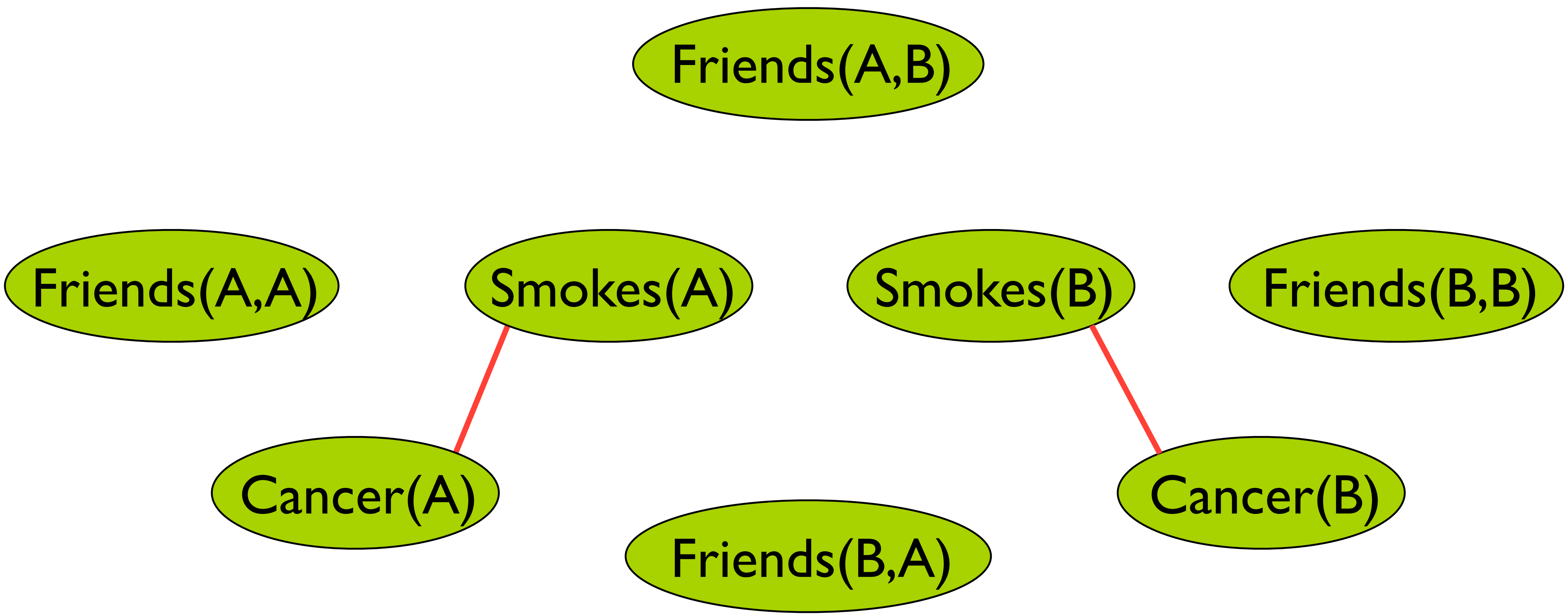


slides by Pedro Domingos

Markov Logic

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

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

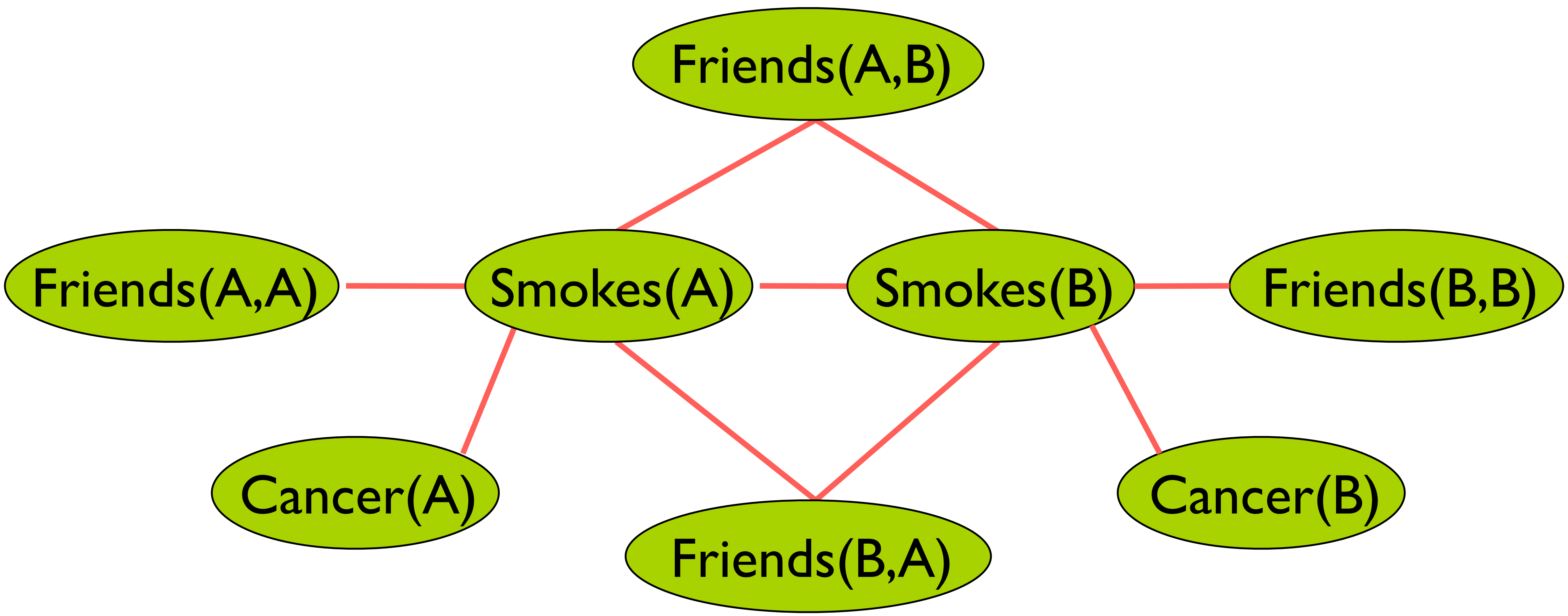


slides by Pedro Domingos

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

Applications

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

Alchemy: Open Source AI

Tutorial

Mailing Lists

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

Repositories

- Code
- Datasets
- MLNs
- Publications

Related Links

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

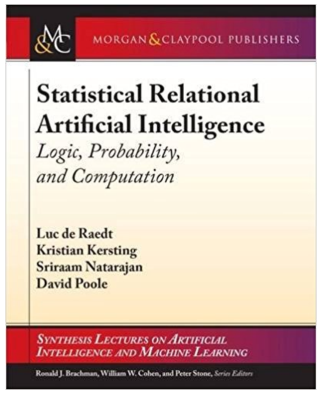
- Collective classification
- Link prediction
- Entity resolution
- Social network modeling
- Information extraction

Choose a version of Alchemy:

Alchemy Lite

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

2. Directed vs Undirected the PGM / StarAI dimension



0.1 :: burglary.

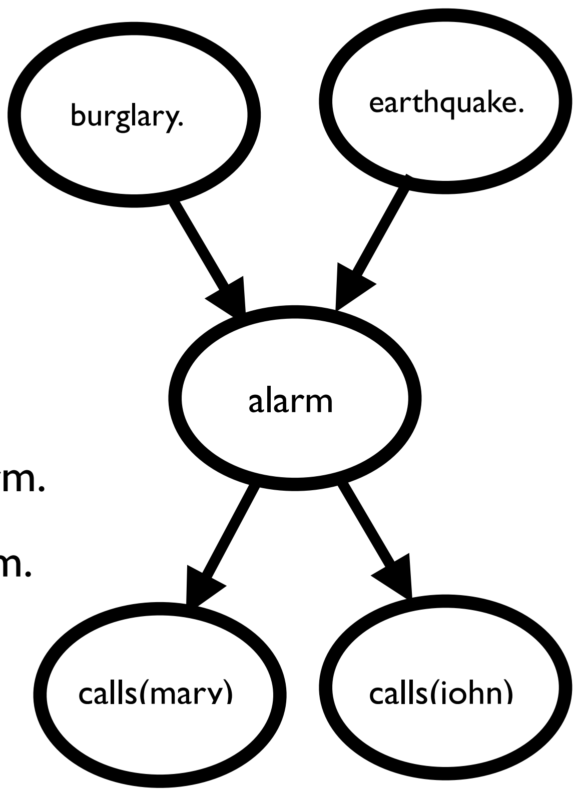
0.05 :: earthquake.

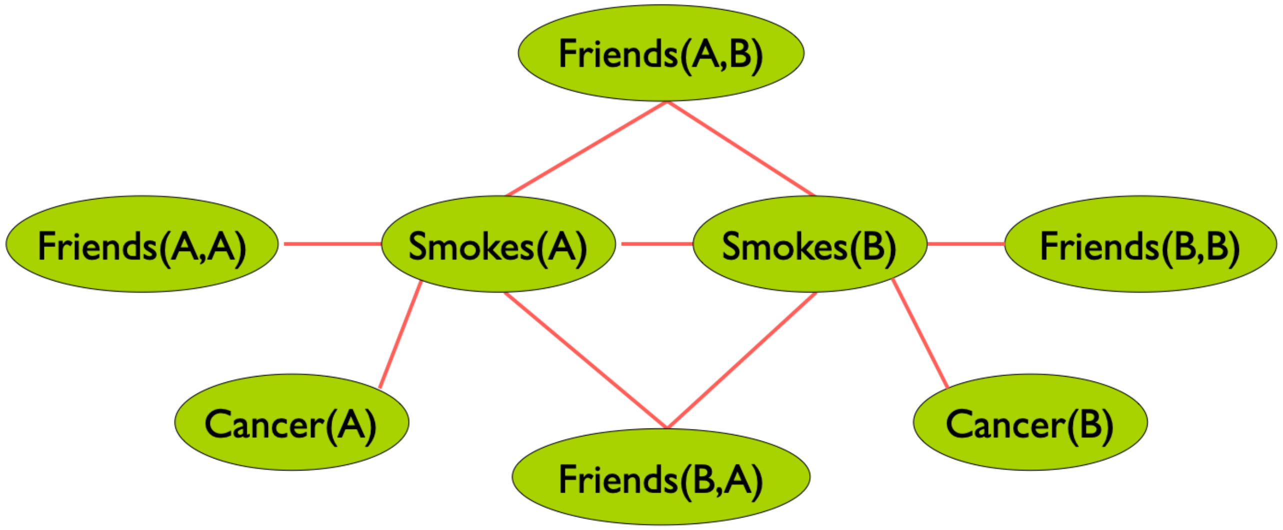
alarm :– earthquake.

alarm :– burglary.

0.7::calls(mary) :– alarm.

0.6::calls(john) :– alarm.

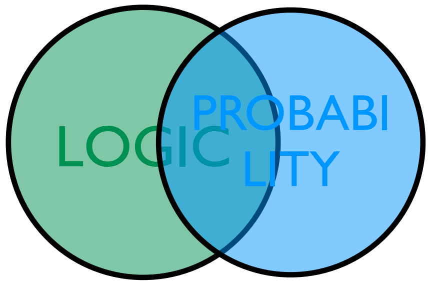




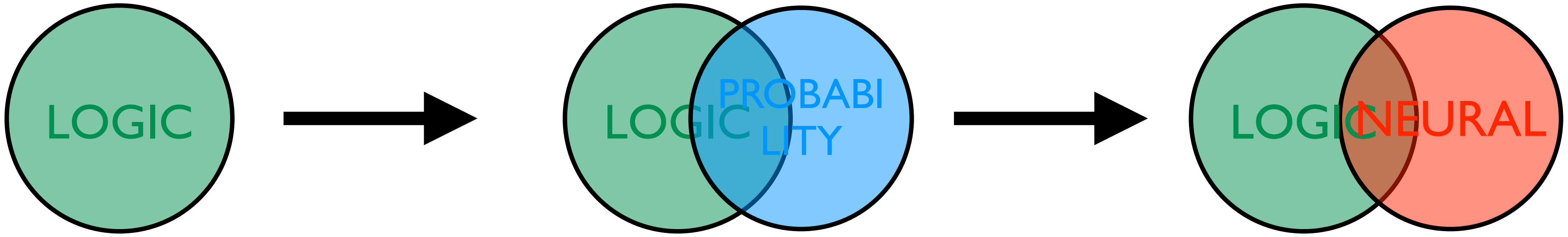
$$1.5 \quad \forall x \; Smokes(x) \Rightarrow Cancer(x)$$

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

Logic is used as a template for a probabilistic graphical model: knowledge based model construction KBMC



1. Proof vs Model based
2. Directed vs Undirected



2. Directed vs Undirected the NeSy dimension

Two types of Neural Symbolic Systems

Logic as a ***neural program***

Directed StarAI approach and logic programs

Logic as a ***regularizer***

undirected StarAI approach and (soft) constraints

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

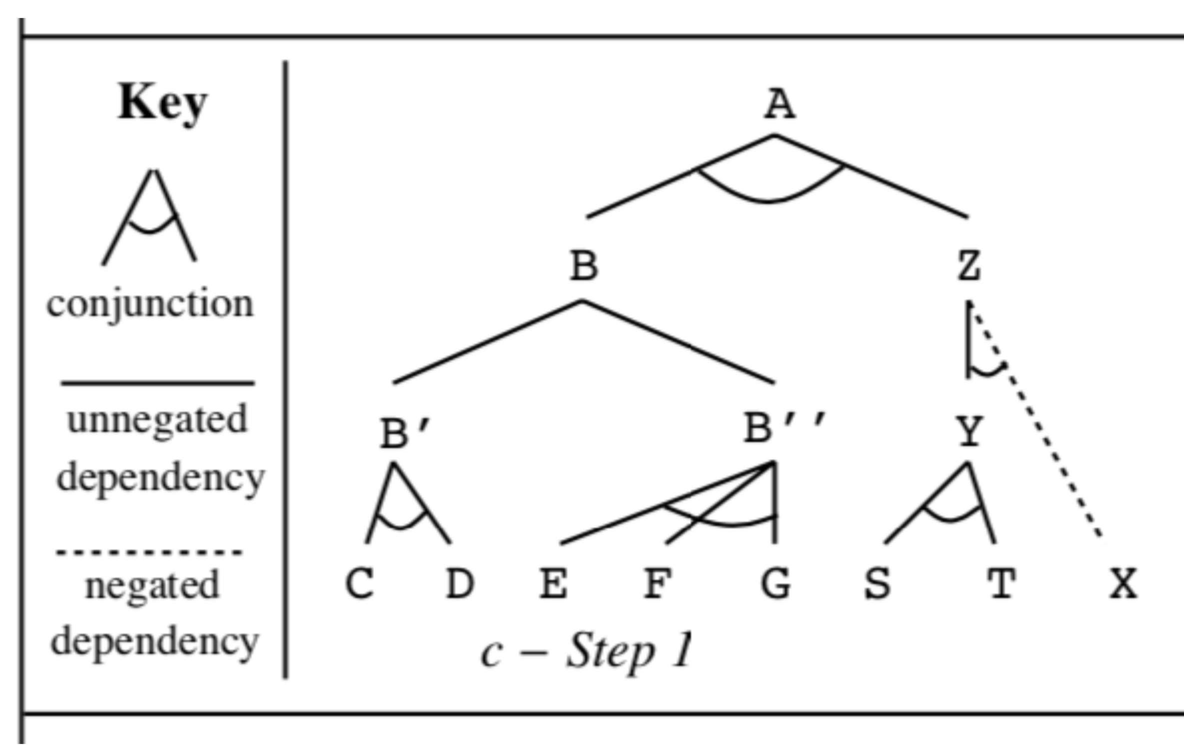
- KBANN (Towell and Shavlik AIJ 94)
- Turn a (propositional) Prolog program into a neural network and learn

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

REWRITE →

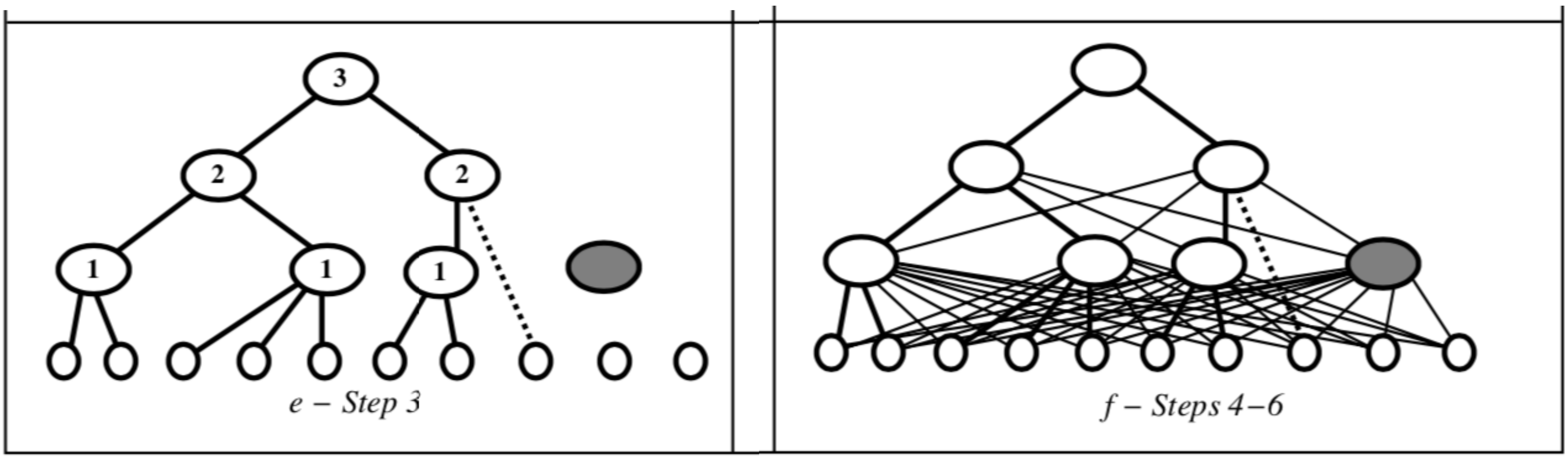
```
A   :- B, Z.
B   :- B'.
B   :- B''.
B'  :- C, D.
B'' :- E, F, G.
Z   :- Y, not X.
Y   :- S, T.
```

→



Logic as a neural program

directed StarAI approach and logic programs



ADD LINKS — ALSO SPURIOUS ONES

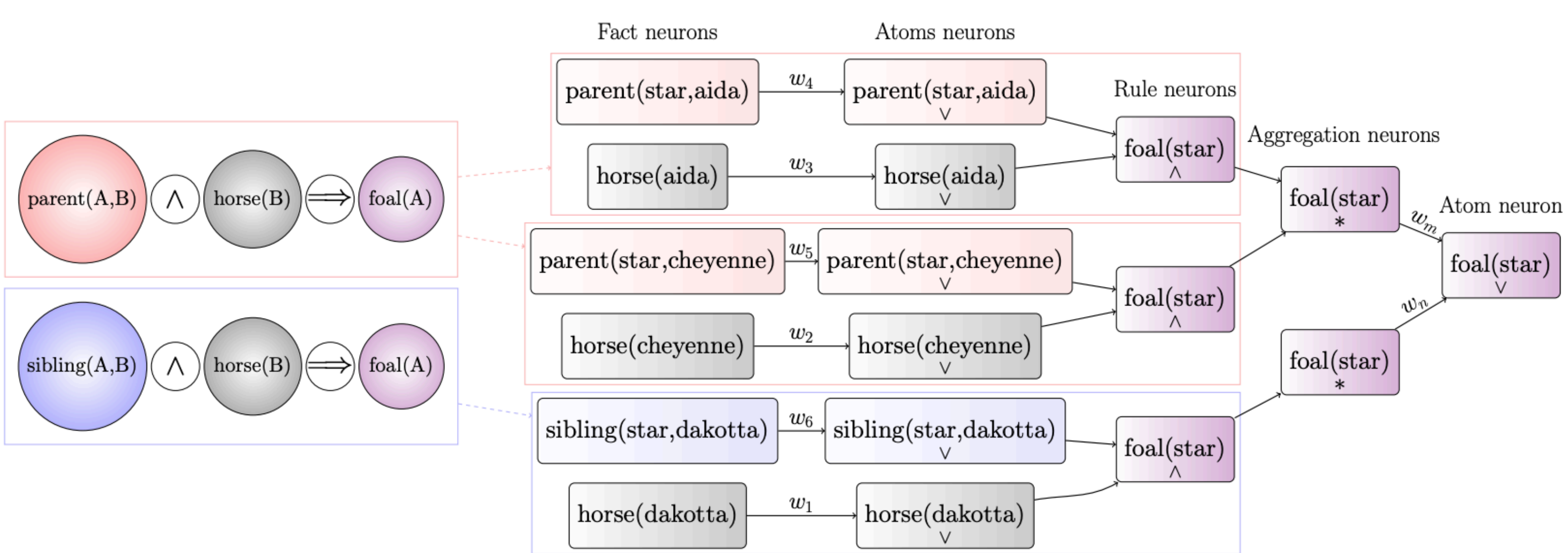
HIDDEN UNIT

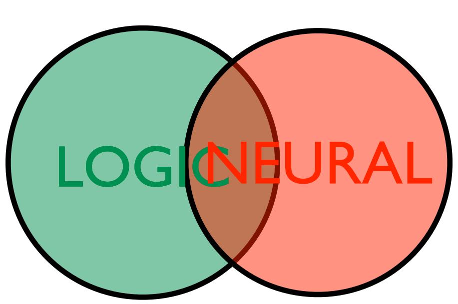
and then learn
(Details of activation & loss functions not mentioned)

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)





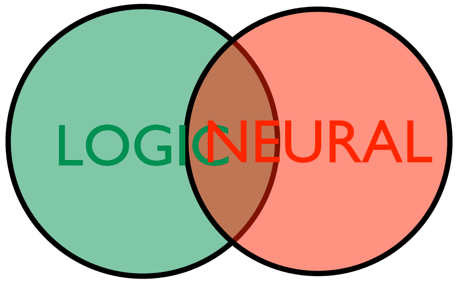
[Sourek, Kuzelka, et al JAIR]

Neural Theorem Prover

directed StarAI approach and logic programs

```
father(Omer,Bart).
father(Abe,Omer).
parent(X,Y) :- father(X,Y).
grandFather(X,Y) :- father(X,Z),
                    parent(Z,Y)
```

```
:- grandFather(Abe,Bart)
        |
:- father(Abe,Z), parent(Z,Bart)
        |  Z=Omer
:- father(Abe,Omer), parent(Omer,Bart)
        |
:- parent(Omer,Bart)
        |
:- father(Omer,Bart)
        |
:- [ ]
```

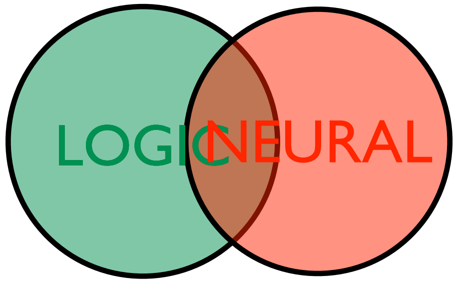


Neural Theorem Prover

directed StarAI approach and logic programs

```
father(Omer,Bart).
father(Abe,Omer).
parent(X,Y) :- father(X,Y).
grandFather(X,Y) :- father(X,Z),
                    parent(Z,Y)
```

```
:- grandPa(Abe,Bart)
         |
       ?????
```



Neural Theorem Prover

directed StarAI approach and logic programs

```
father(Omer,Bart).
father(Abe,Omer).
parent(X,Y) :- father(X,Y).
grandFather(X,Y) :- father(X,Z),
                    parent(Z,Y)
```

:- grandPa(Abe,Bart)

w ~ distance(grandPa, grandFather)

:- father(Abe,Z), parent(Z,Bart)

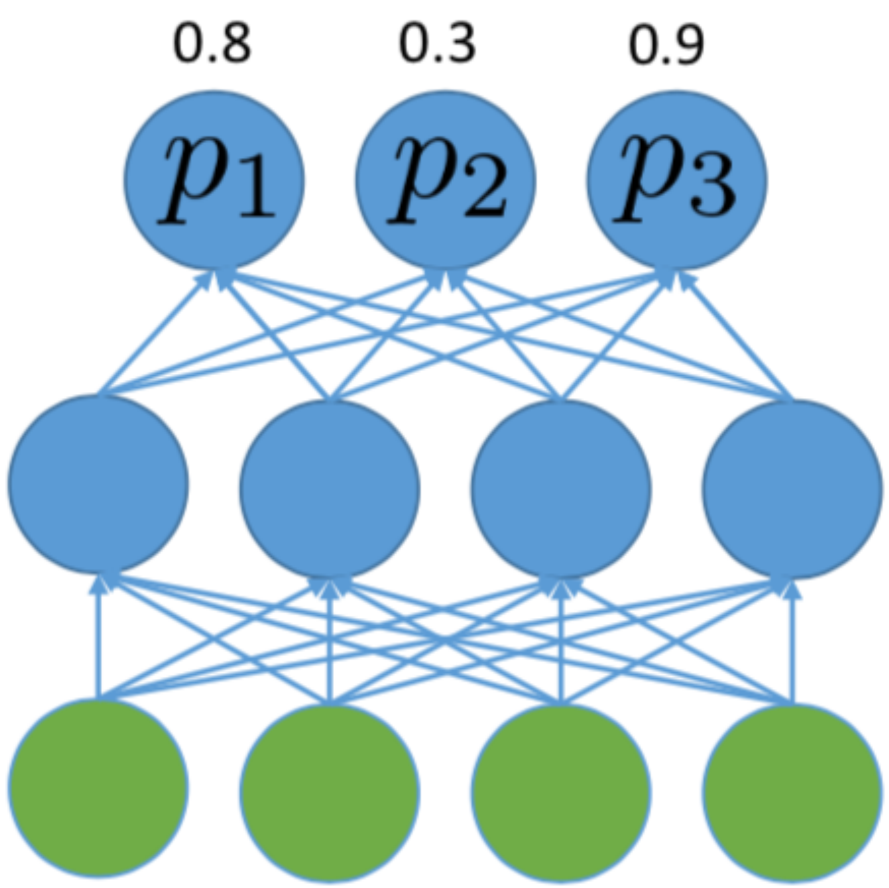
Z=Omer

…

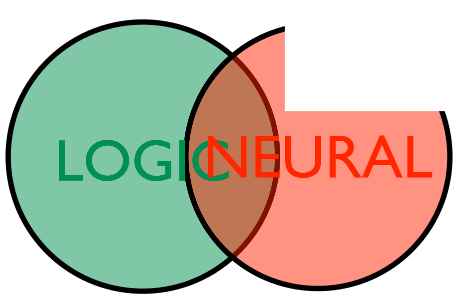
Logic as constraints

undirected StarAI approach and (soft) constraints

multi-class classification



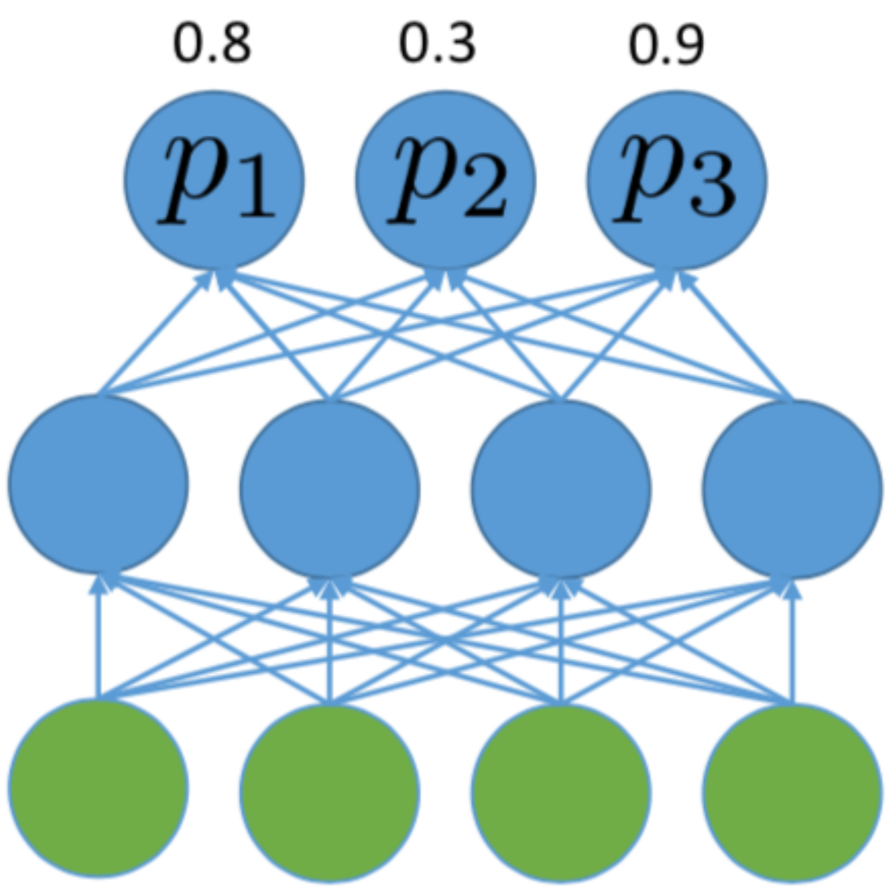
from Xu et al., ICML 2018



Logic as constraints

undirected StarAI approach and (soft) constraints

multi-class classification



This constraint should be satisfied

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

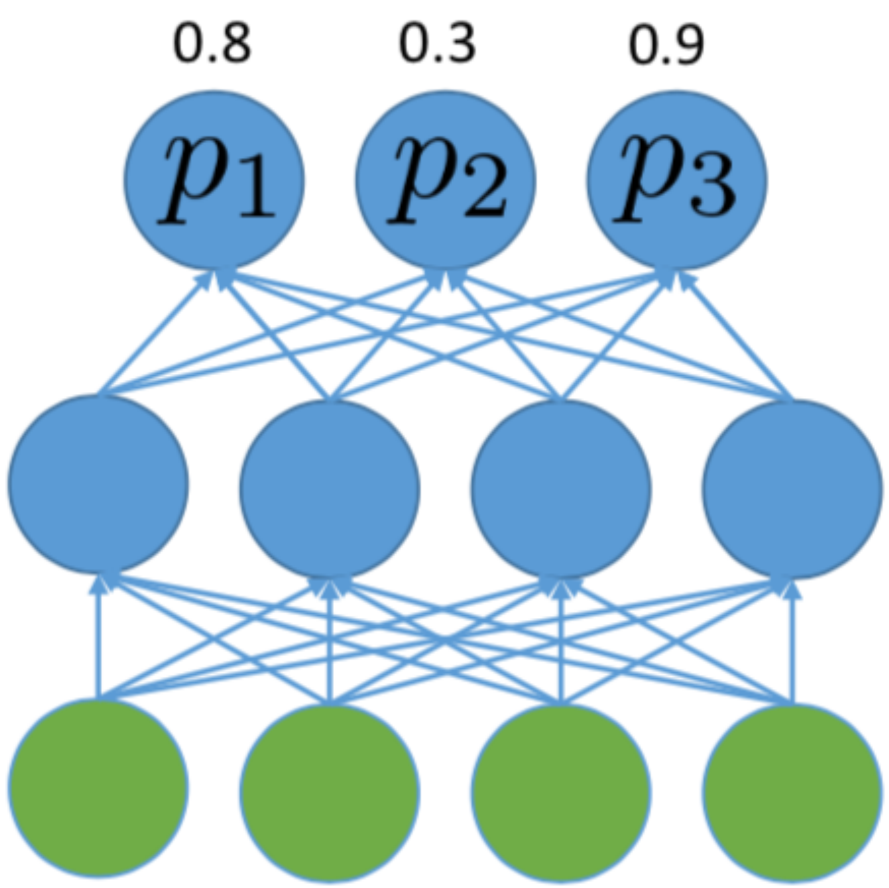
from Xu et al., ICML 2018

LOGIC NEURAL

Logic as constraints

undirected StarAI approach and (soft) constraints

multi-class classification

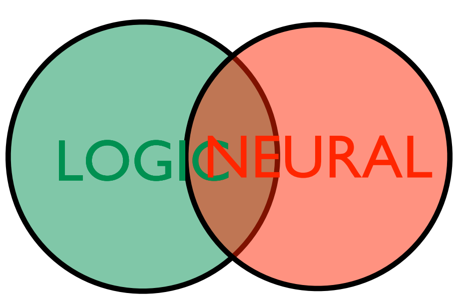


Probability that constraint is satisfied

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

basis for SEMANTIC LOSS

(weighted model counting)

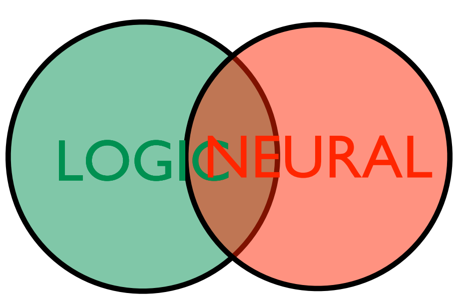


Logic as a regularizer

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



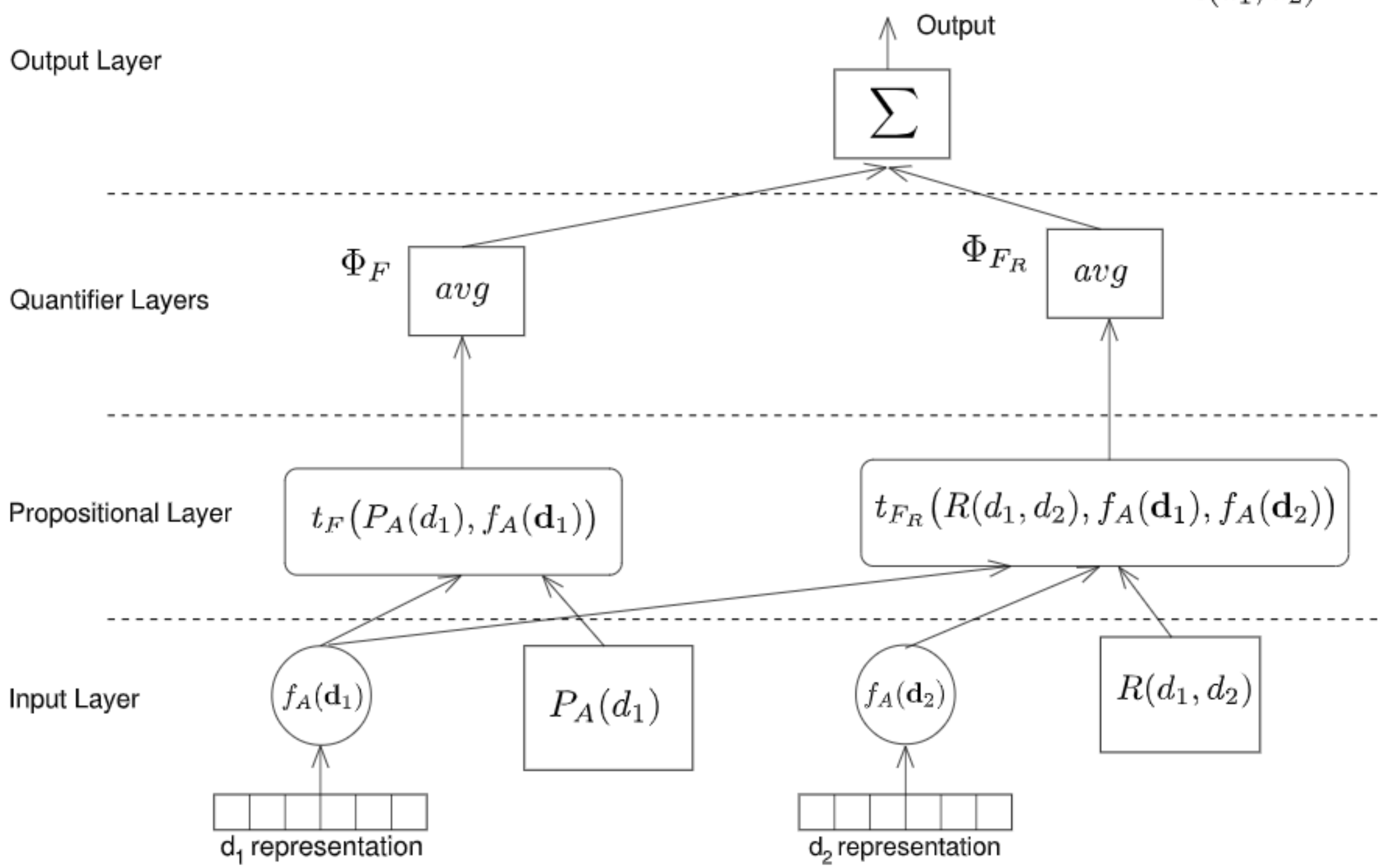
Semantic Based Regularization

undirected StarAI approach and (soft) constraints

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

Evidence Predicate Groundings

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

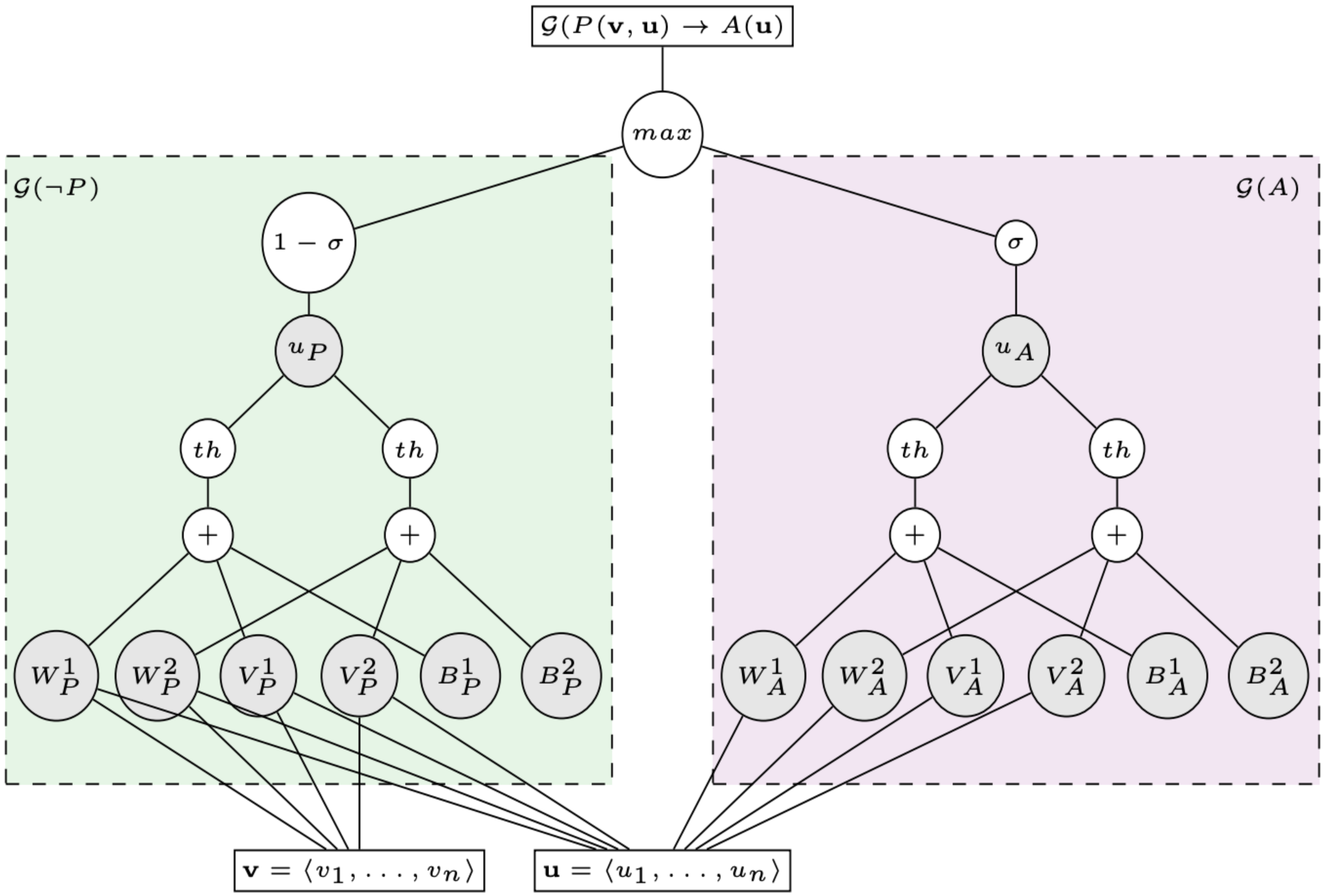


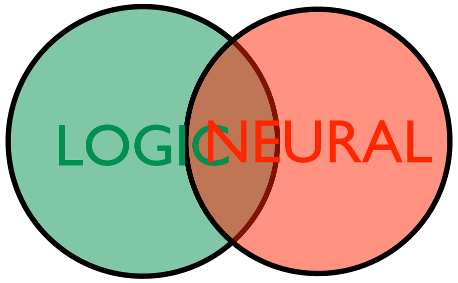
Diligenti et al.

Logic Tensor Networks

undirected StarAI approach and (soft) constraints

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





Serafini & Garcez

Two types of Neural Symbolic Systems

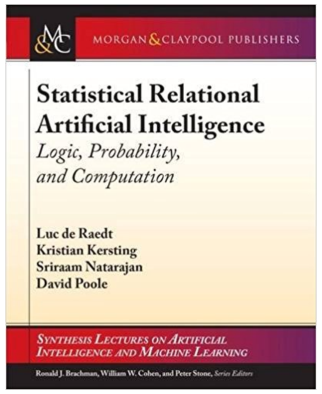
Logic as a ***neural program***

Directed StarAI approach and logic programs

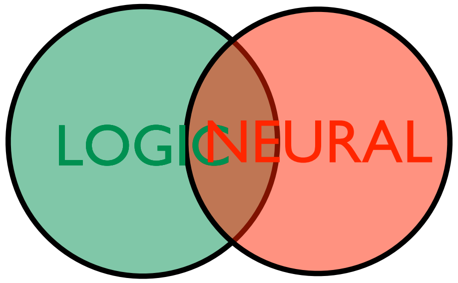
Logic as a ***regularizer***

undirected StarAI approach and (soft) constraints

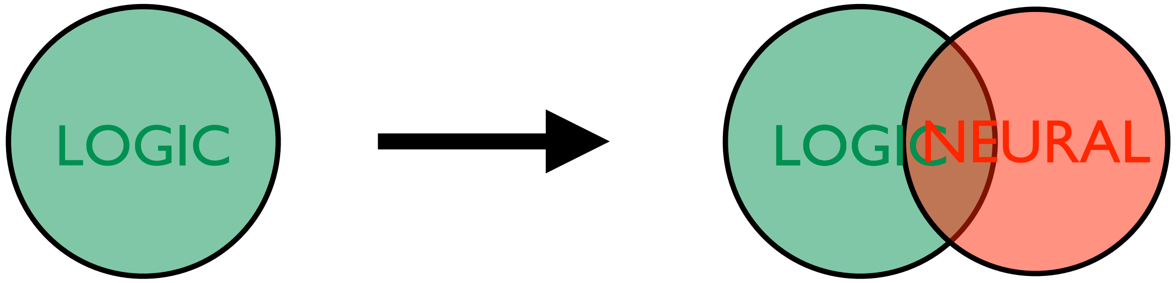
Also, many NeSy systems are doing
knowledge based model construction KBMC
where logic is used as a template



Just like in StarAI



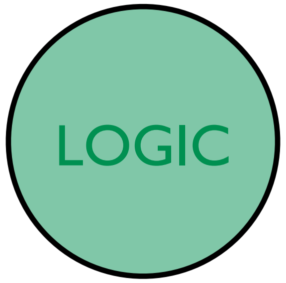
3. Types of Logic



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).
```

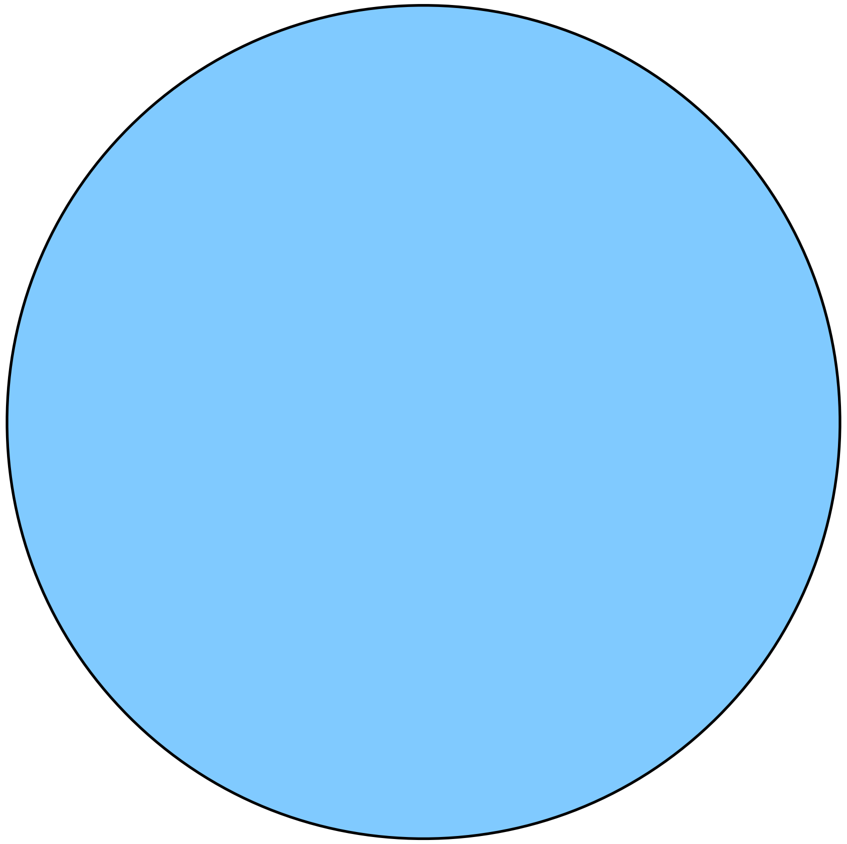
First-order logic

LOGIC

Various flavours of first-order logic

Logic programs
= programming language

FOL constraints



LOGIC

Logic programming and Prolog

Full-fledged programming language

structured terms

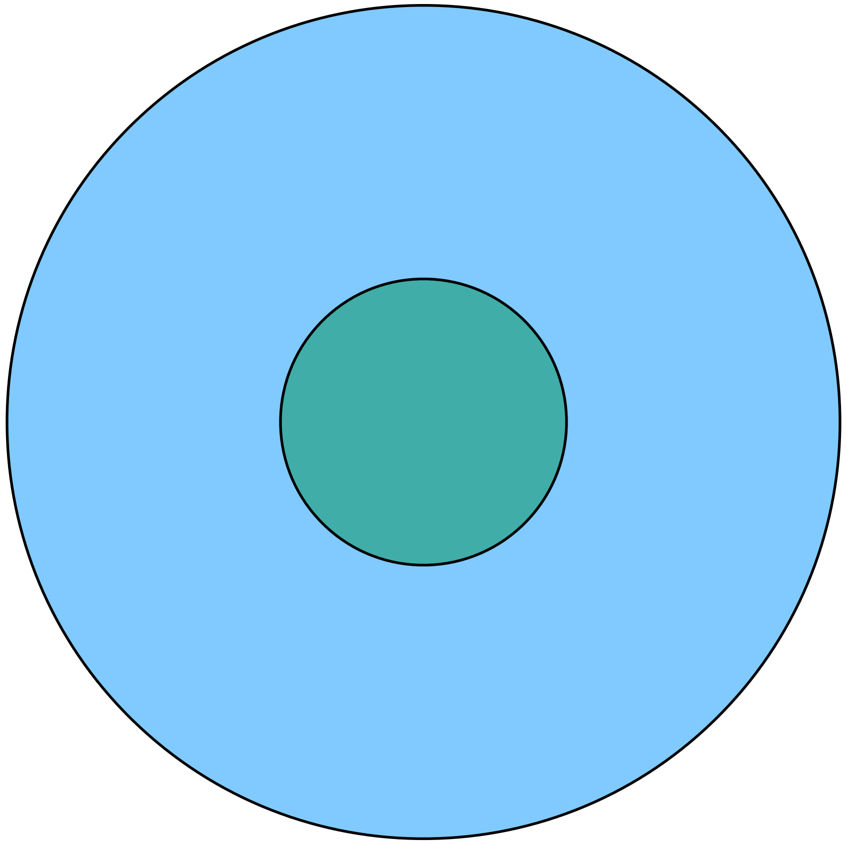
```
member(X, [X|_]).

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

recursion

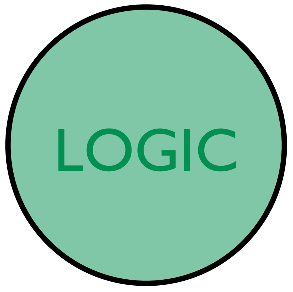
LOGIC

Various flavours of first-order logic

Logic programs
= programming language



Datalog
= Logic programs
that always terminate



Datalog

Query language for deductive databases

no structured terms

guaranteed to terminate

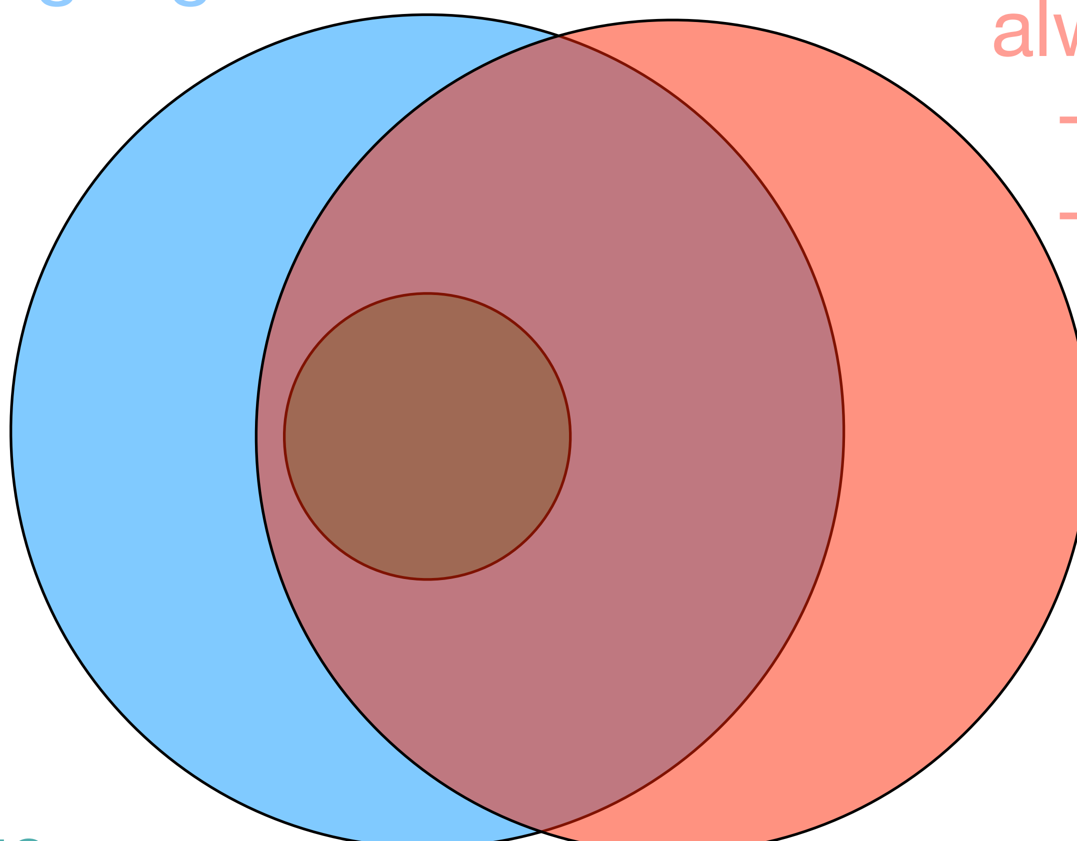
```
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

LOGIC

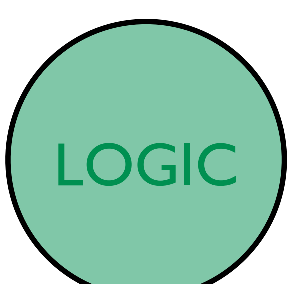
Various flavours of first-order logic

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

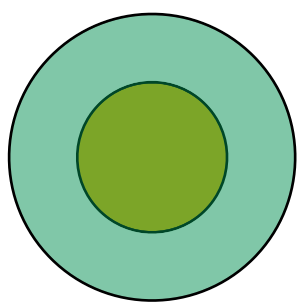
LOGIC

What can it do?

Propositional logic:
simple propositional reasoning

LOGIC

What can it do?

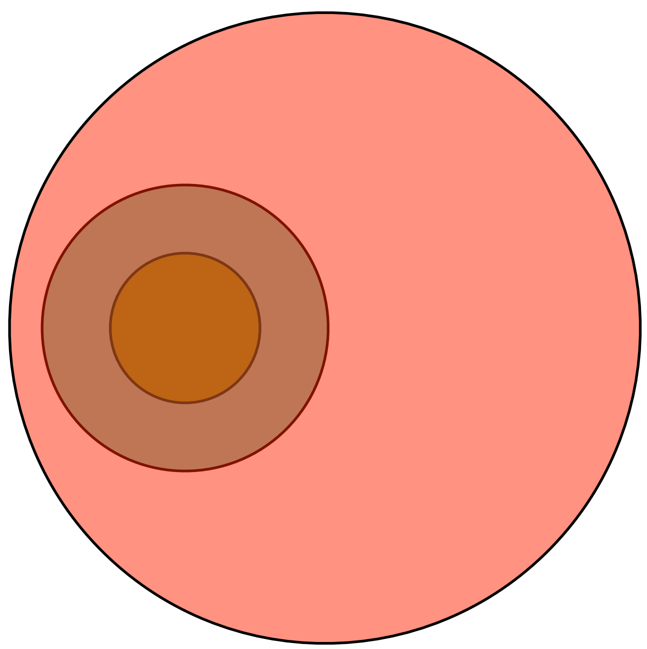


Datalog:
database queries

Propositional logic:
simple propositional reasoning

LOGIC

What can it do?



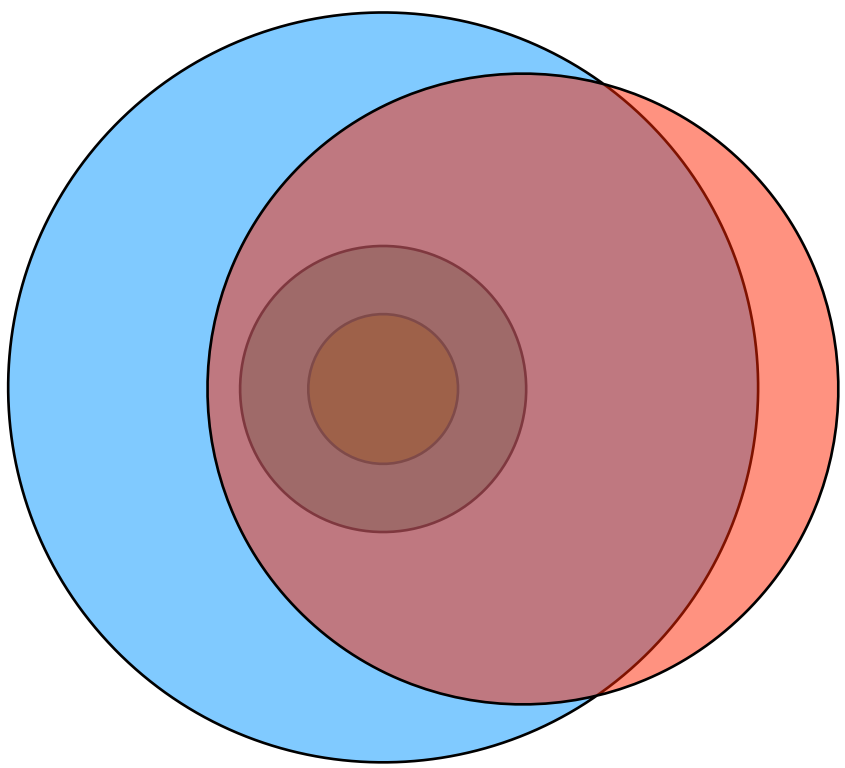
Answer-set programming:
database queries, common-sense reasoning, preferences

Datalog:
database queries

Propositional logic:
simple propositional reasoning

LOGIC

What can it do?



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

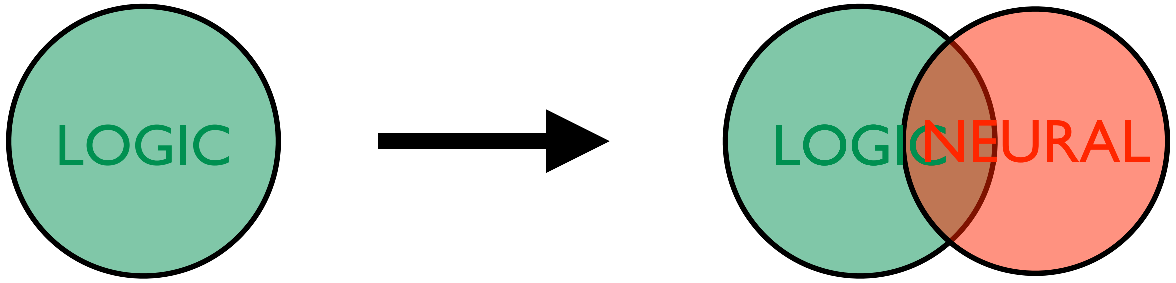
Answer-set programming:
database queries, common-sense reasoning, preferences

Datalog:
database queries

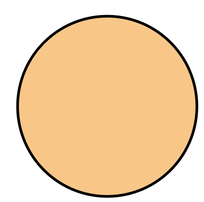
Propositional logic:
simple propositional reasoning

LOGIC

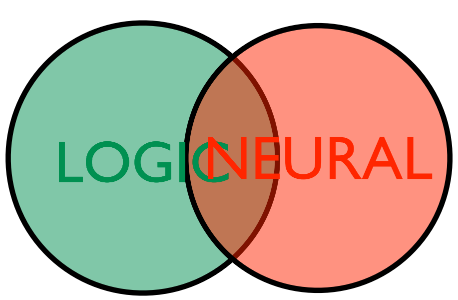
3. Types of Logic



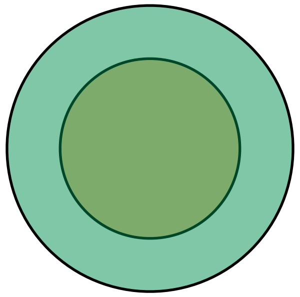
Logic in NeSy - Propositional logic



Semantic loss

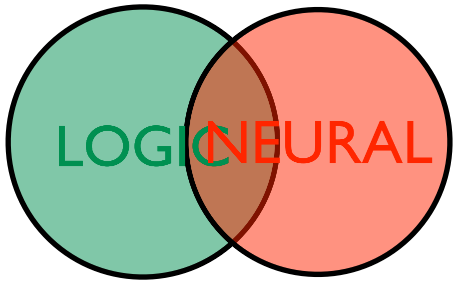


Logic in NeSy - Datalog

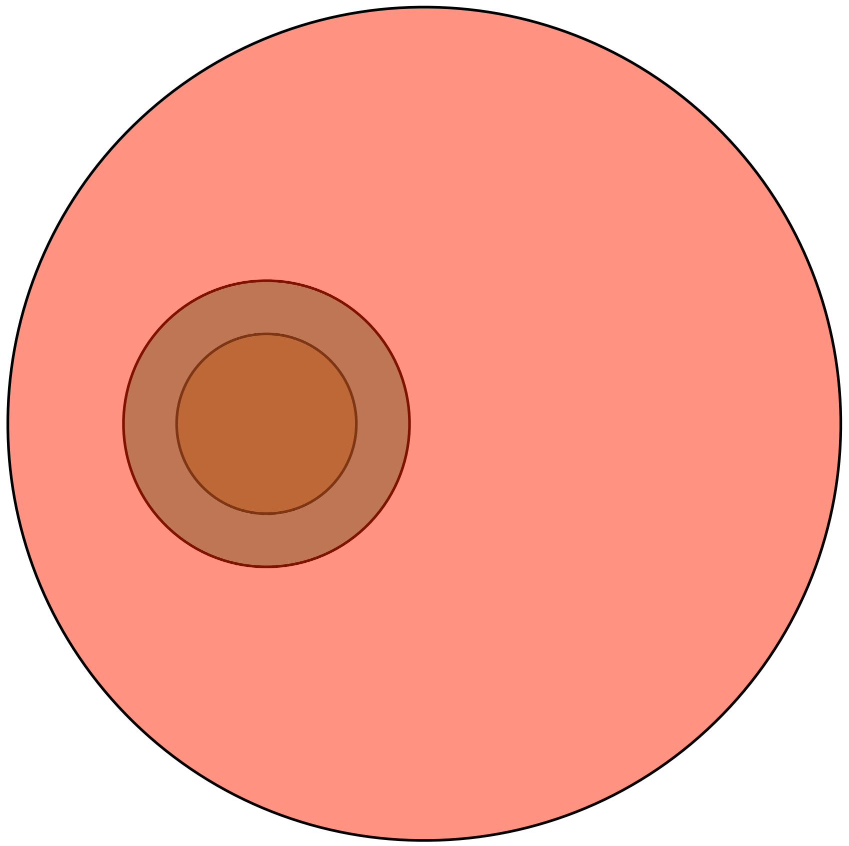


∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



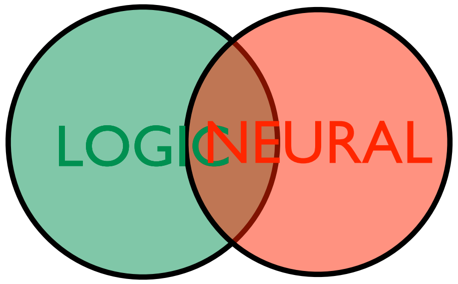
Logic in NeSy - Answer-set programming



NeurASP

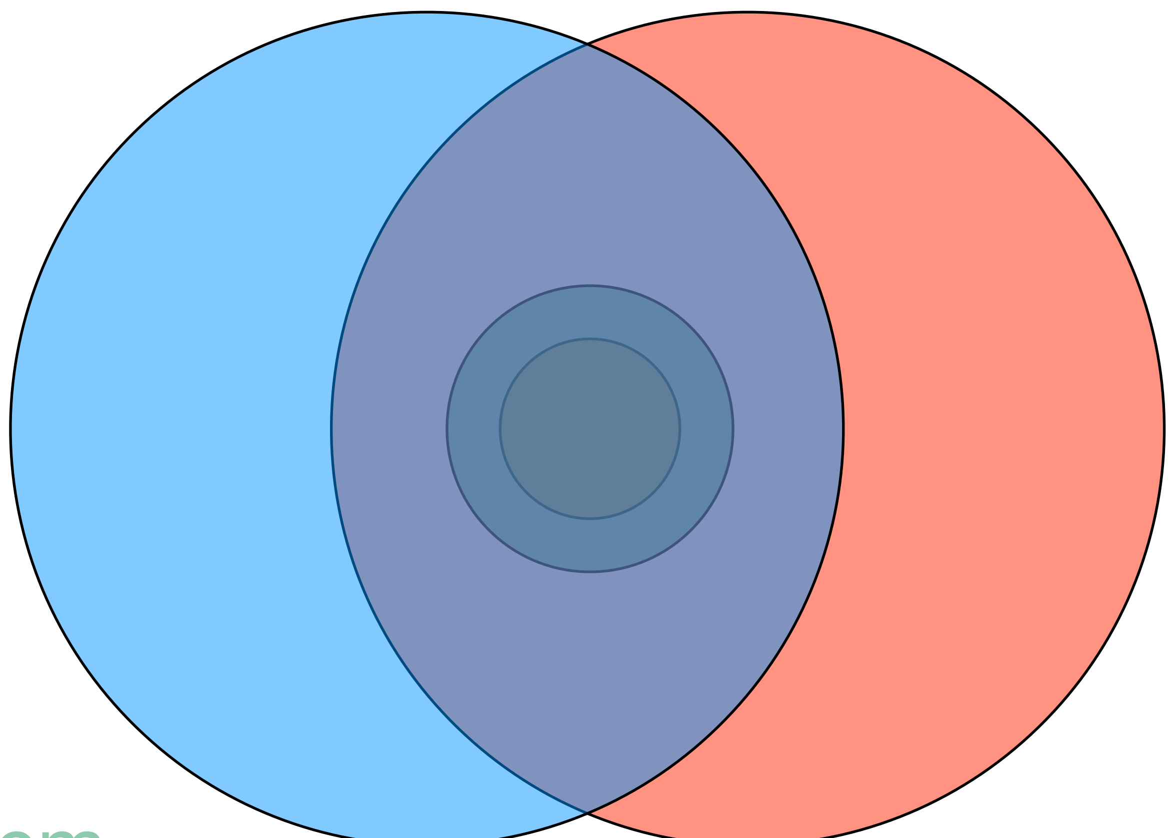
∂ILP, Neural Theorem Provers, LRNN, DiffLog, …

Semantic loss



Logic in NeSy - Logic programming

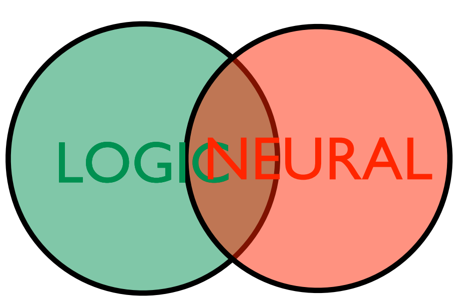
DeepProblog,
NLProlog



NeurASP

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

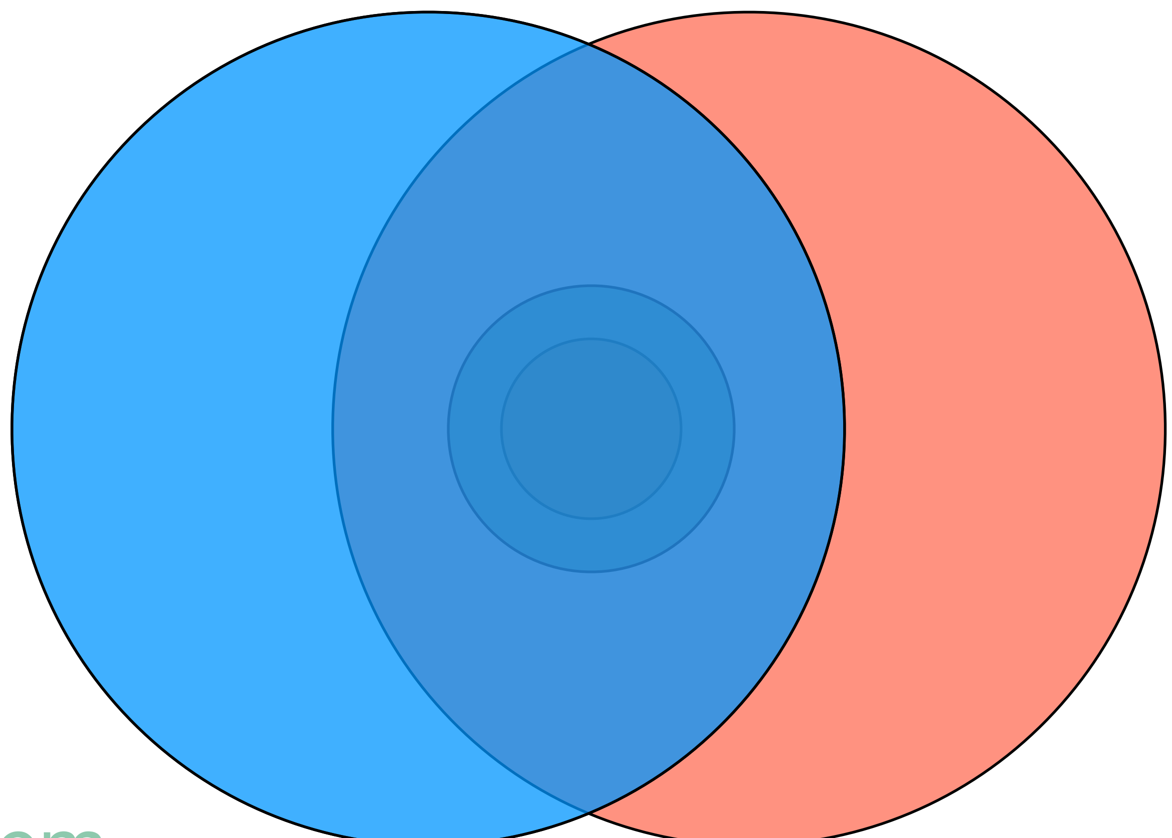
Semantic loss



Logic in NeSy - First-order logic

Logic tensor networks, NMLN,
SBR, RNM

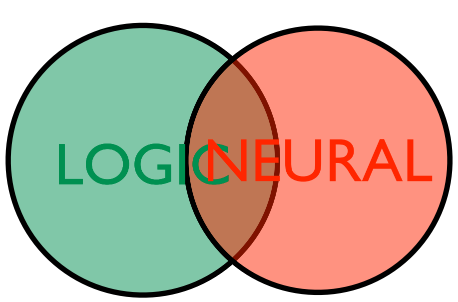
DeepProblog,
NLProlog



NeurASP

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

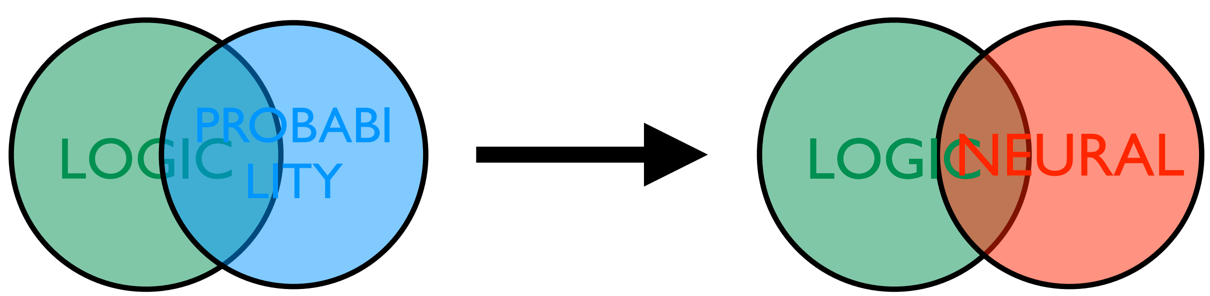
Semantic loss



3. Types of Logic
Key Messages

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

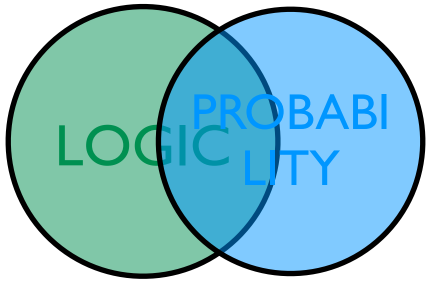
5. Structure vs parameter learning



5. Learning
Key Messages

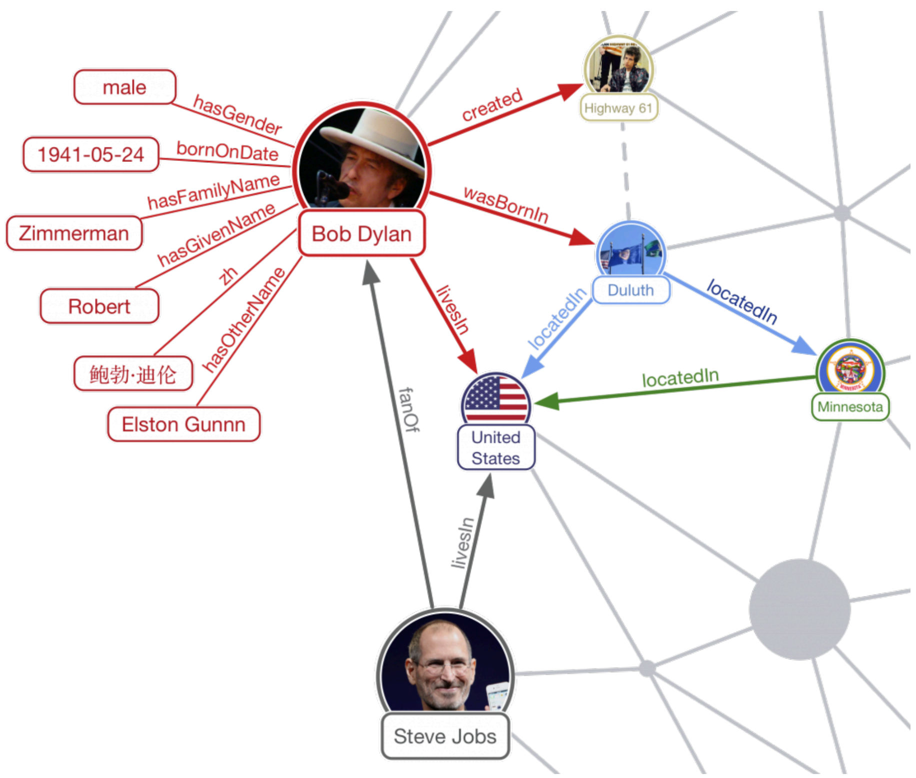
- Learning: finding logical formulas and estimating probabilities
- Structure learning: both formulas and probabilities
- Parameter learning: only probabilities
- Many flavours of learning in NeSy

5. Structure vs parameter learning



Learning in StarAI

Obtaining models from data

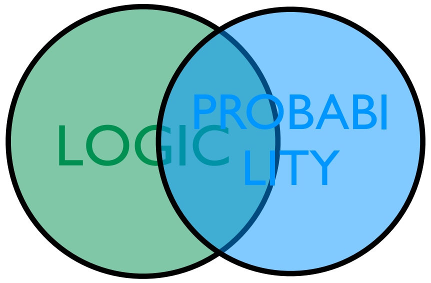


$\mapsto$

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

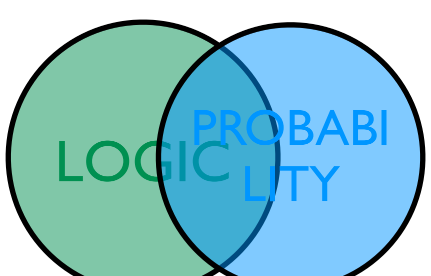
0.7::nationality(X,Y) :-
livesIn(X,Z), locatedIn(Z,Y).

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



StarAI learning paradigms

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

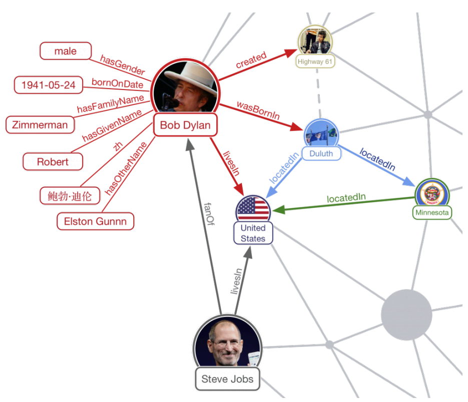


Learning types: Parameter learning

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).

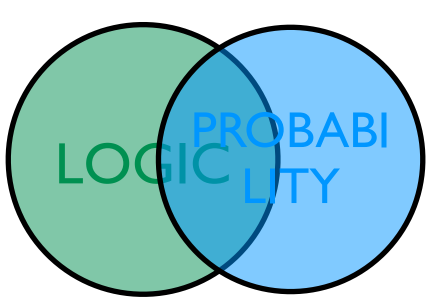
nationality(X,Y) :-
bornIn(X,Y).

↦

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).



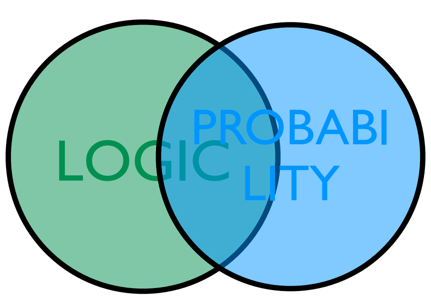
Learning types: Parameter learning

Learning the probabilities/weights of a specified model

Model (the formulas) are given

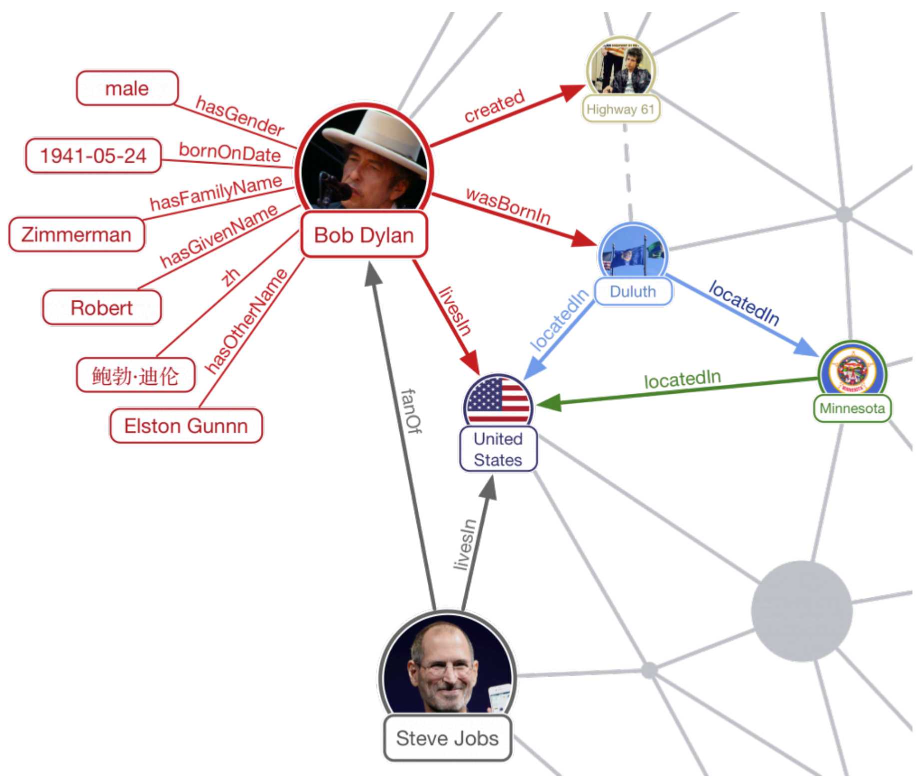
Learning principles: identical to learning parameters of any parametric model

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



Learning types: Structure learning

Finding the clauses/logical formulas of a model



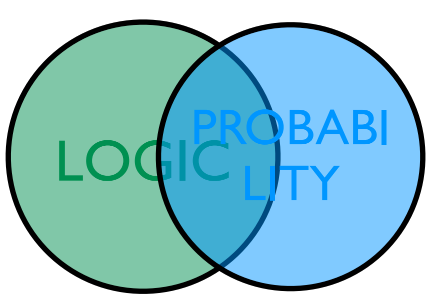
↦

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

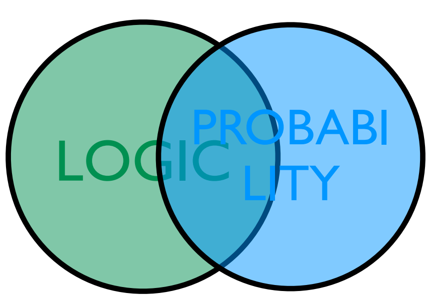
Two types of structure learning

Discriminative

- specific target relation
- separate background knowledge

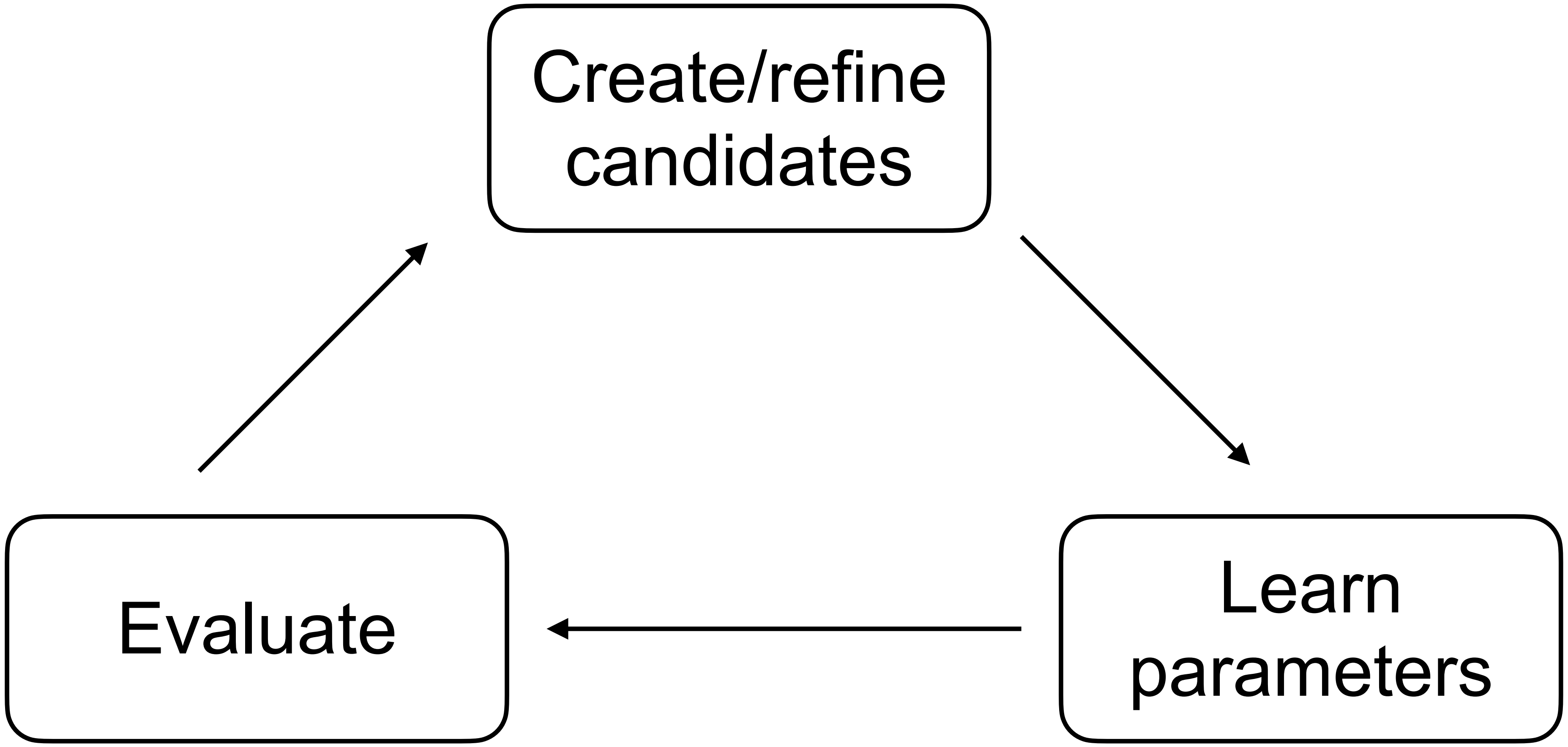
Generative

- no specific target relation
- learning generative process behind data



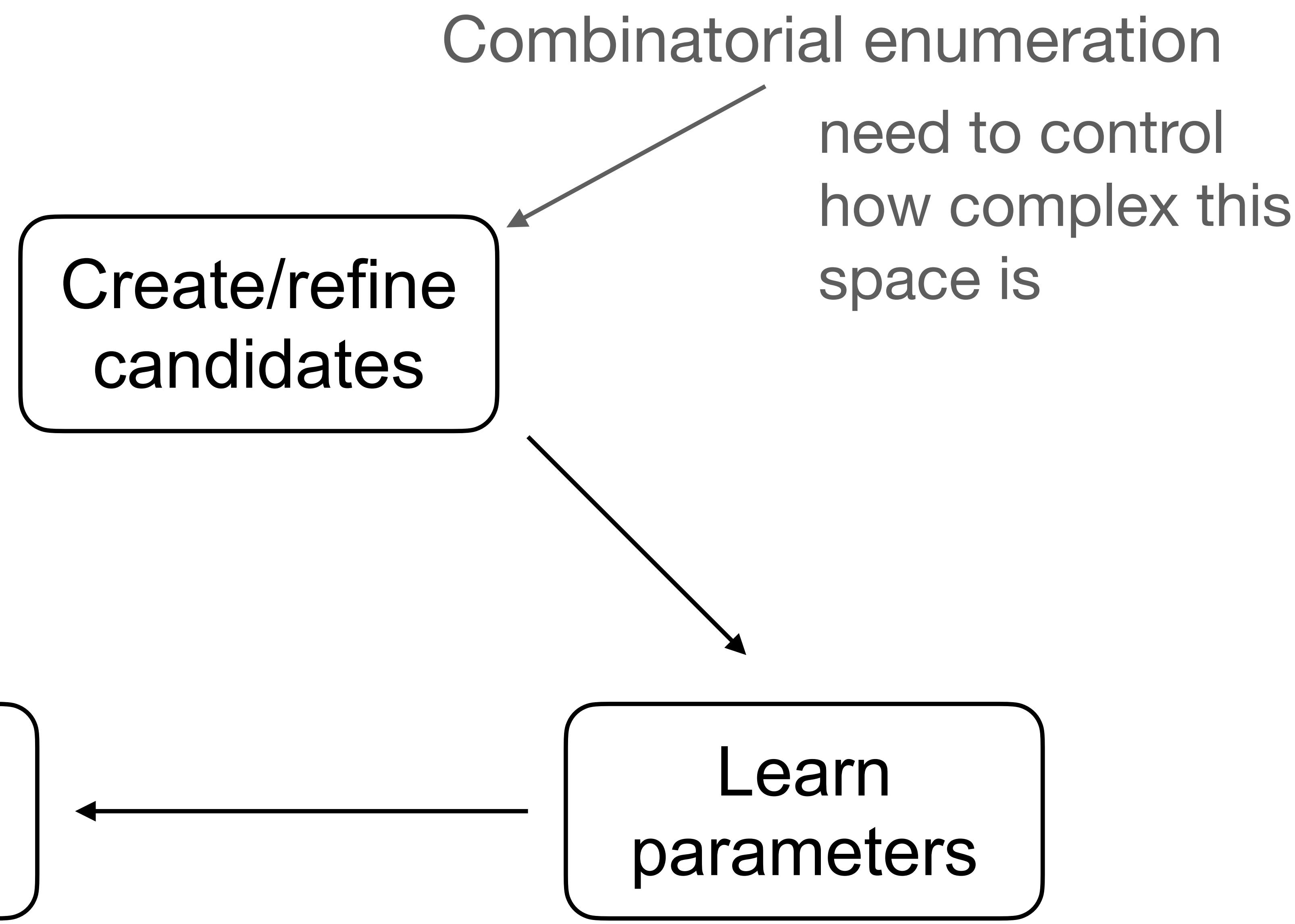
Learning types: Structure learning

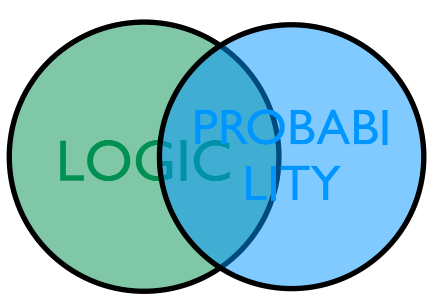
Learning by searching



Learning types: Structure learning

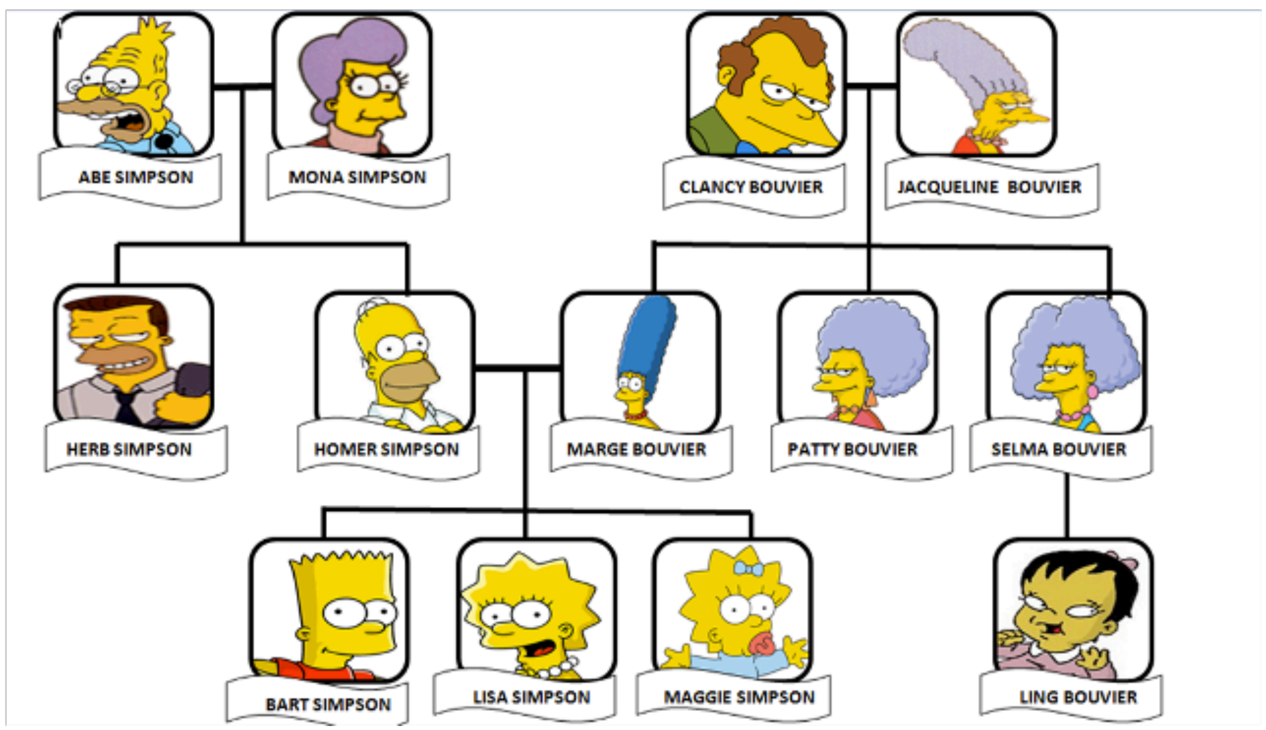
Learning by searching





Learning via enumeration - Probfoil+

[De Raedt et al, 2015]



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

LOGIC PROBABI LITY

Learning via enumeration - Probfoil+

[De Raedt et al, 2015]

Model: {}

Learning via enumeration - Probfoil+

[De Raedt et al, 2015]

Model: {}

Learn one rule: p:: grandparent(X,Y) ← *true*

Learning via enumeration - Probfoil+

[De Raedt et al, 2015]

Model: {}

if not good enough, refine!

Learn one rule: ~~p:: grandparent(X,Y) ← *true*~~

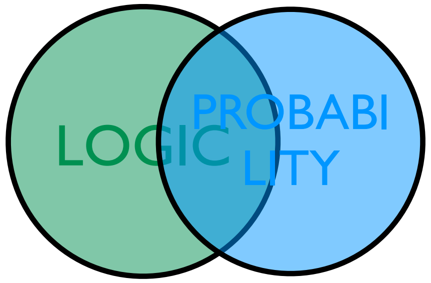
p:: grandparent(X,Y) ← mother(X,Y)

p:: grandparent(X,Y) ← mother(Y,X)

p:: grandparent(X,Y) ← mother(X,Z)

p:: grandparent(X,Y) ← father(X,Y)

…..



Learning via enumeration - Probfoil+

[De Raedt et al, 2015]

Model: {}

if not good enough, refine!

Learn one rule: ~~p:: grandparent(X,Y) ← *true*~~

~~p:: grandparent(X,Y) ← mother(X,Y)~~
~~p:: grandparent(X,Y) ← mother(Y,X)~~
~~p:: grandparent(X,Y) ← mother(X,Z)~~
~~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 enumeration - Probfoil+

[De Raedt et al, 2015]

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

if not good enough, refine!

Learn one rule:

~~p:: grandparent(X,Y) ← *true*~~

~~p:: grandparent(X,Y) ← mother(X,Y)~~
~~p:: grandparent(X,Y) ← mother(Y,X)~~
~~p:: grandparent(X,Y) ← mother(X,Z)~~
~~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 enumeration - Probfoil+

[De Raedt et al, 2015]

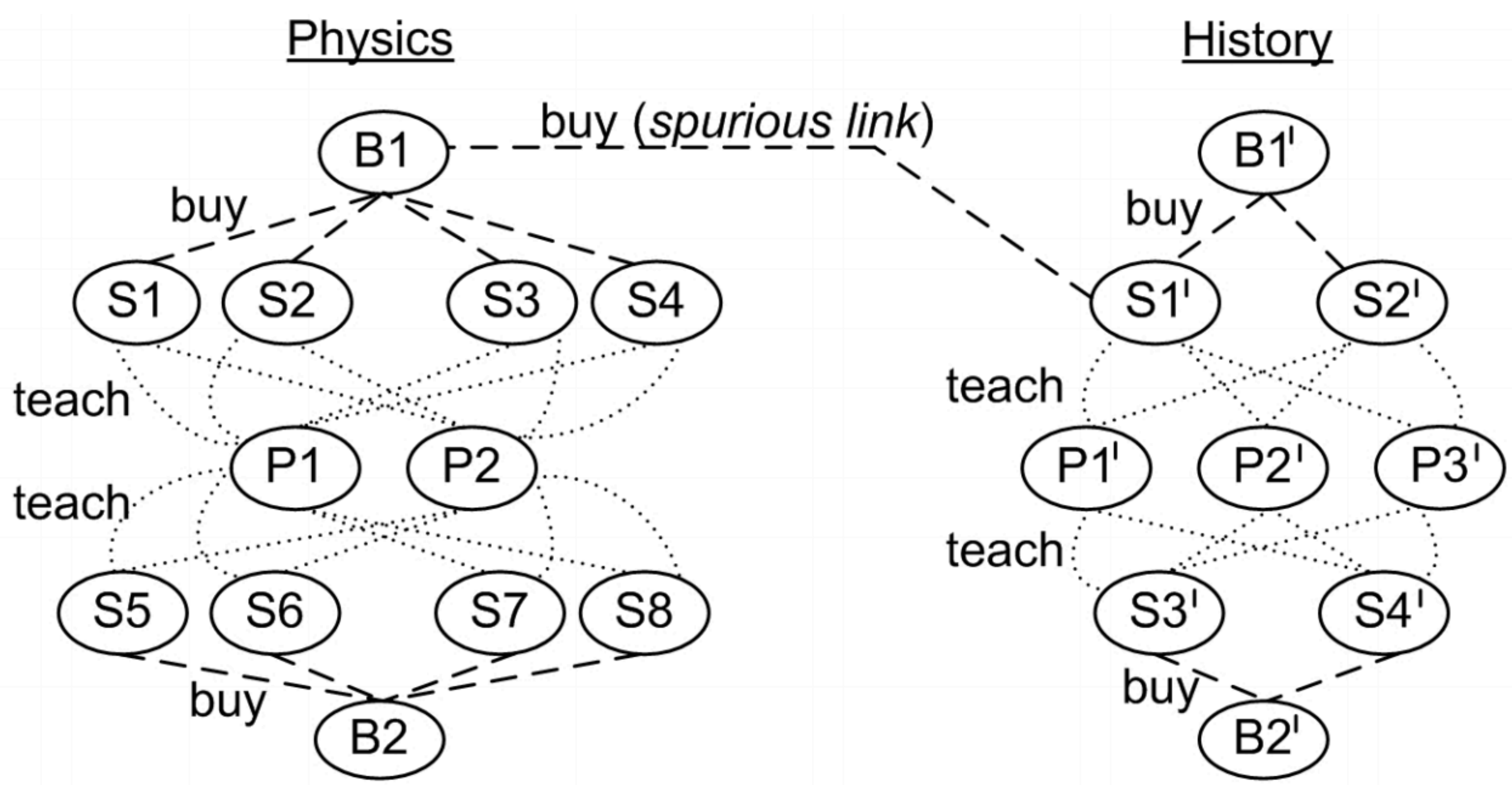
Model: {1.0:: grandparent(X,Y) ← mother(X,Z), father(Z,Y)}

start again with a single rule!

Learn one rule: p:: grandparent(X,Y) ← *true*

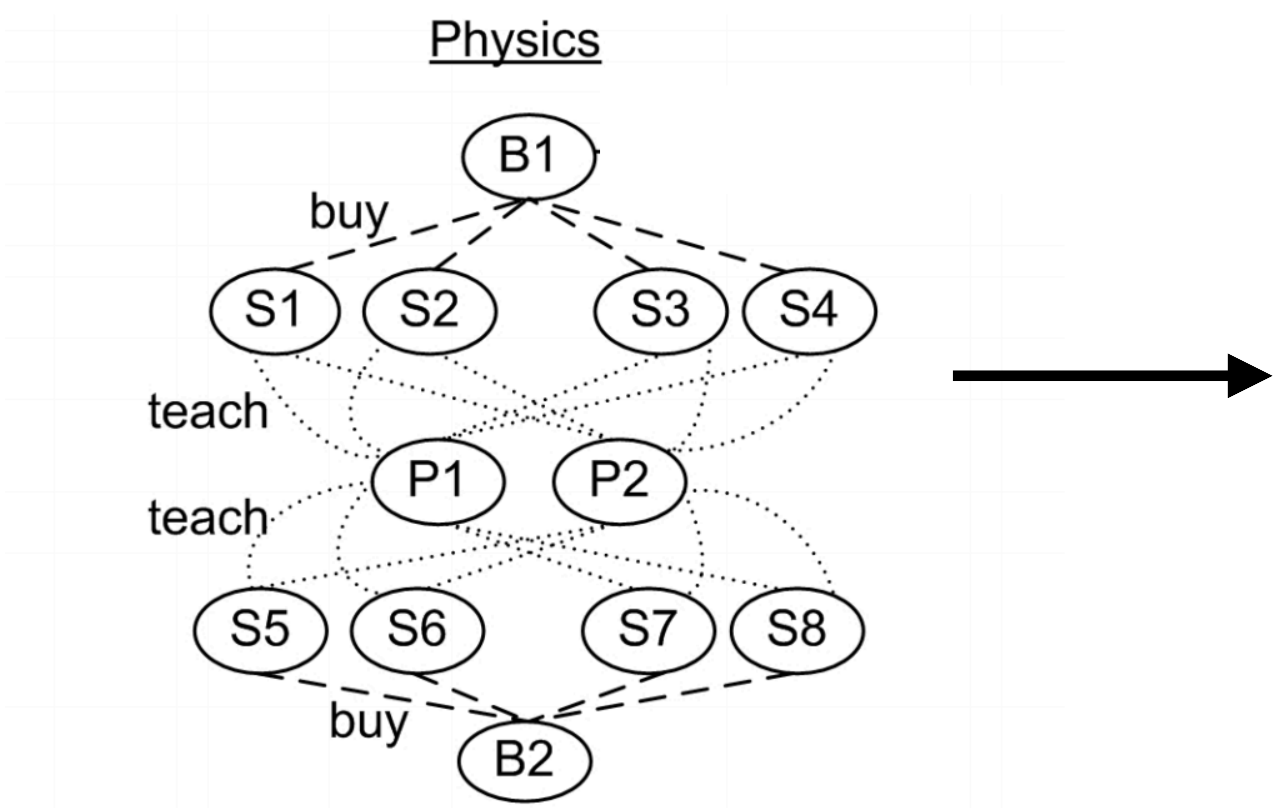
Learning via random walks

[Kok & Domingos, 2009]

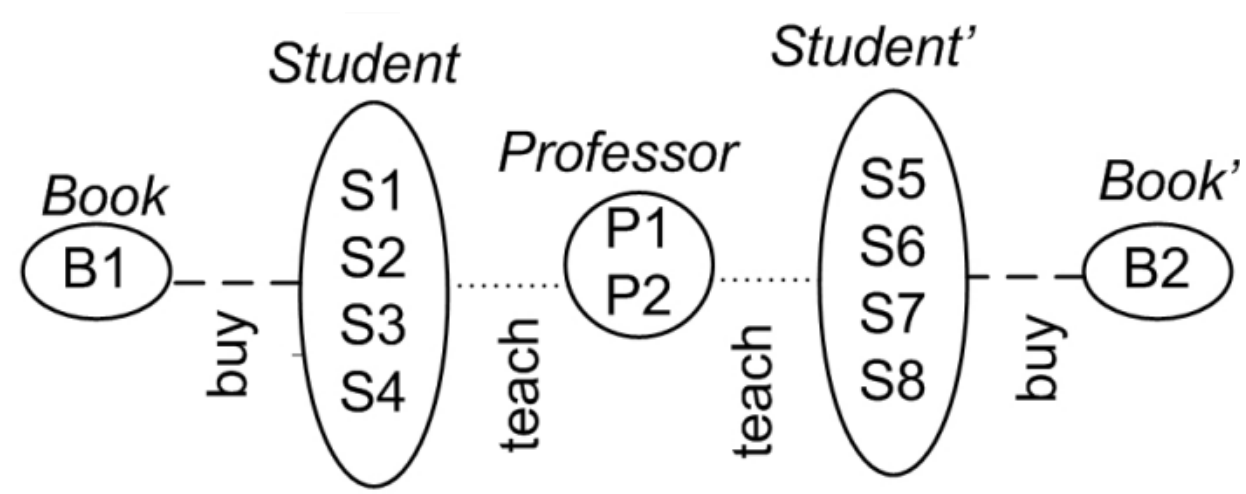


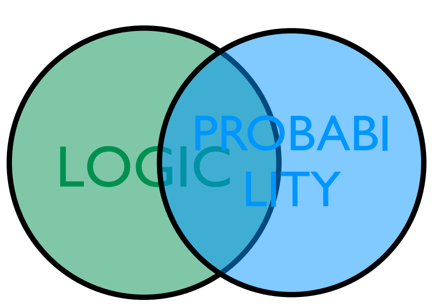
Learning via random walks

[Kok & Domingos, 2009]



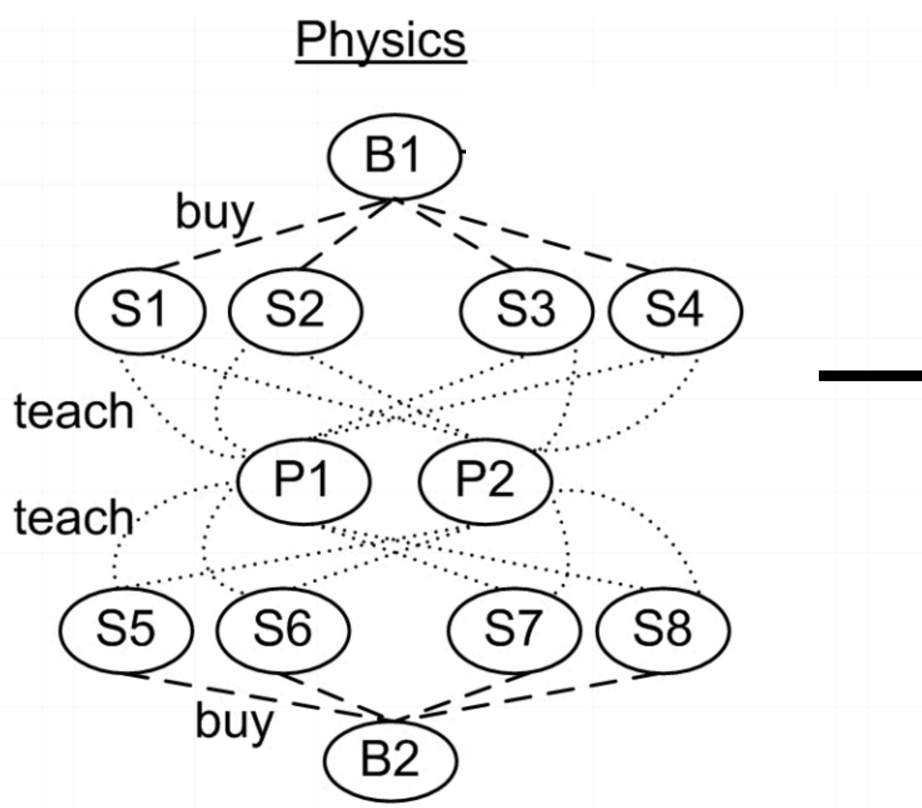
"Lift" a knowledge graph by identifying nodes with the same role



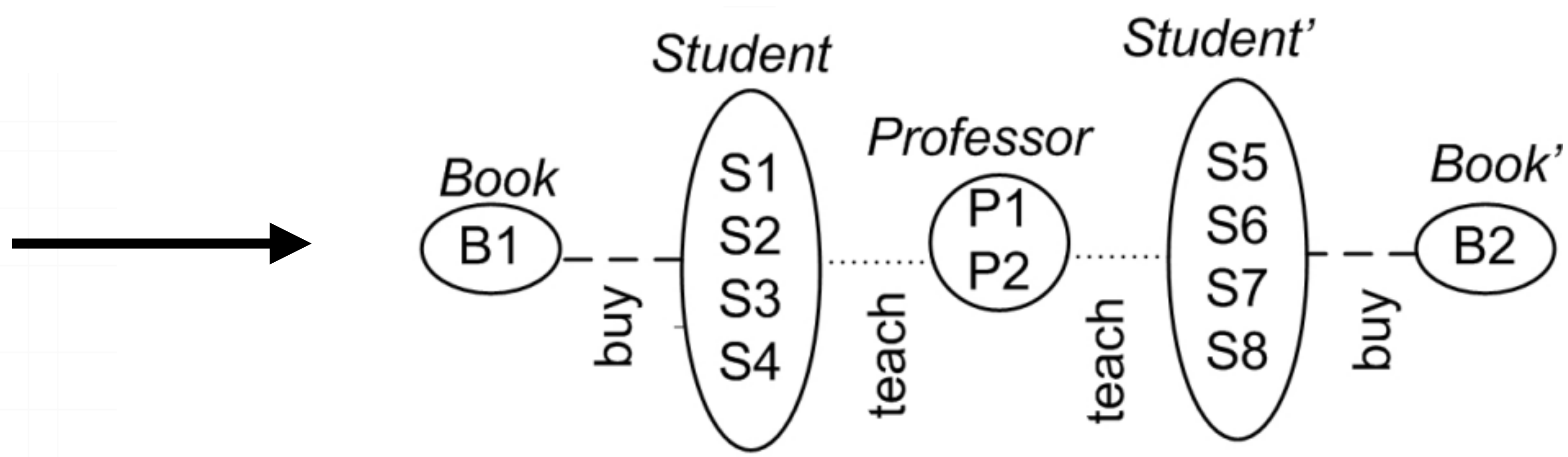


Learning via random walks

[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

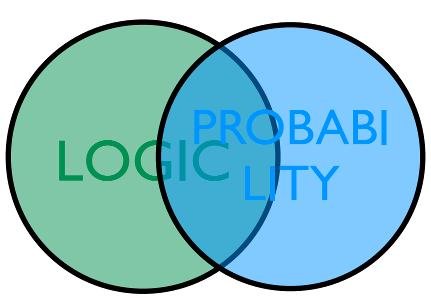
Learning in StarAI - overview

Structure learning

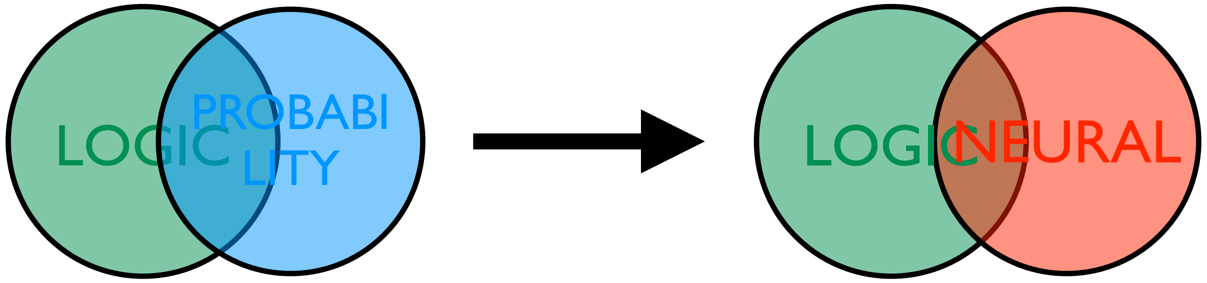
- (+) Starts directly from data
- (−) Combinatorial problem
- (−) User needs to design a language

Parameter learning

- (+) Learning is easier
- (+) Scales better
- (−) An expert needs to provide the rules
- (−) Sensitive to the choice of rules



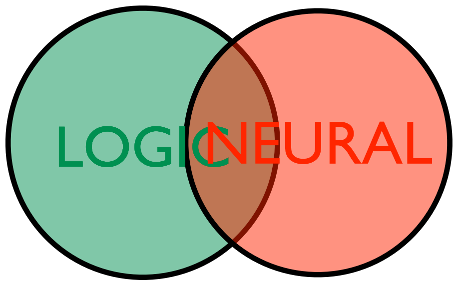
5. Structure vs parameter learning



Spectrum of learning paradigms

Structure learning

Parameter learning



Spectrum of learning paradigms

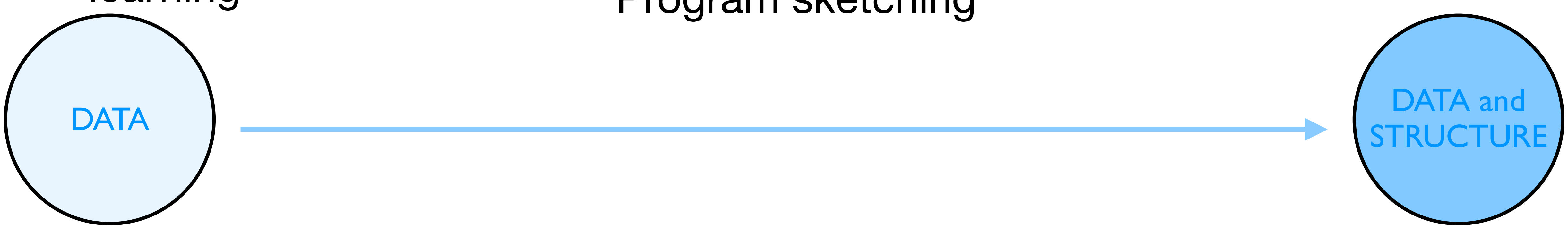
Soft patterns

Neural generation

Structure via
parameter learning

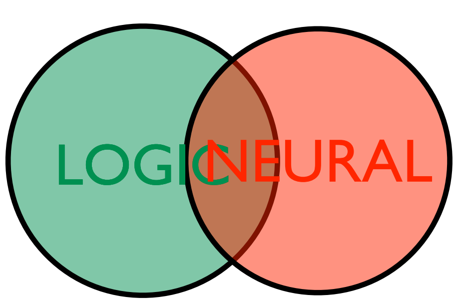
Neurally-guided
learning

Program sketching



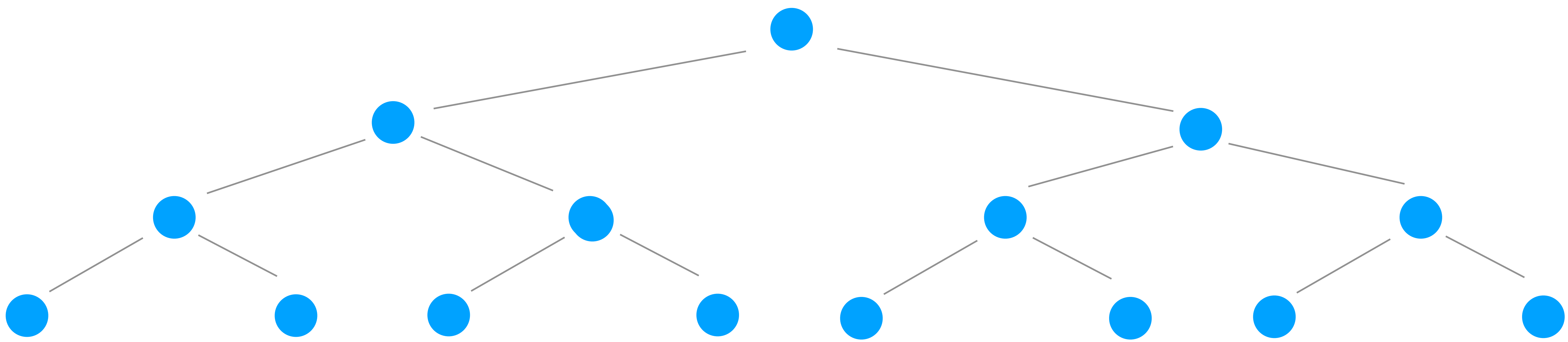
Structure learning

Parameter learning

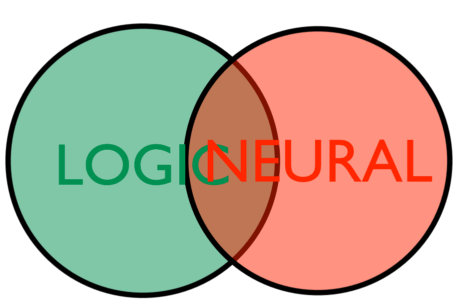


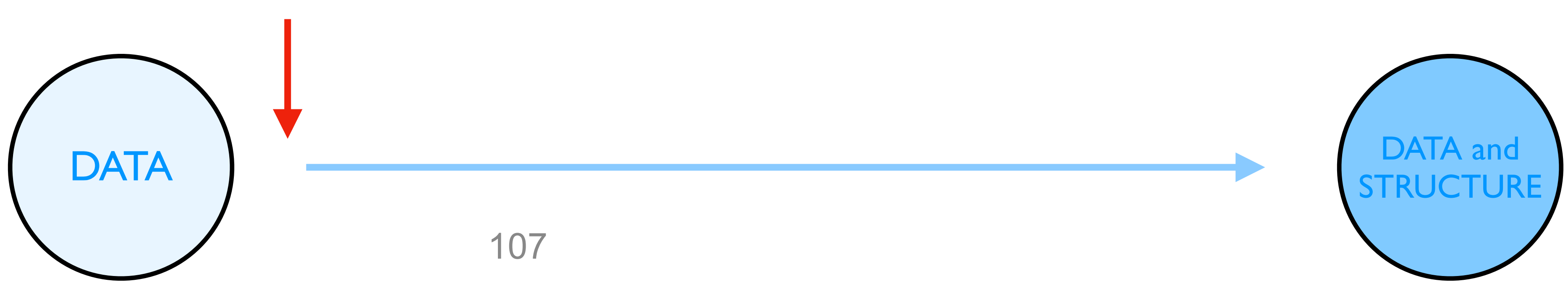
DeepCoder

[Balog et al, 2017]



StarAI techniques search for clauses/rules systematically

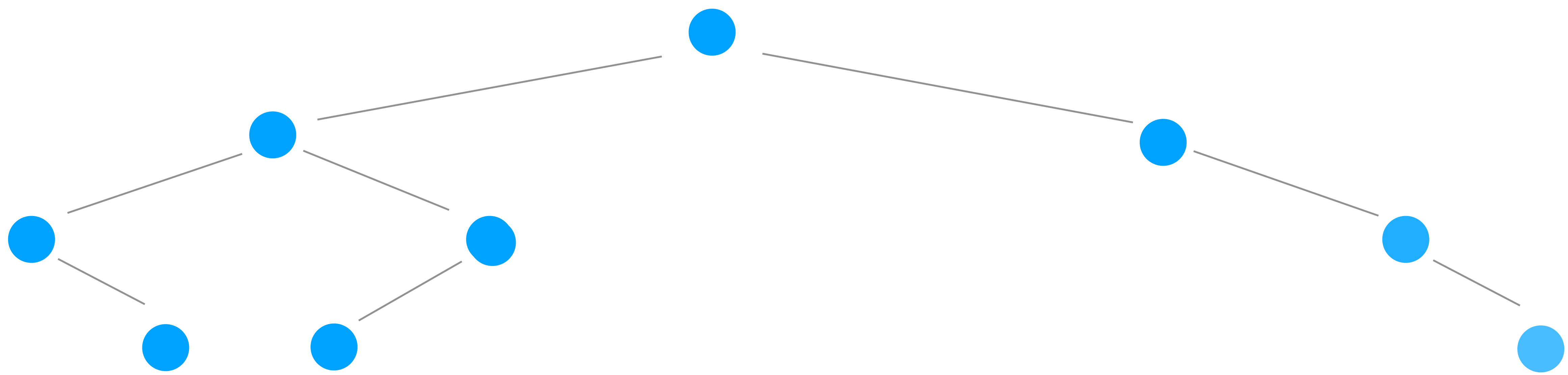




DeepCoder

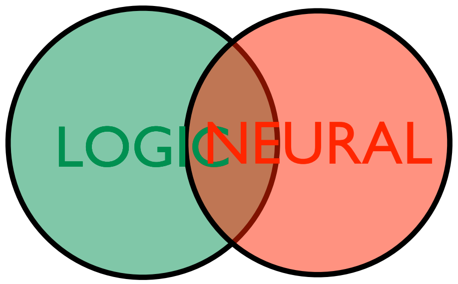
[Balog et al, 2017]

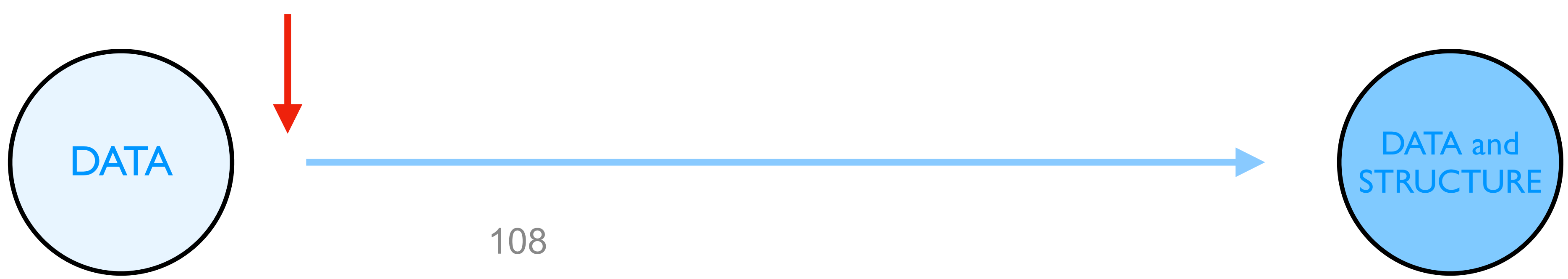
Preferences of learning ‘primitives’



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’

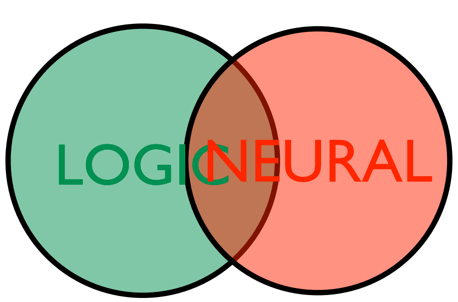
Learn from pairs
(examples, program)

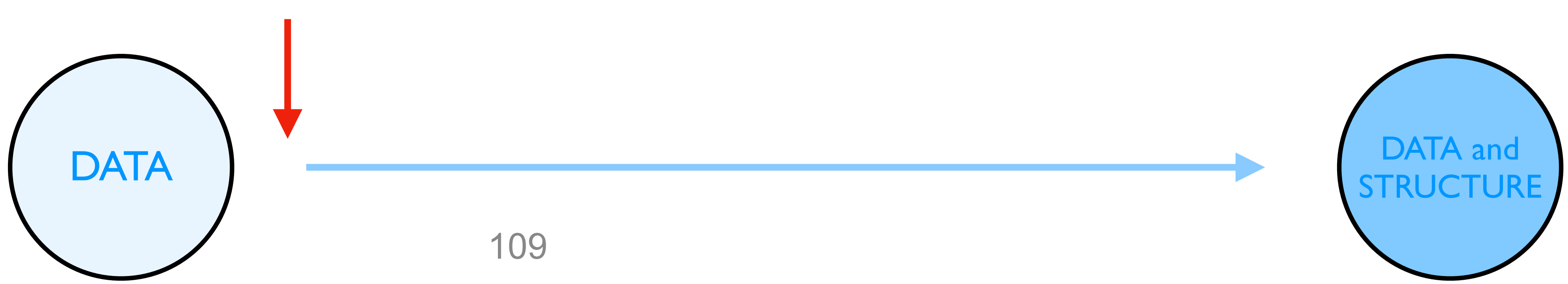
```
a ← [int]
b ← FILTER (<0) a
c ← MAP (*4) b
d ← SORT c
e ← REVERSE d
```

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

Given examples, predict
which functions to use

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





DeepCoder

[Balog et al, 2017]

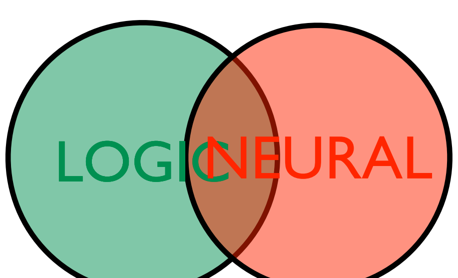
Preferences of learning 'primitives'

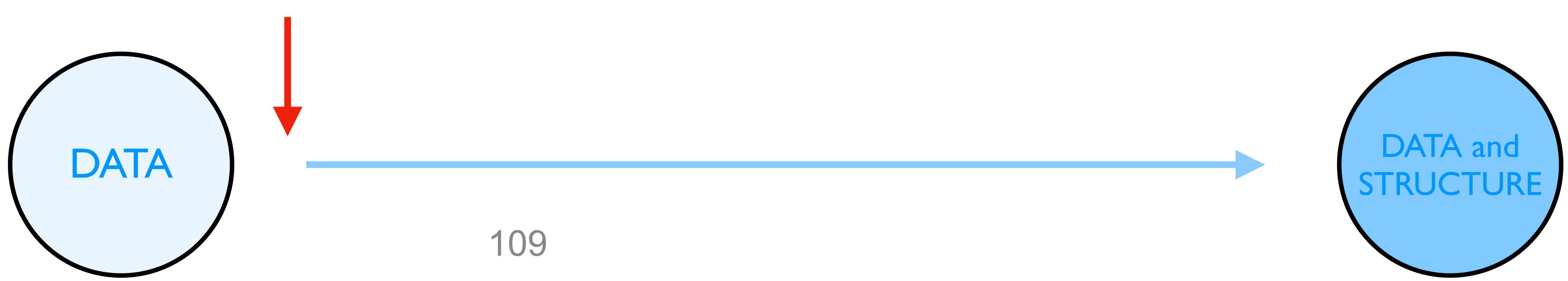
Learn from pairs
(examples, program)

```
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]`

Given examples, predict
which functions to use





DeepCoder

[Balog et al, 2017]

Preferences of learning 'primitives'

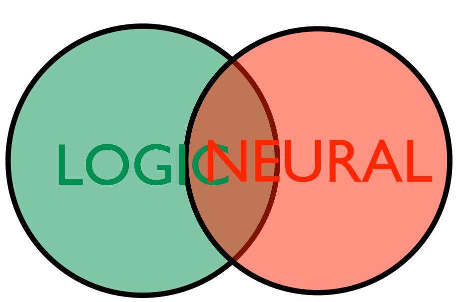
Learn from pairs
(examples, program)

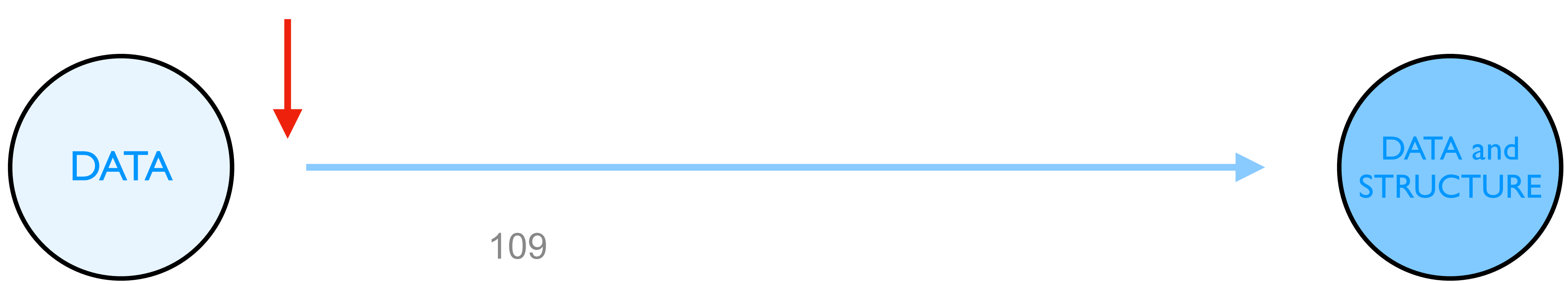
```
a ← [int]
b ← FILTER (<0) a
c ← MAP (*4) b
d ← SORT c
e ← REVERSE d
```

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

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





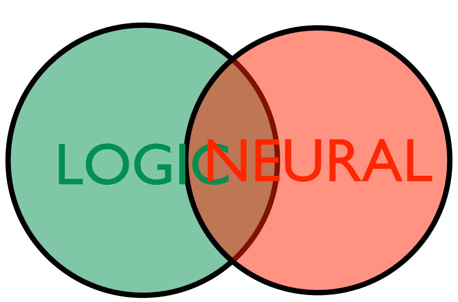
DreamCoder

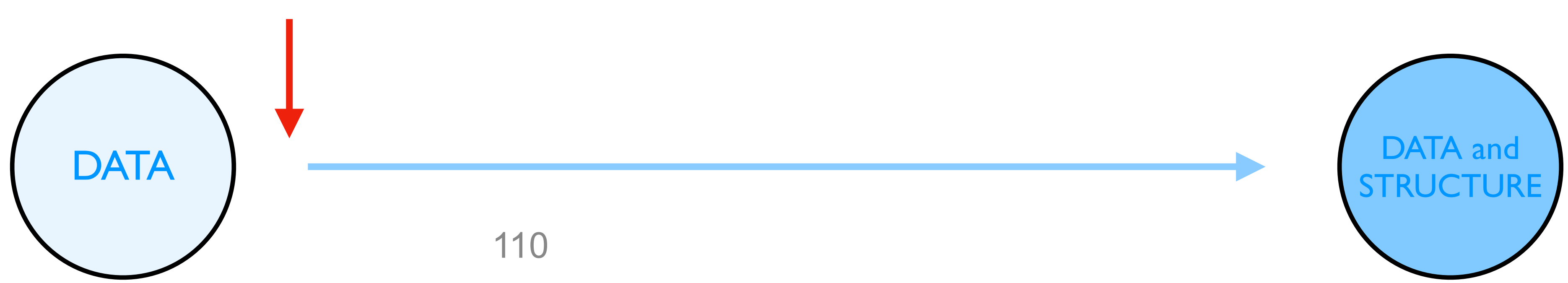
[Ellis et al, 2018]

Distribution of primitives defines a generative model of programs

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

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

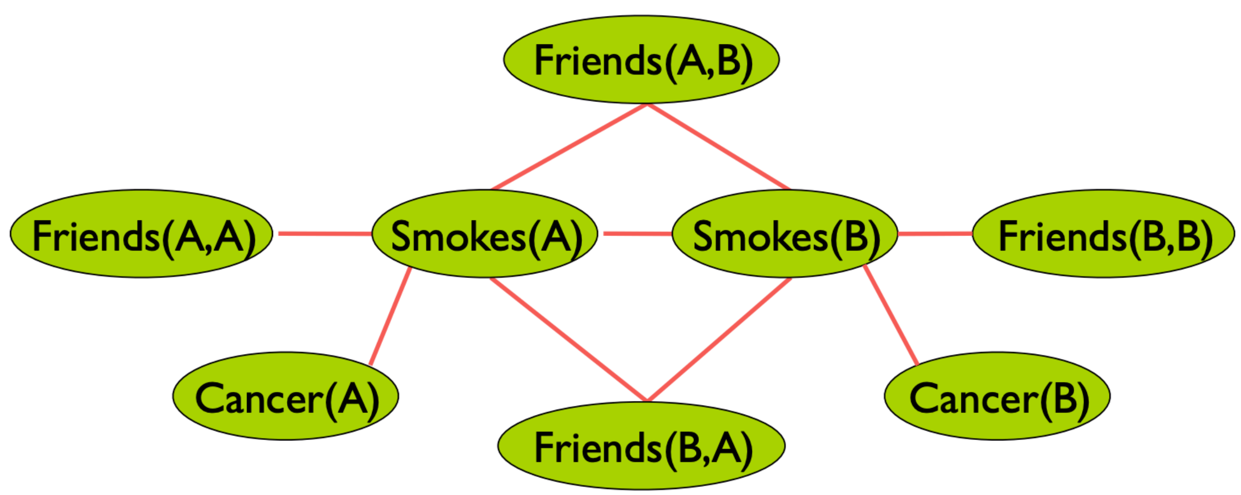




Neural Markov Logic Networks

[Marra et al, 2020]

MLNs can be interpreted as log-linear models



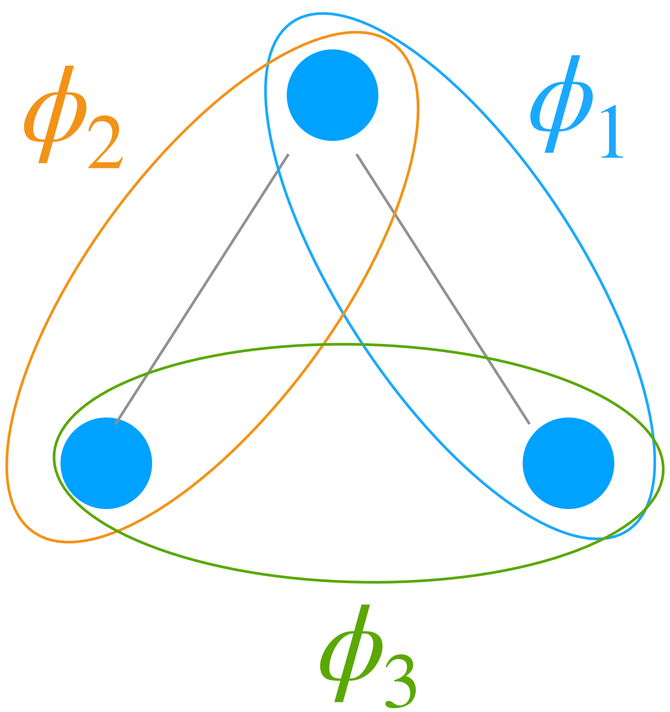
$$P(X = x) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)}$$

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

Neural Markov Logic Networks

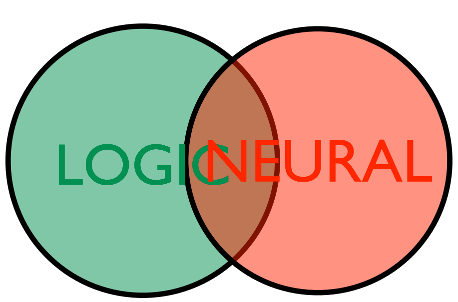
[Marra et al, 2020]

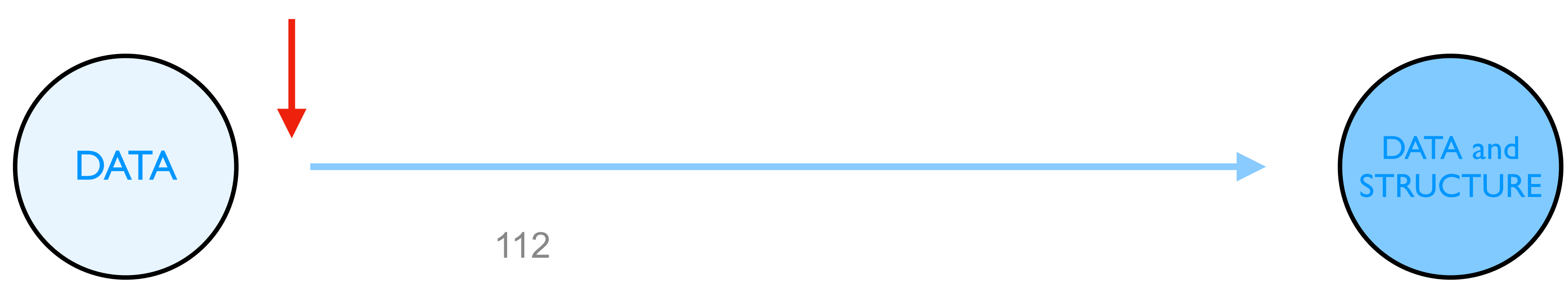
Learn neural potentials from fragments of data



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

potentials come from fragments of data (knowledge graph)

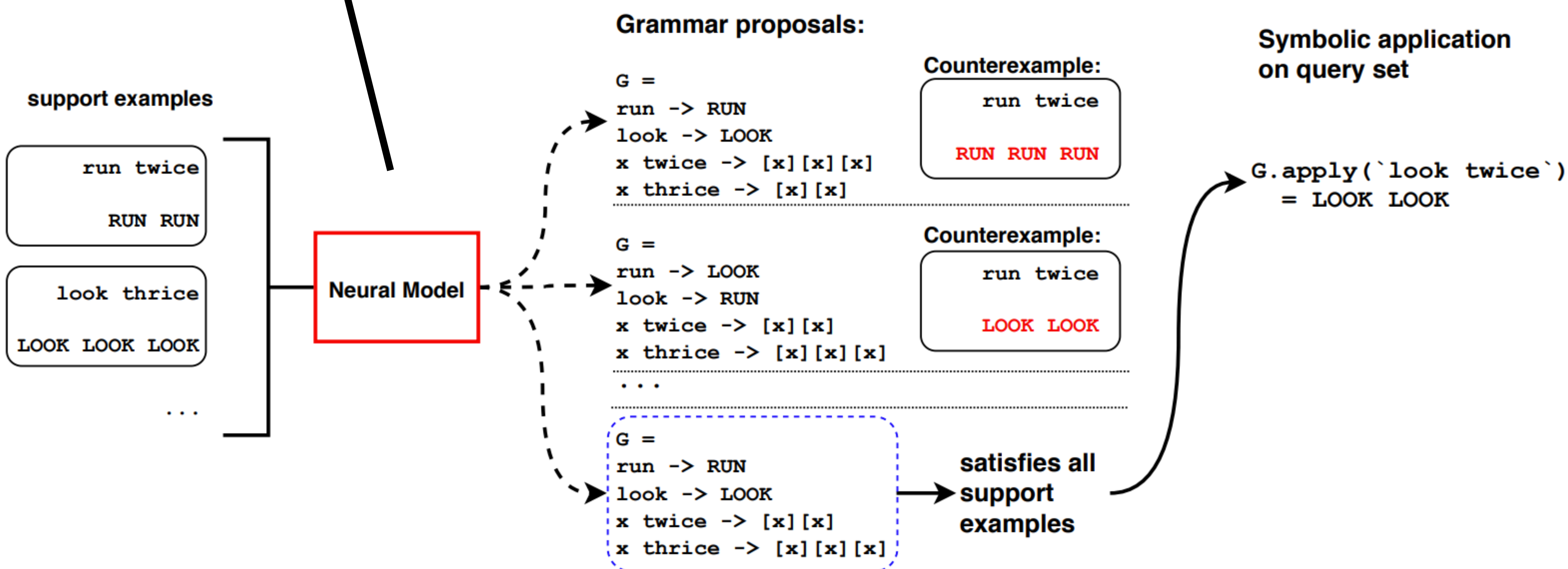


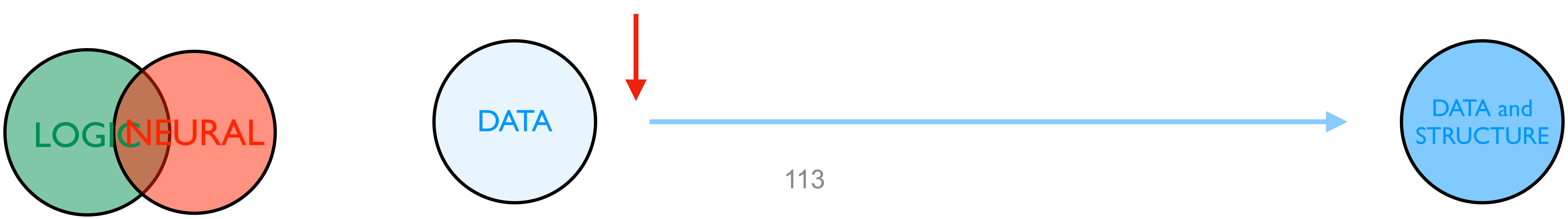


Neural Generation

[Nye et al, 2020]

Neural model generates discrete structure





Program sketching

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

Provide partial code

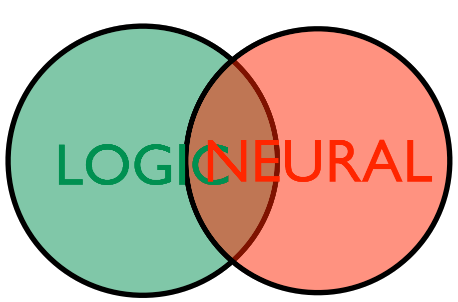
Fill in the missing functionality with neural networks

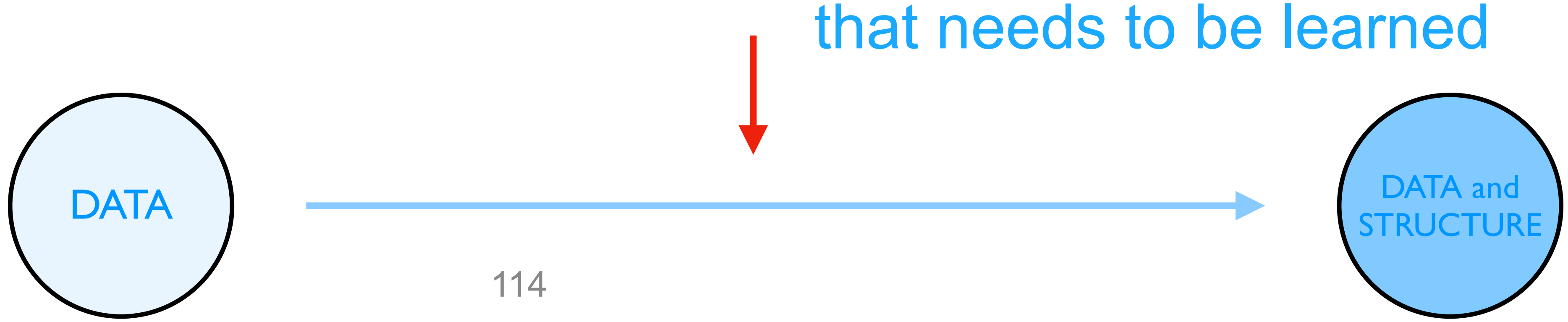
Examples:

[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



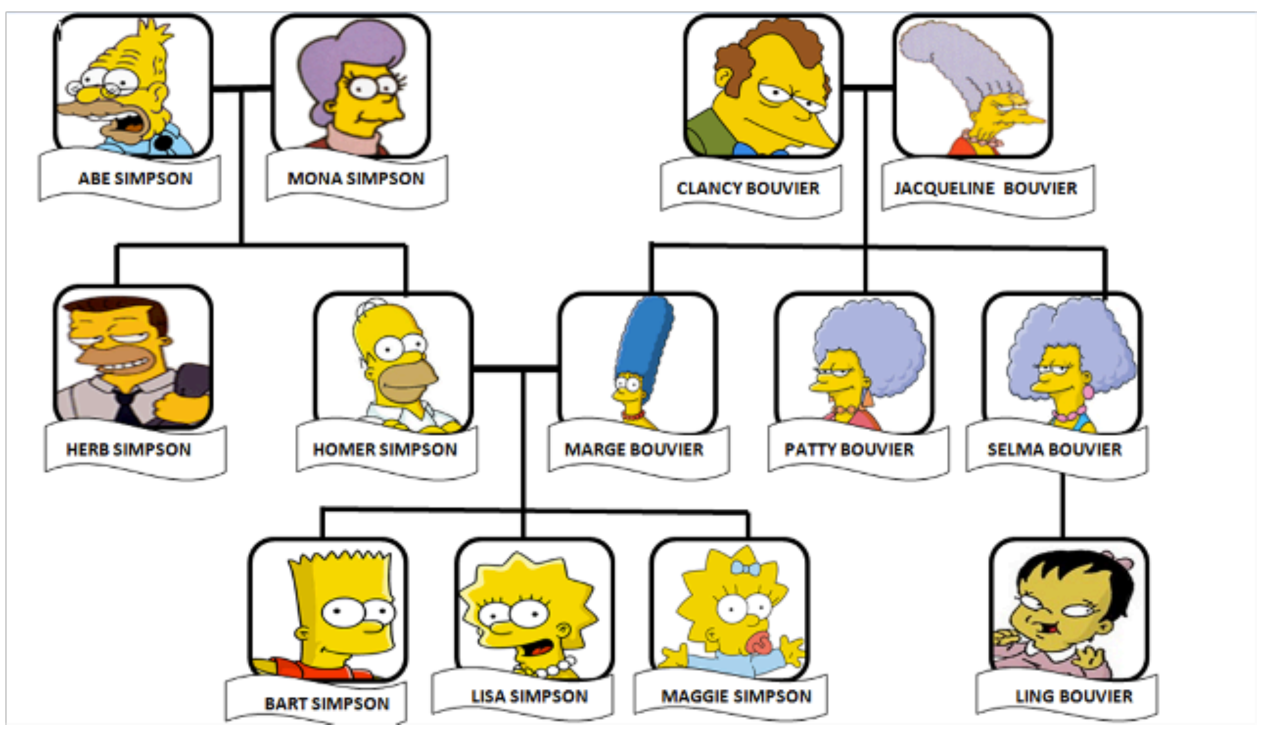


Structure learning via parameter learning

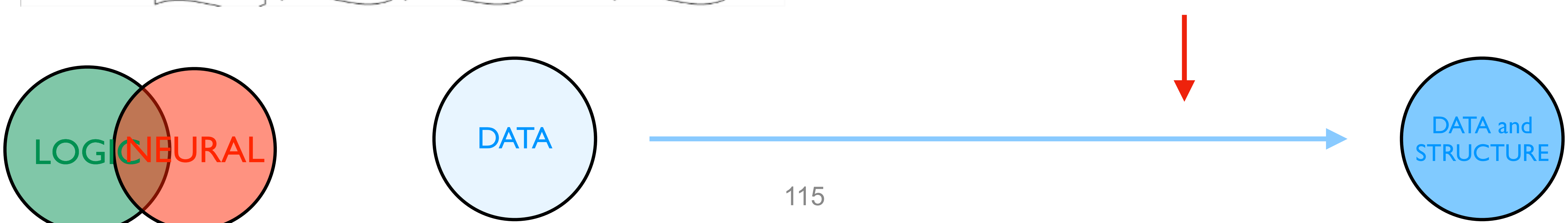
[Su et al, 2019]

Enumerate (lots of) logical formulas from templates

and learn their probabilities/weights



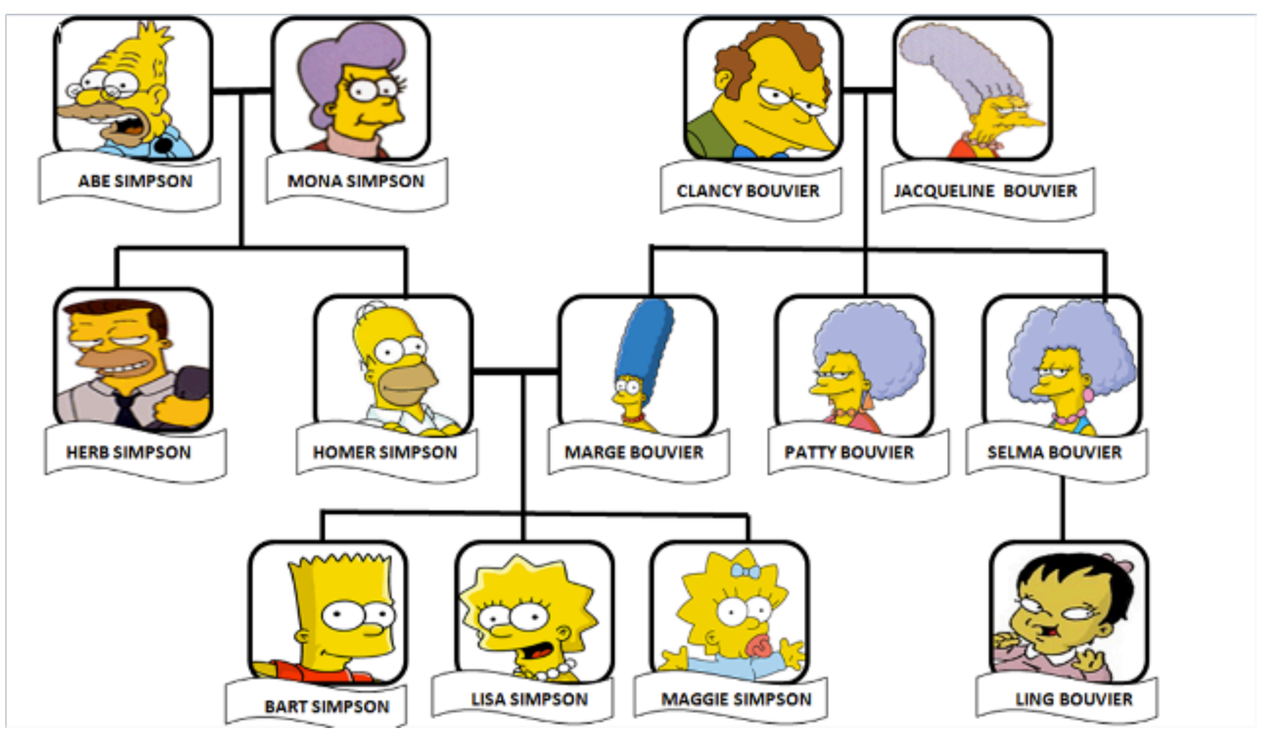
grandparent(abe,lisa).
grandparent(abe,bart).
grandparent(jacqueline,lisa).
grandparent(jacqueline,maggie.)



Program sketching

[Su et al, 2019]

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



Program templates

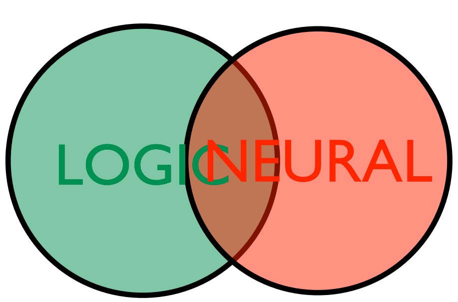
$T(X,Y) \leftarrow P(X,Y).$

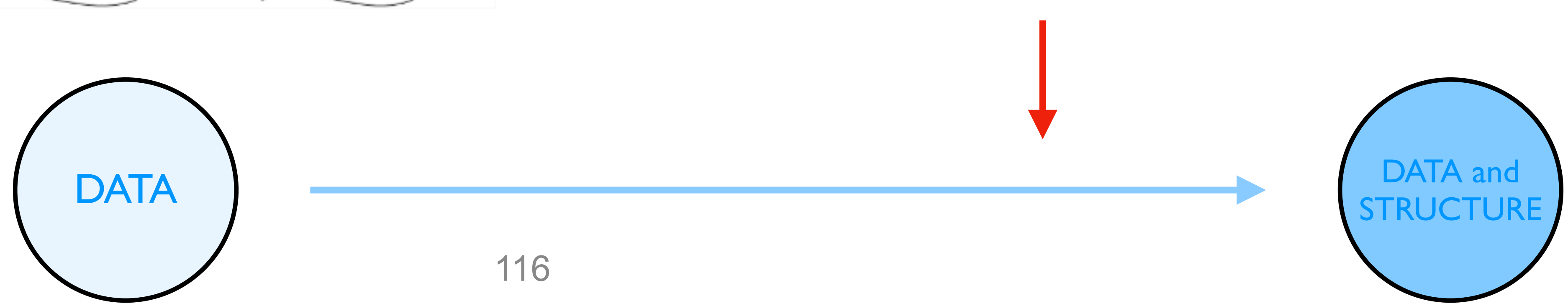
$T(X,Y) \leftarrow P(Y,X).$

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

Target: grandparent

Other predicates: father, mother





Program sketching

[Su et al, 2019]

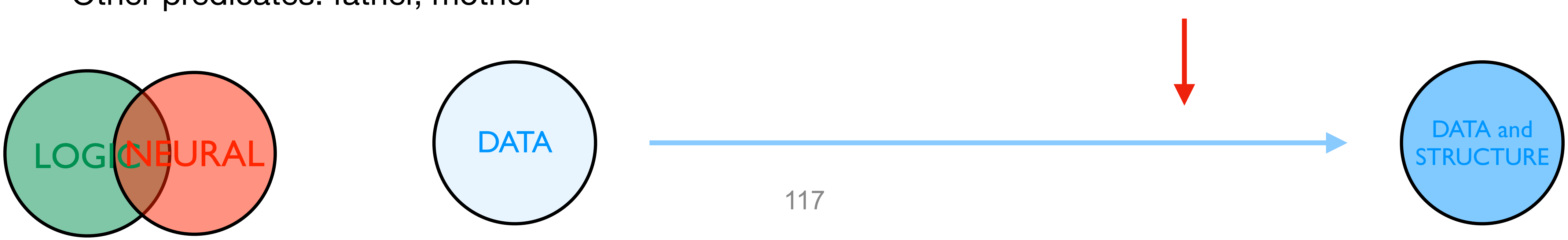
Enumerate (lots of) logical formulas from templates
and learn their probabilities/weights

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



Program sketching

[Su et al, 2019]

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

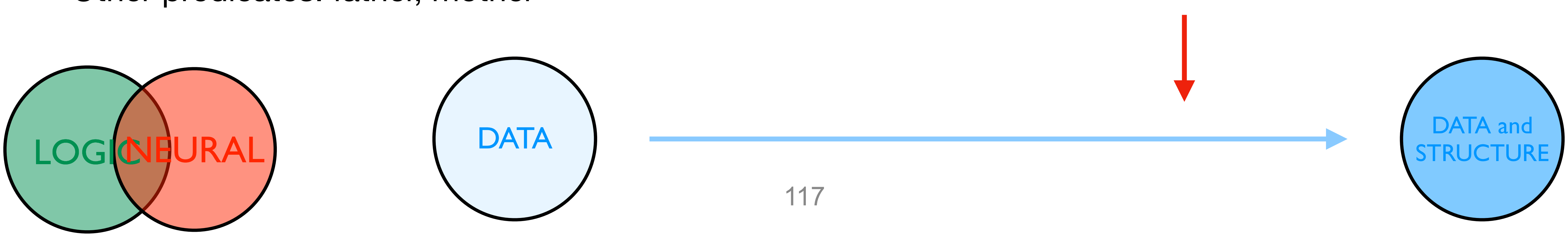
Program templates

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

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

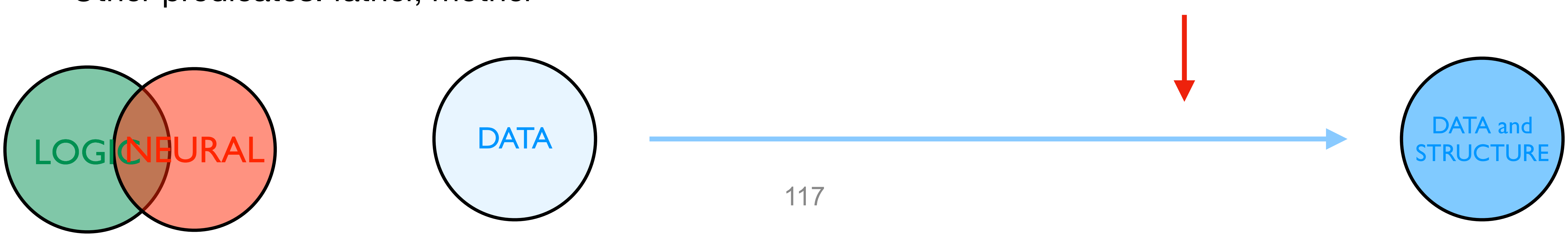
T(X,Y) ← P(X,Y).
T(X,Y) ← P(Y,X).
T(X,Y) ← P(X,Z), Q(Z,Y).

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

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

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).

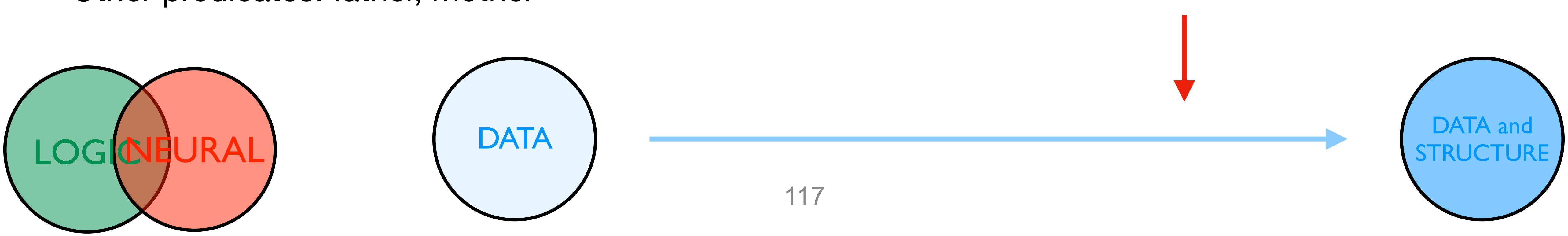
grandparent(X,Y) ← father(X,Y).
grandparent(X,Y) ← mother(X,Y).

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

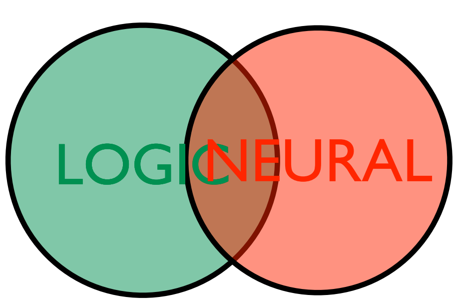
grandparent(X,Y) ← mother(X,Z), mother(Z,Y).
grandparent(X,Y) ← mother(Y,X), father(Z,Y).
……

Target: grandparent

Other predicates: father, mother



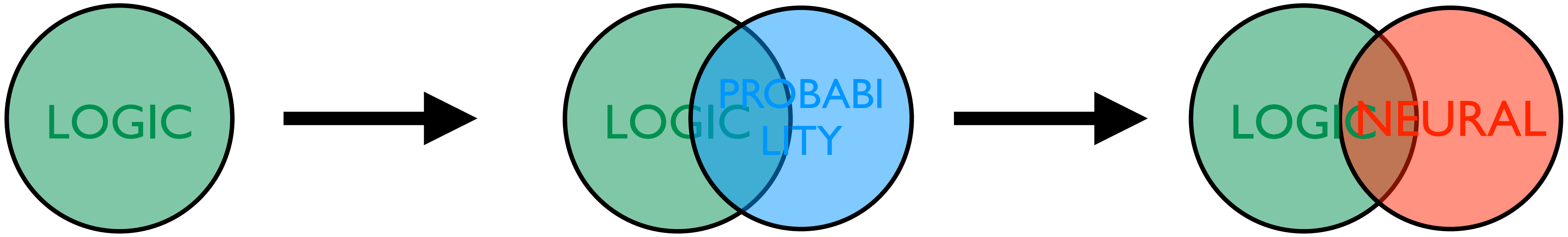
	Pros	Cons
Neural guidance	makes discrete search tractable	lots of training data
Soft patterns	efficient learning	no explicit structure
Neural generation	focused combinatorial search	lots of training data
Sketching	reduces combinatorial search	significant user effort
Structure via params	removes combinatorial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



5. Learning
Key Messages

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

6. Semantics



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.

“*human(socrates)*”

“*47(42)*”

Semantics

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

"*human(socrates)*" = ***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.**

$$\ell_S(Q) = v$$

We are interested in the algebraic (differentiability!) and computational properties of such labelling functions!

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

Semantics in Boolean Logic

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

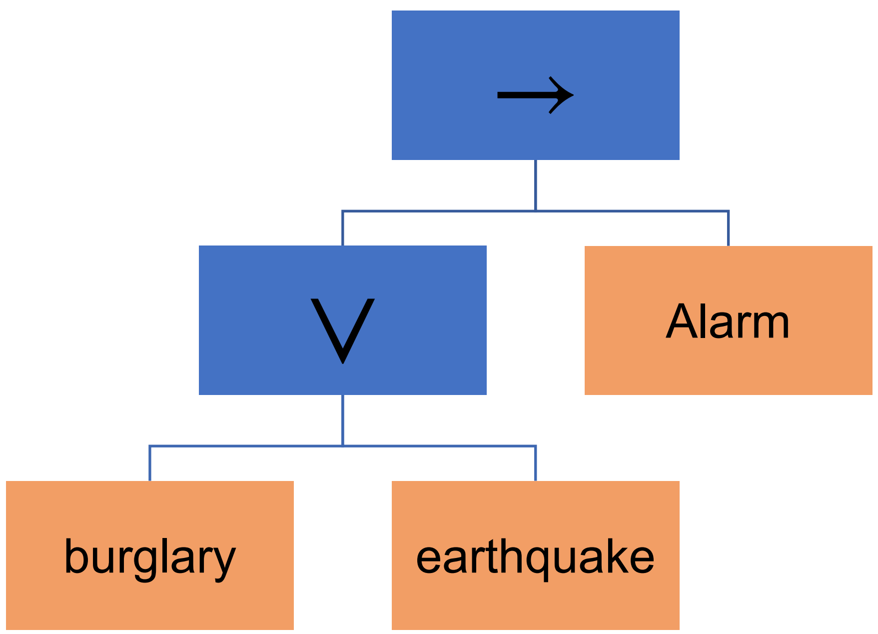
$$\ell_B(earthquake \wedge burglary) = \ell_B^{\wedge}(\ell_B(earthquake), \ell_B(burglary))$$

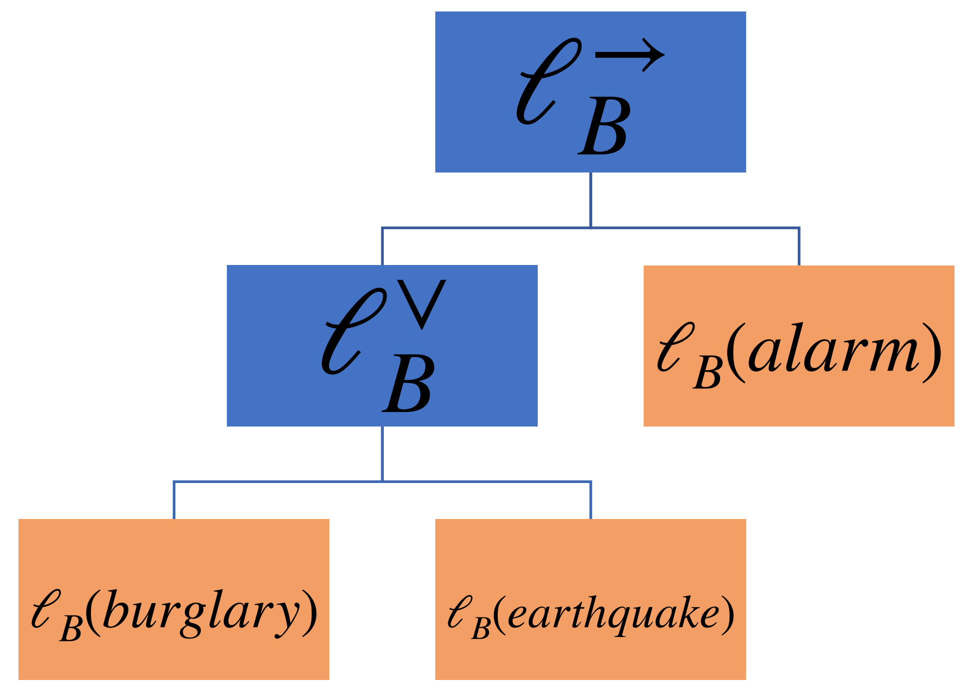
This recursive evaluation of formulas is said to be extensional approach.

LOGIC

Semantics in Boolean Logic

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





LOGIC

Semantics in Boolean Logic

- Boolean semantics is not differentiable, thus it is hard to connect to a learning component (goal of both StarAI and NeSy)
- How to solve?
  - **Alternative logic semantics -> Fuzzy Logic**
  - **Additional layer of semantics -> Probabilistic Logic**

LOGIC

6. Semantics

Fuzzy logic

LOGIC

Semantics in Fuzzy Logic

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

LOGIC

Semantics in Fuzzy Logic

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

- Fuzzy truth 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

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.

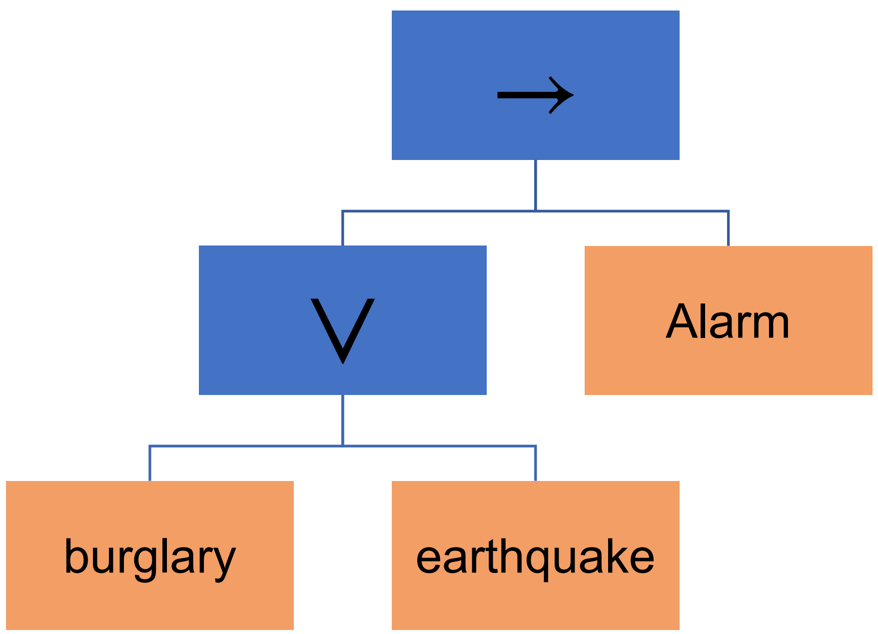
$\ell_F(burglary) = 0.9$, $\ell_F(alarm) = 0.3$,

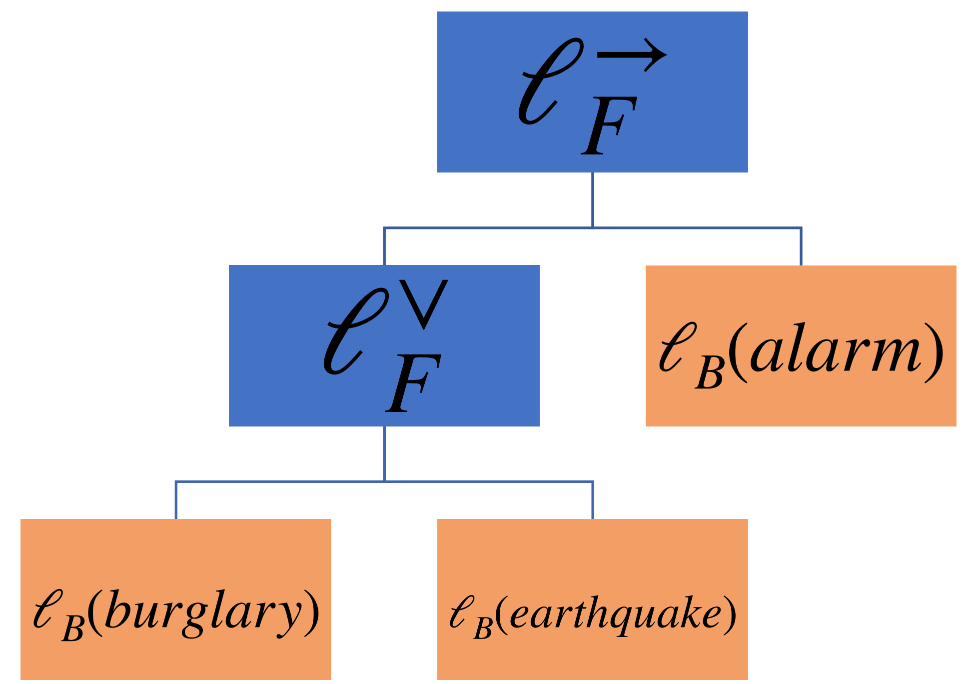
$\ell_F^{\rightarrow} = \min(1, 1 - x + y) = \min(1, 1 - 0.9 + 0,3) = 0.4$

LOGIC

Semantics in Fuzzy Logic

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





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, $\ell_F(human(Socrates)) = 0.8$????

LOGIC

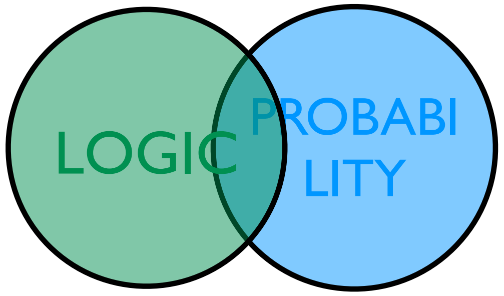
Fuzzy vs Boolean

- 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

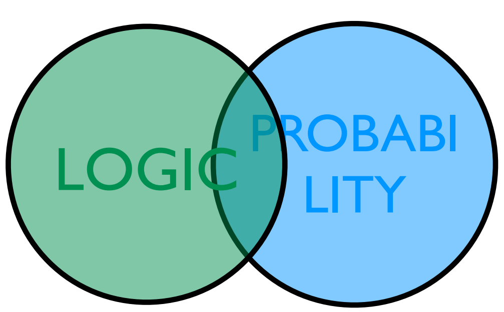
Probabilistic logic



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:

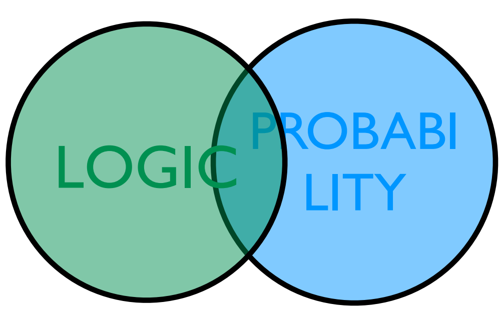
- Define a **probability distribution over interpretations / worlds (i.e. boolean semantics)**

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

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

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

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

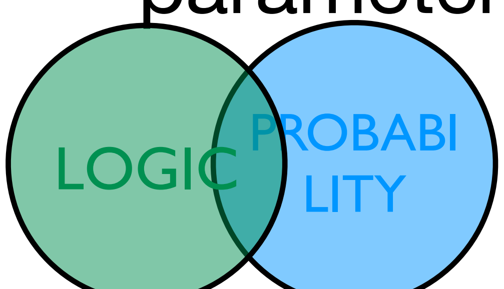


Probabilistic Logic Semantics
Problog

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)



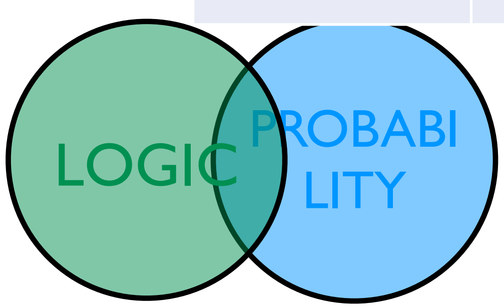
Probabilistic Logic Semantics
Problog

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

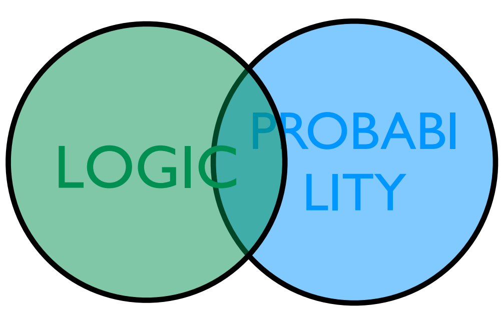
1.5 : calls(Mary) <- hears_alarm(Mary), alarm

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	$\propto \exp(0 + 2.0 + 0.5)$
…	…	…	…	…	…	

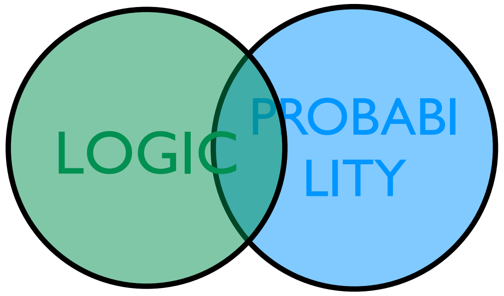
Probabilistic Logic Semantics

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

For example:

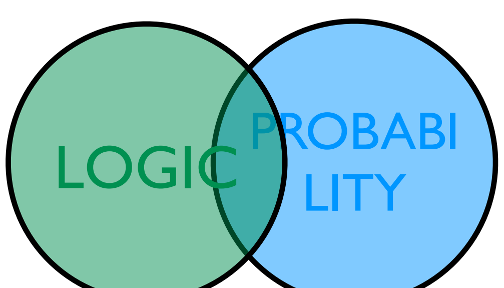
0.1 :: burglary. (B)
0.05 ::earthquake. (E)
0.6 ::hears_alarm(john). (H)
alarm :– earthquake.
alarm :– burglary.
calls(john) :- alarm, hears_alarm(john)

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

Query = burglary ^ hears_alarm(john)

$$Q = B \wedge H$$

$$p(Q) = 0.06$$

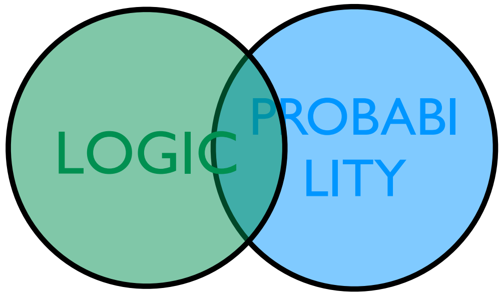


Probabilistic Logic Semantics

Probabilistic Semantics is different from a pure logic semantics

1. It is built on top of a logical semantics; $p(\ell_B(x_1), \ldots, \ell_B(x_n))$.

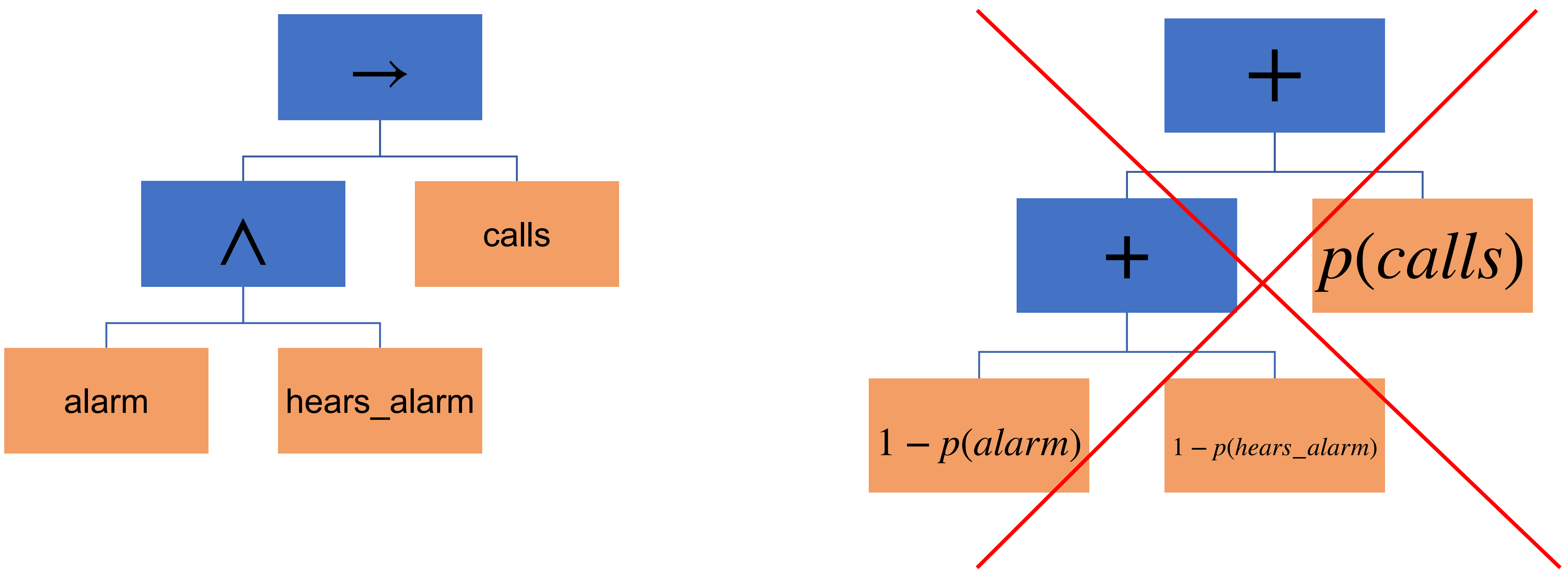
2. Probability is NOT extensional, the probability of a formula
   A. cannot be defined recursively by the probabilities of its arguments
   B. requires WMC



Probabilistic Logic Semantics

$$(alarm \wedge hears\_alarm) \rightarrow calls$$

- Consider:



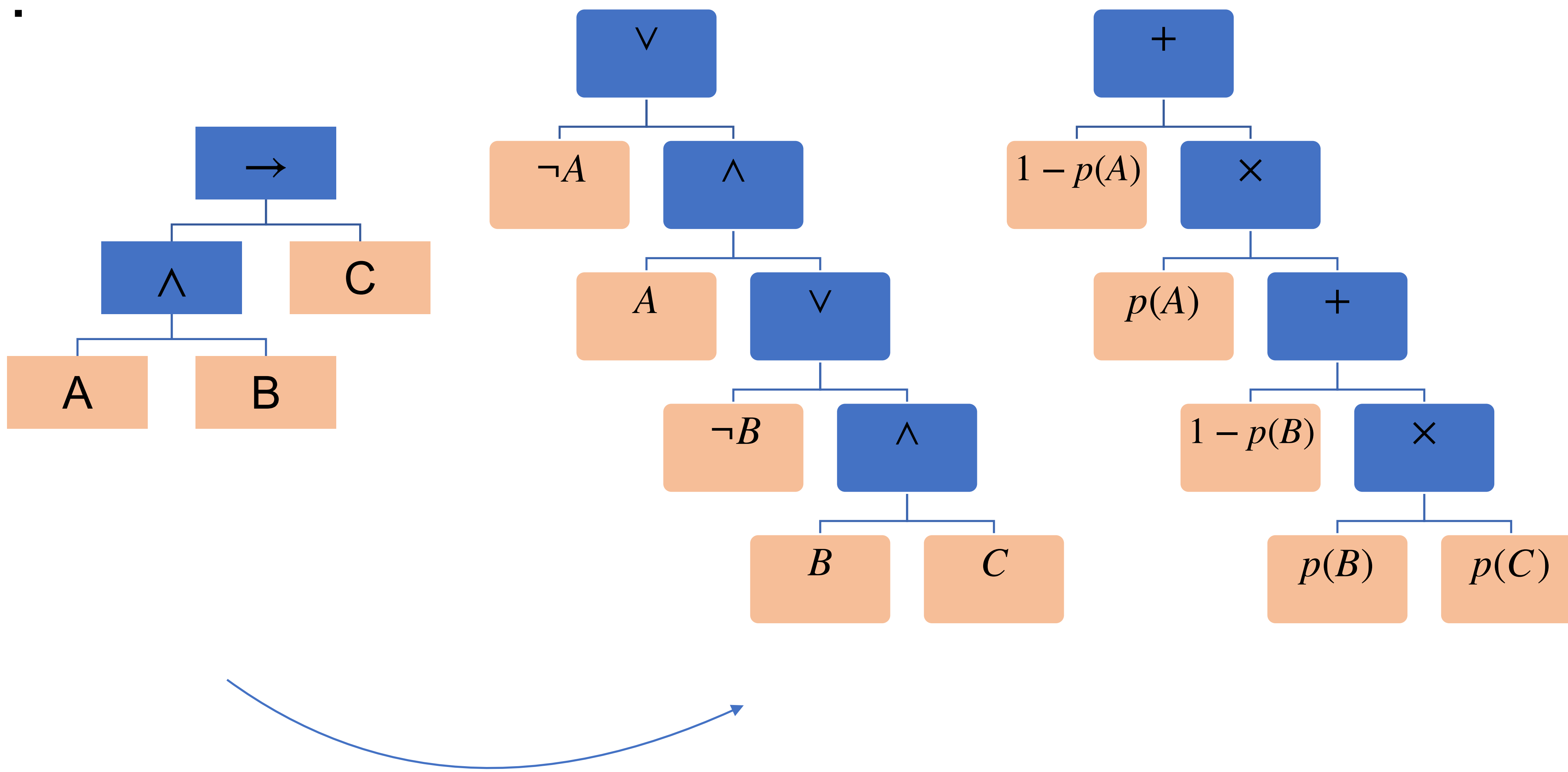
LOGIC

Probabilistic Logic Semantics

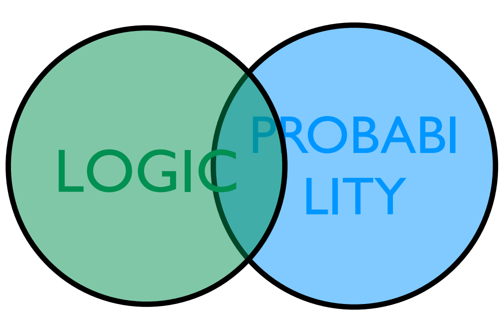
$$\ell_P(Q) = \sum_{\ell_B(x_1),\ldots,\ell_B(x_n) \models Q} p(\ell_B(x_1),\ldots,\ell_B(x_n))$$

- Consider:

$(A \wedge B) \rightarrow C$



Knowledge Compilation

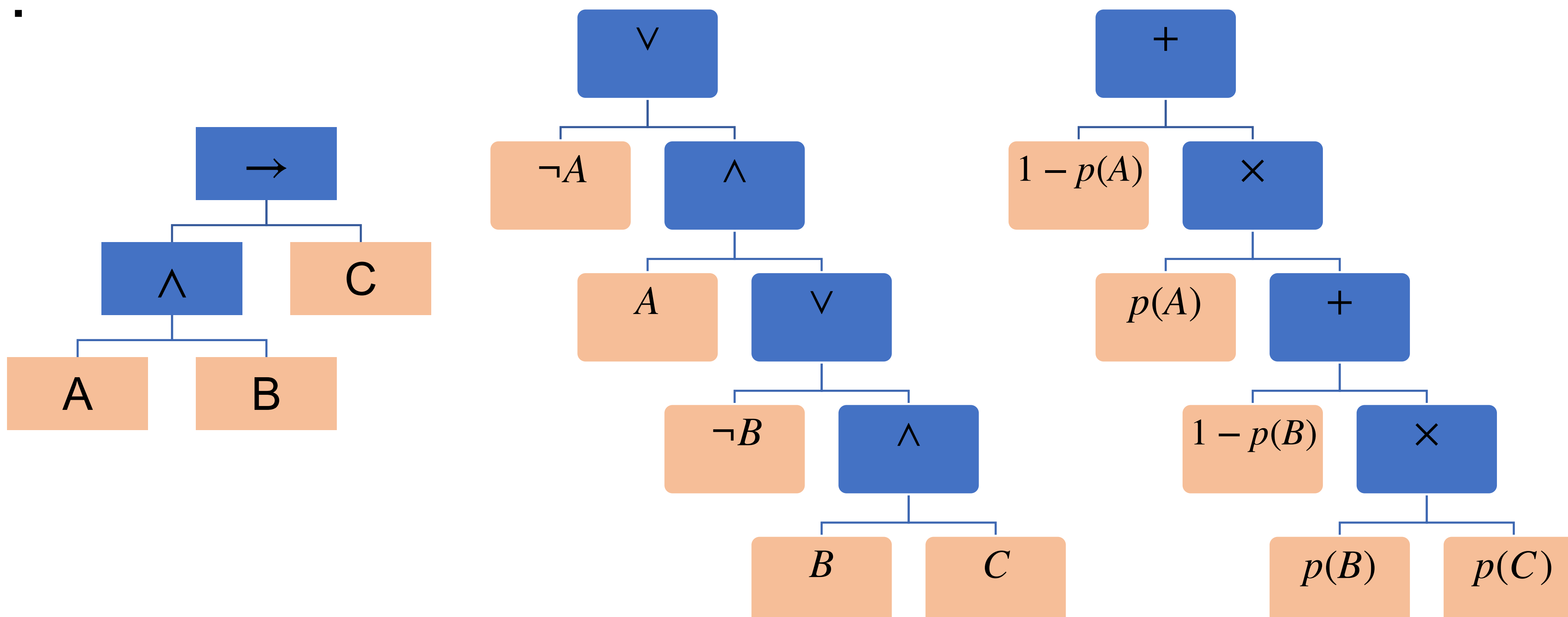


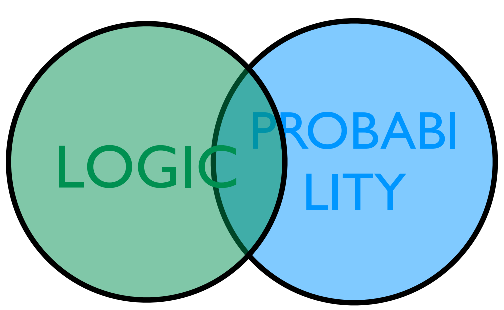
The probabilistic structure is now explicit in the compiled formula.

Probabilistic Logic Semantics

- Consider:

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





The circuit is differentiable!

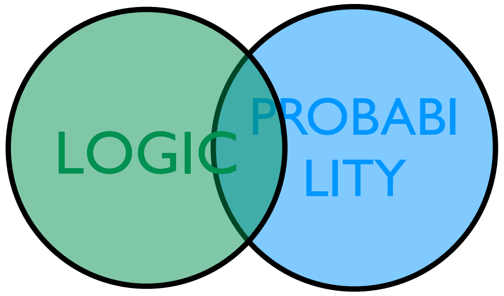
Probabilistic Logic Semantics

- 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:

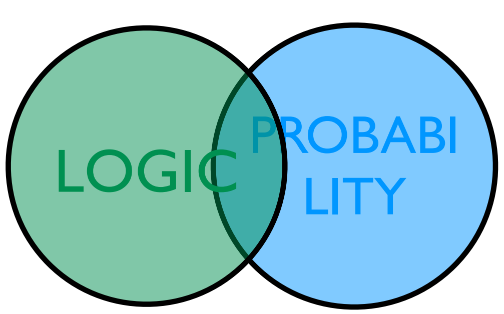
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \frac{1}{Z}\exp\left(\sum_{\alpha} w_\alpha \, \ell_B(\alpha)\right)$$

- 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

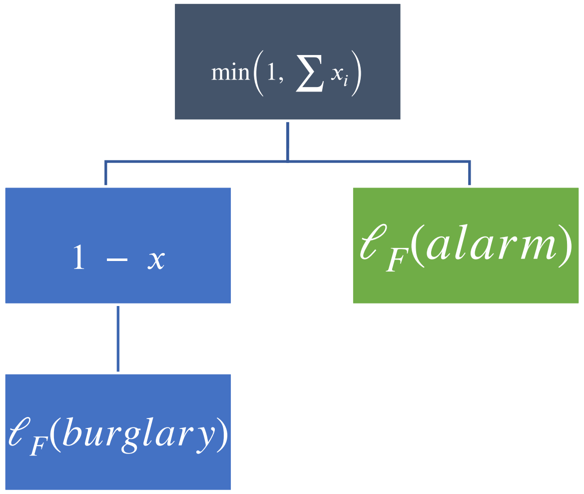
Each formula contributes with a value in [0,1]



Probabilistic Soft Logic (PSL)

$\alpha : burglary \rightarrow alarm$

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

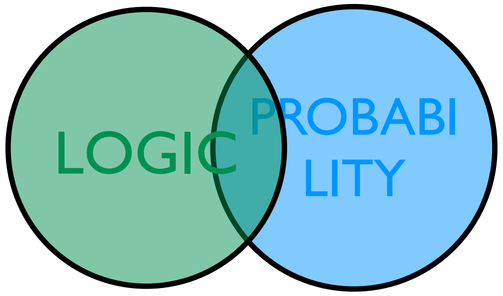


MPE:

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

This is soft SAT using fuzzy logic

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

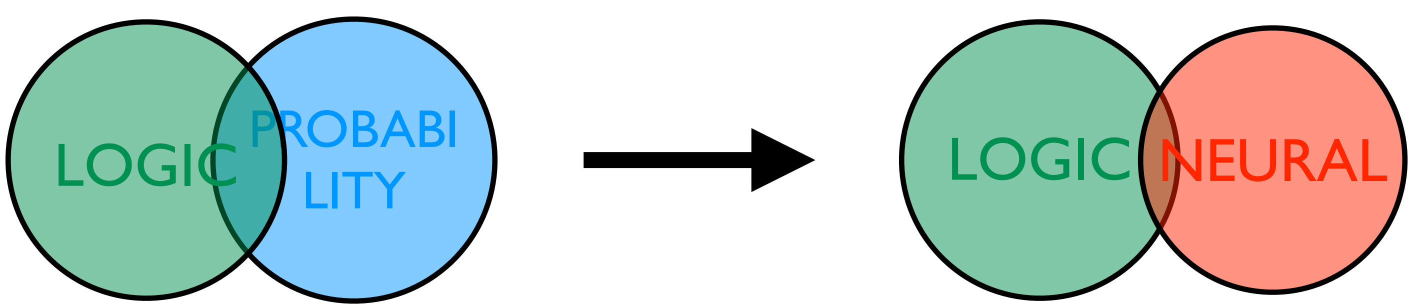


Probabilistic vs Fuzzy

- 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 $A = B = C = D = E = 0.2$)

6. Semantics

Neural Symbolic



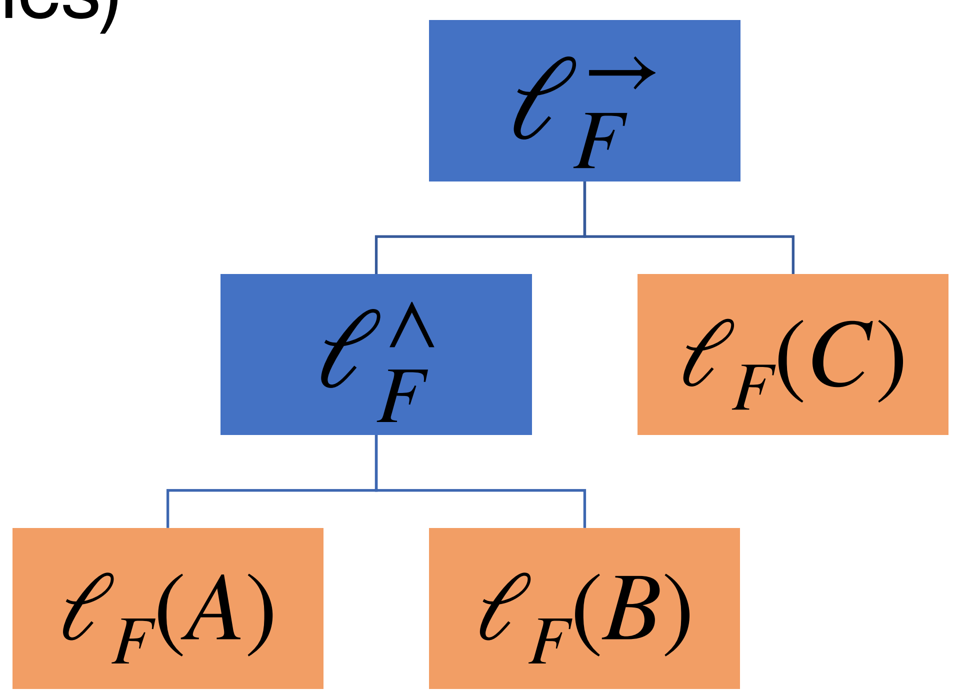
Neural Symbolic

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

$$\ell(Q)$$

Labelling functions (semantics) = Parametric circuit

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



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

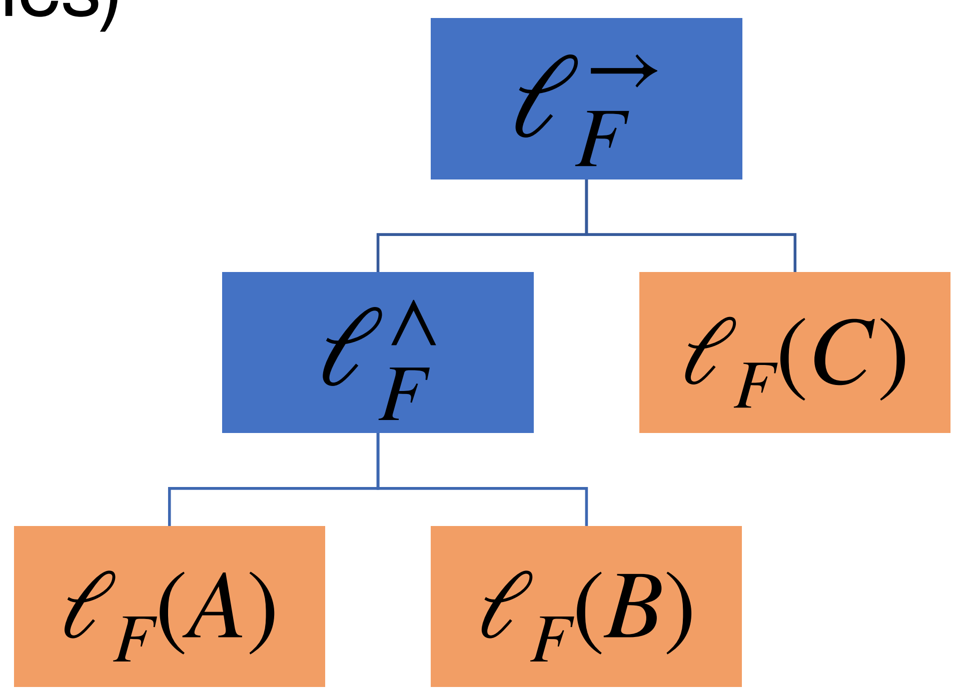
Neural Symbolic

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

$$\ell(Q)$$

Labelling functions (semantics) = Parametric circuit

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



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

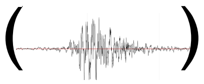
```
0.1 :: burglary.   (B)
0.05 ::earthquake. (E)
0.6 ::hears_alarm(john).  (H)
alarm :– earthquake.
alarm :– burglary.
```

L	p
Burglary	0.1
Earthquake	0.05
…	

Neural Symbolic

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

-

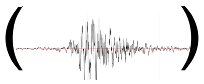
? :: burglary()
? ::earthquake. ()
? ::hears_alarm(john).
alarm :– earthquake.
alarm :– burglary.

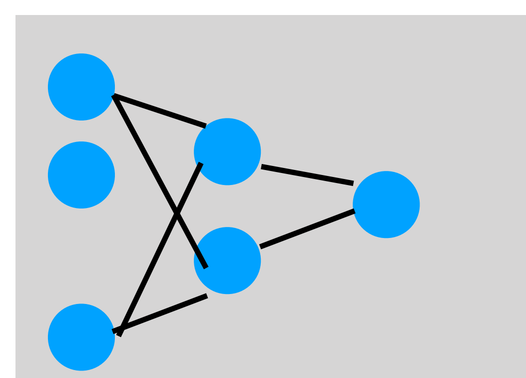
?

Neural Symbolic

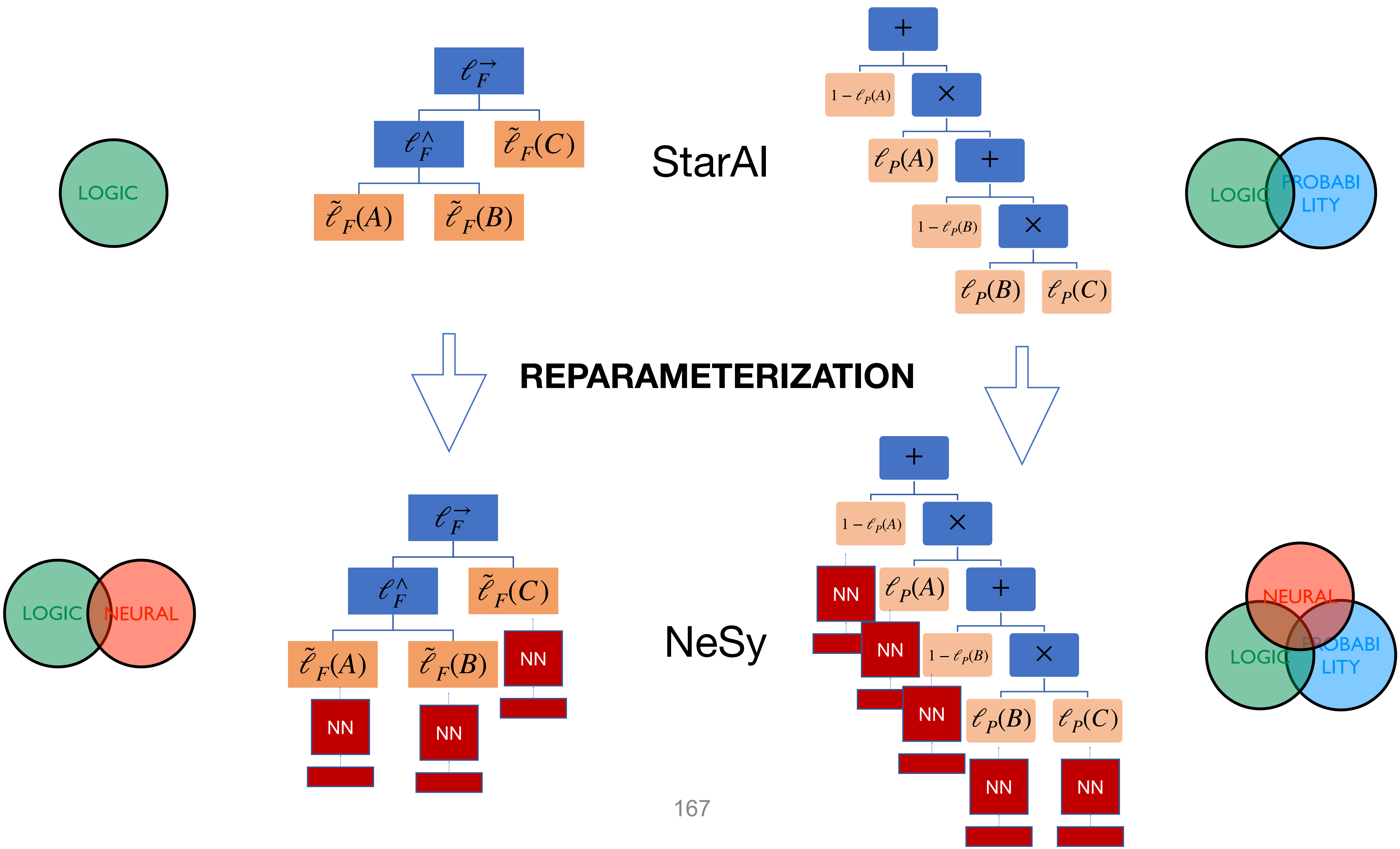
How to carry over concepts from the semantics of StarAI to neural symbolic?

- What if atomic labels are just neural networks?

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

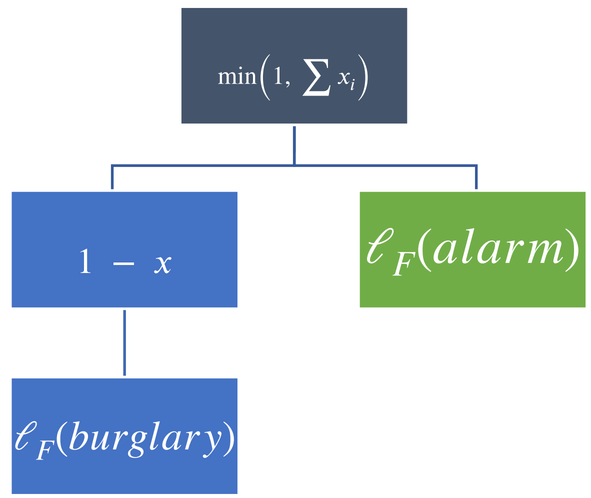


StarAI to Neural Symbolic



Fuzzy Reparameterization

$\alpha : burglary \rightarrow alarm$

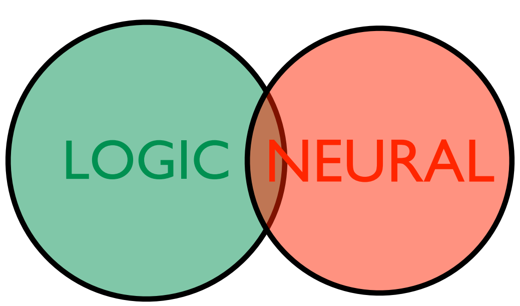


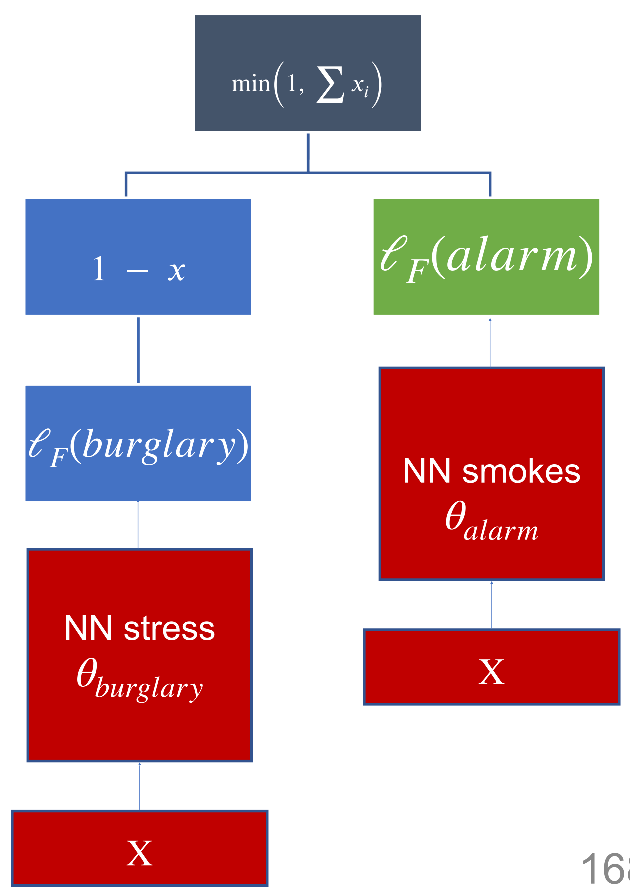
StarAI (PSL)

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

Semantic Based Regularization (Diligenti et al, AI 2017)

Logic Tensor Network (Donadello et at, IJCAI 2017)





NeSy (SBR, LTN, DLM)

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

Parameters of the neural nets

Probabilistic Reparameterization

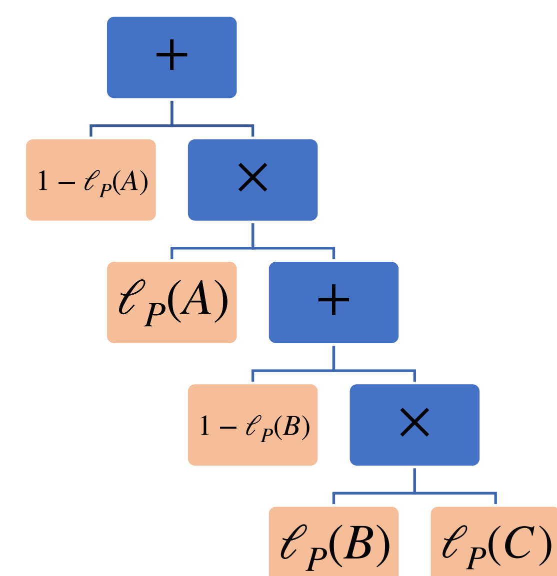
■ Probabilistic parameters

- ProbLog:

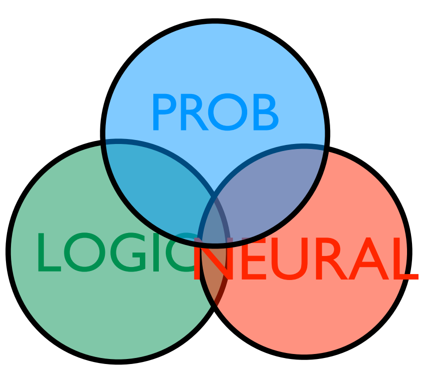
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \prod_{i:\ell_B(x_i)=True} p(x_i) \prod_{i:\ell_B(x_i)=False} (1 - p(x_i))$$

- 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

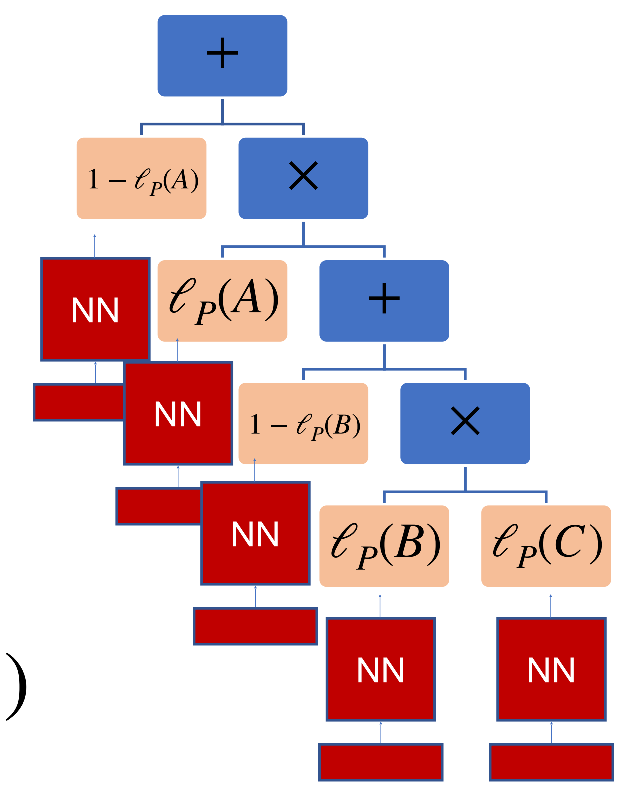
Neural parameters

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

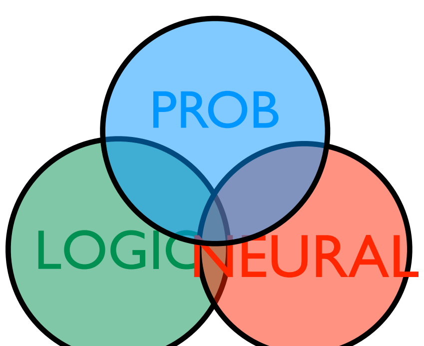
$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \prod_{i:\ell_B(x_i)=True} p(x_i) \prod_{i:\ell_B(x_i)=False} (1-p(x_i))$$

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

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



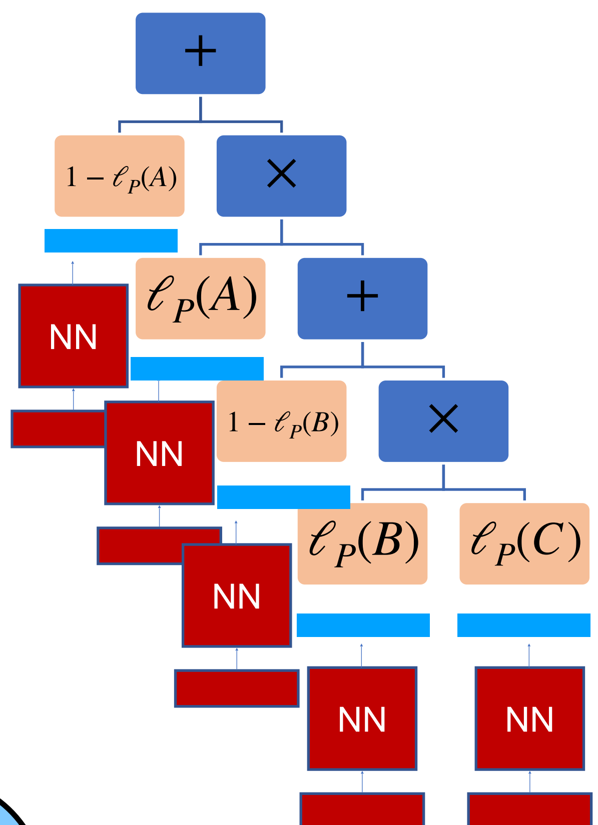
WMC



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

Probabilistic Reparameterization

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



Probabilistic fact

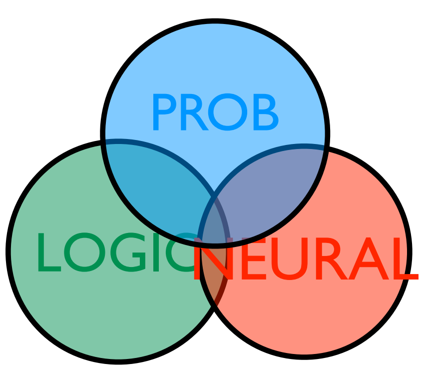
0.01 :: burglary.

↓

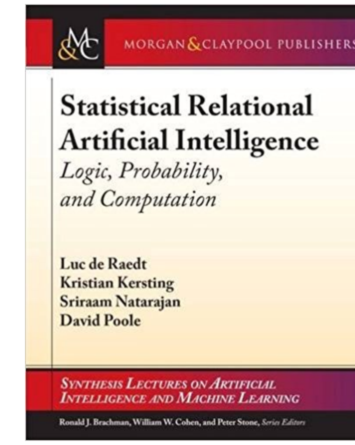
Neural Predicate

Interface ←

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

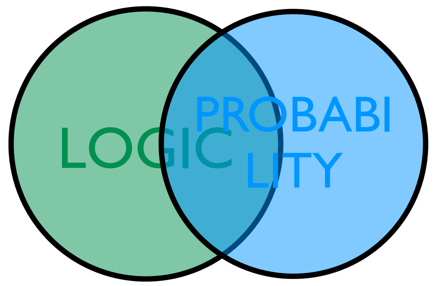


Conclusions

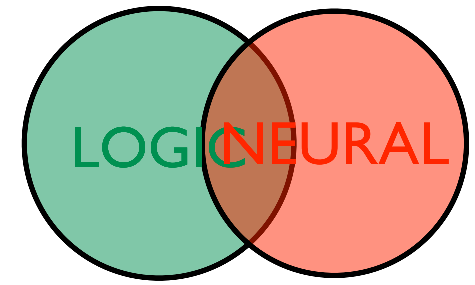


Key Message

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

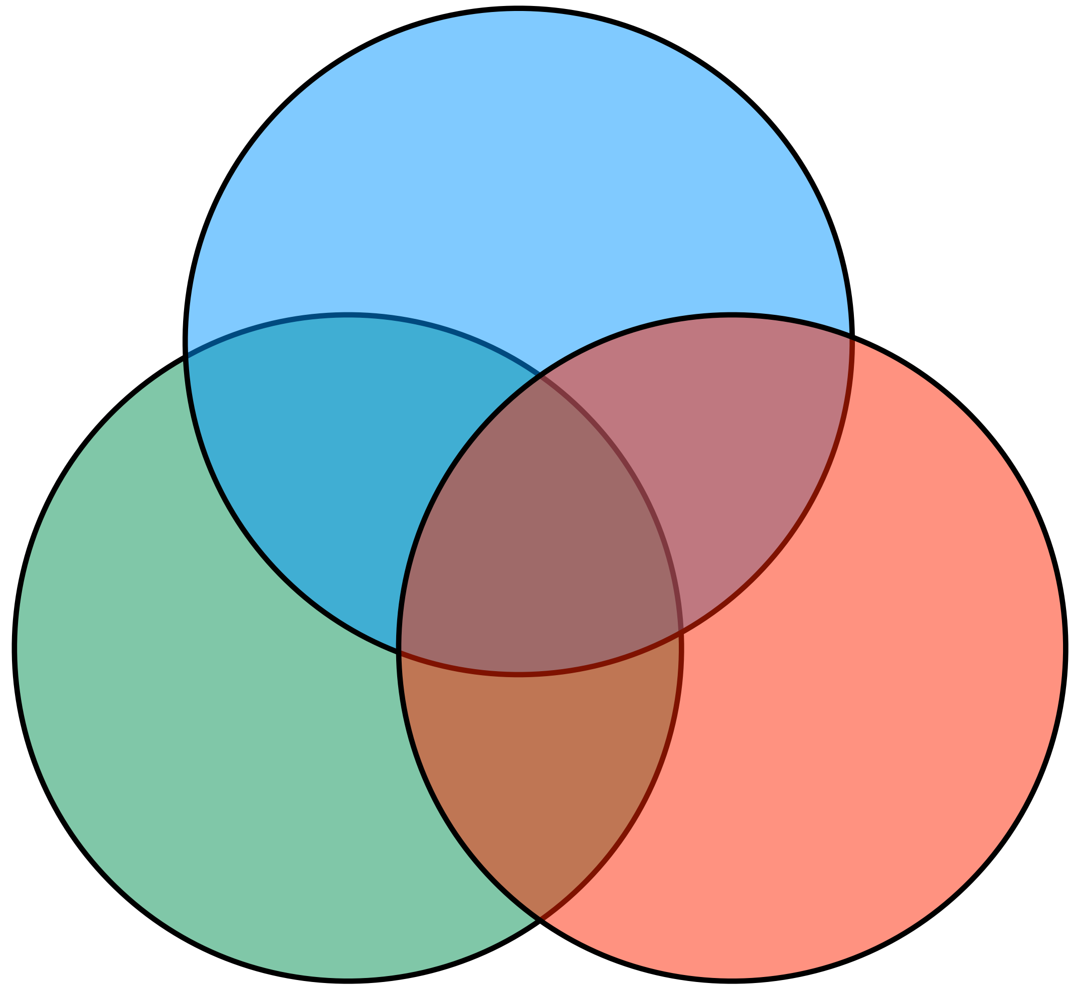
- What properties should integrated representations satisfy
  - Should one representation take over ?
    - (As in most approaches to NeSy — push the logic inside and forget about it afterwards)
    - Should one have the originals as a special case ?
  - Should one build a pipeline (e.g. first neural then logic) or a bi-directional interface between the integrated representations?
  - Can neural and logic features be intermixed more closely?

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 ?

Challenges

- For NeSy,
  - Better understanding
  - 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

From Statistical Relational to Neuro-Symbolic Artificial Intelligence
Luc de Raedt, Sebastijan Dumančić, Robin Manhaeve, Giuseppe Marra
(https://www.ijcai.org/Proceedings/2020/688)

(long-version) From Statistical Relational to Neural-Symbolic Artificial Intelligence: a Survey.
Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, Luc De Raedt
(https://arxiv.org/pdf/2108.11451.pdf)

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 Dumanˇci´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