Probabilistic (Logic) Programming and Statistical Relational Learning Part II

Luc De Raedt and Angelika Kimmig

Also many slides from Guy Van den Broeck and others





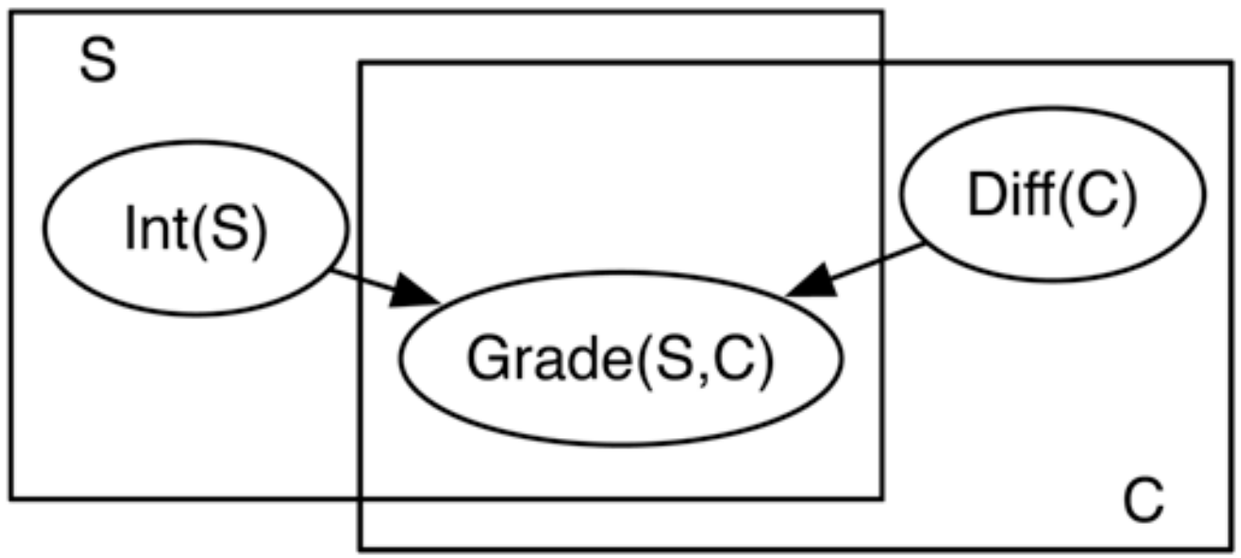


RECAP in 6 slides

Why are we doing this ?

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

Flexible and Compact Relational Model for Predicting Grades



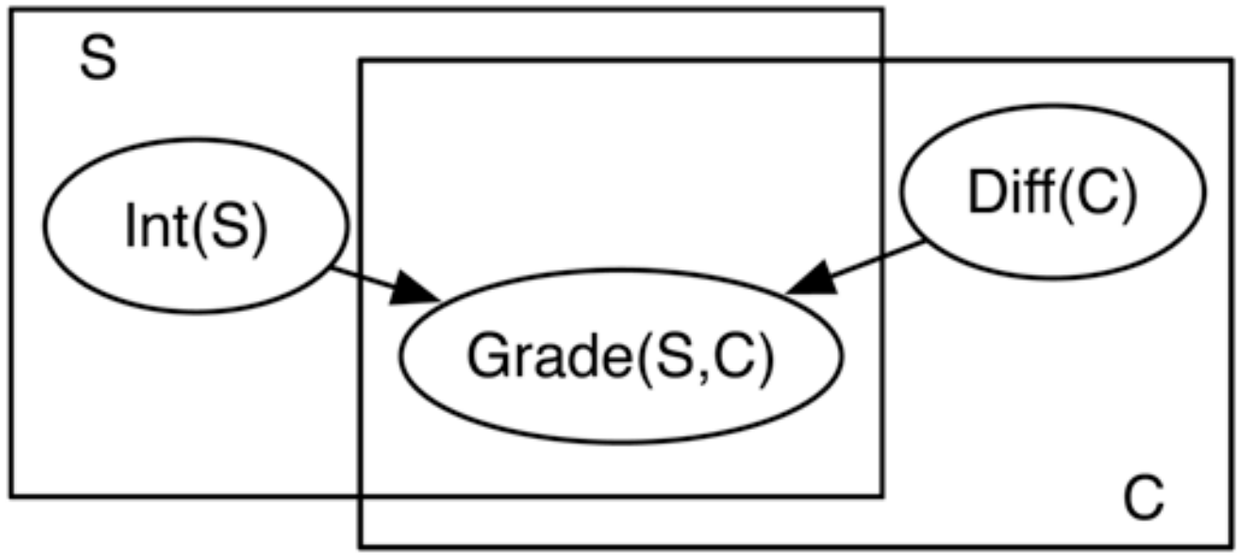
“Program” Abstraction:

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

Grounding:

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

ProbLog by example: Grading



Shows relational structure

grounded model: replace variables by constants

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

With SRL / PP

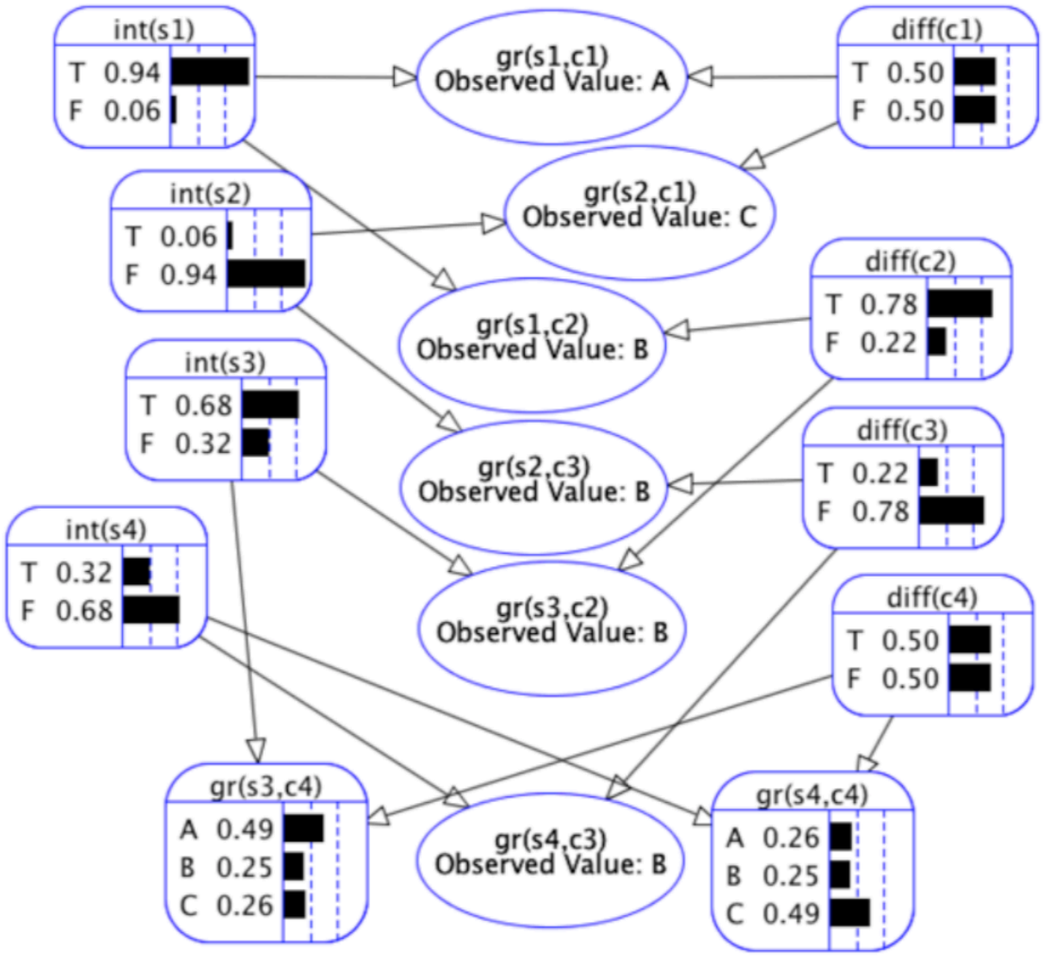
build and learn compact models,

from one set of individuals - > other sets;

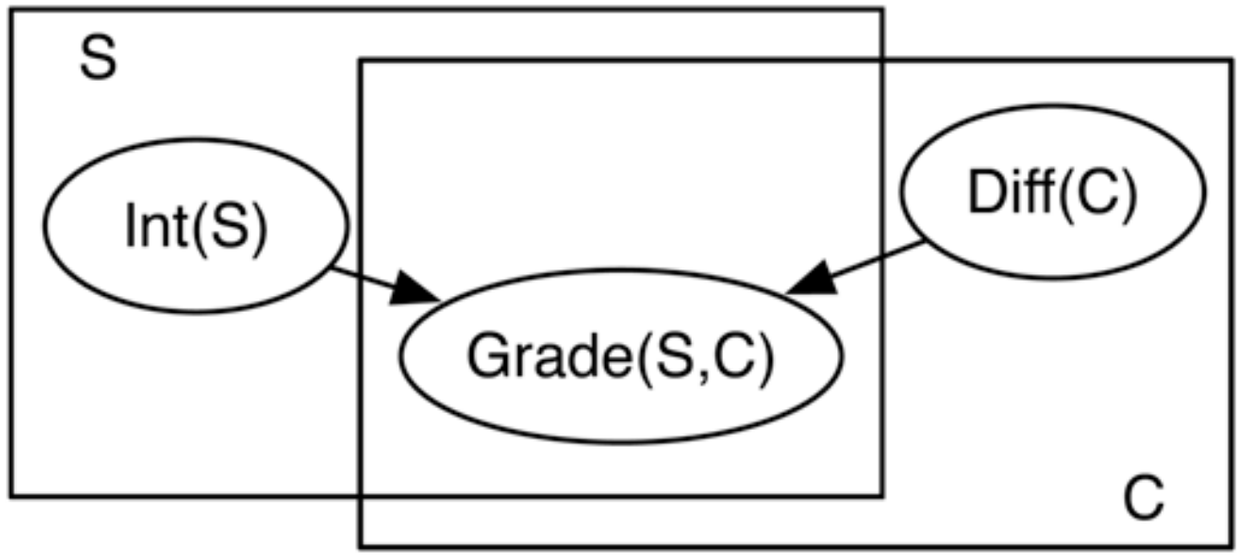
reason also about exchangeability,

build even more complex models,

incorporate background knowledge



ProbLog by example: Grading



Shows relational structure

grounded model: replace variables by constants

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

With SRL / PP

build and learn compact models,

from one set of individuals - > other sets;

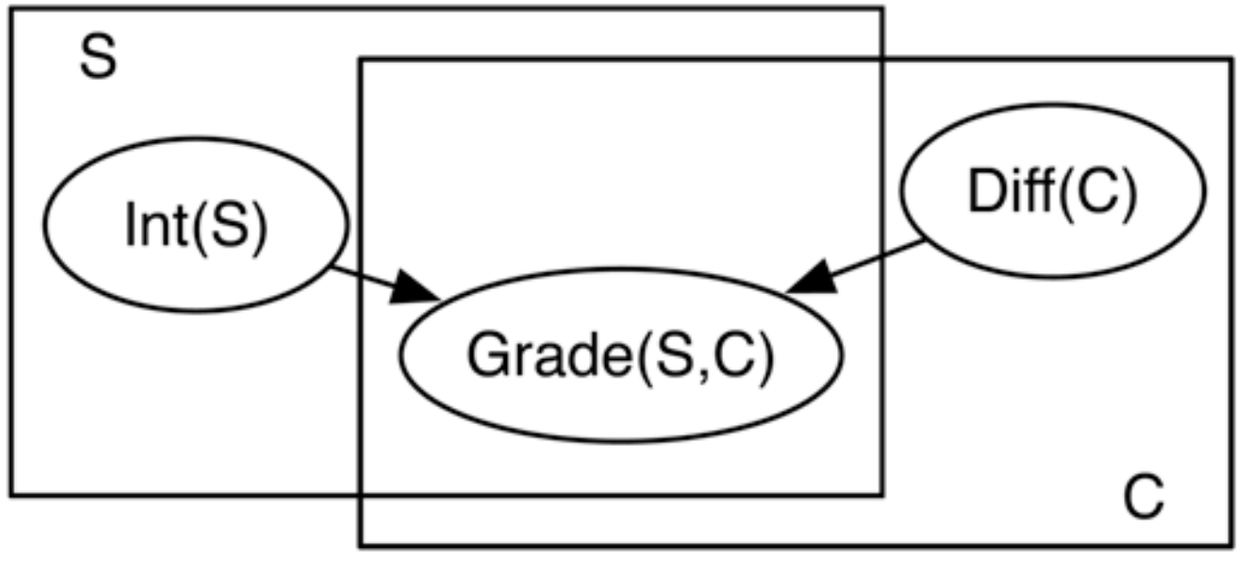
reason also about exchangeability,

build even more complex models,

incorporate background knowledge

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

ProbLog by example: Grading




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

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

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

Inference: Two Steps

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

Why are we doing this ?

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

Roadmap

- Prob. Programming - Modeling
- Inference
- Learning
- Dynamics
- Logic, Probability and Neural Nets
- KBMC & Markov Logic
- Lifted Inference

... with some detours on the way

Approximate Inference

- Standard techniques apply
  - rejection, importance, likelihood weighting, MCMC.
- Can also get logical approximations
  - Lower and upper bounds

$$\phi_L \models \phi \models \phi_U$$

$$P(\phi_L) \leq P(\phi) \leq P(\phi_U)$$

Part III : Learning

Part III : Learning

a. Parameters

Parameter Learning

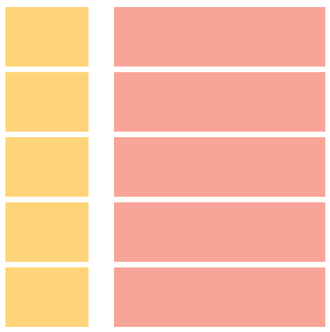
e.g., webpage classification model

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

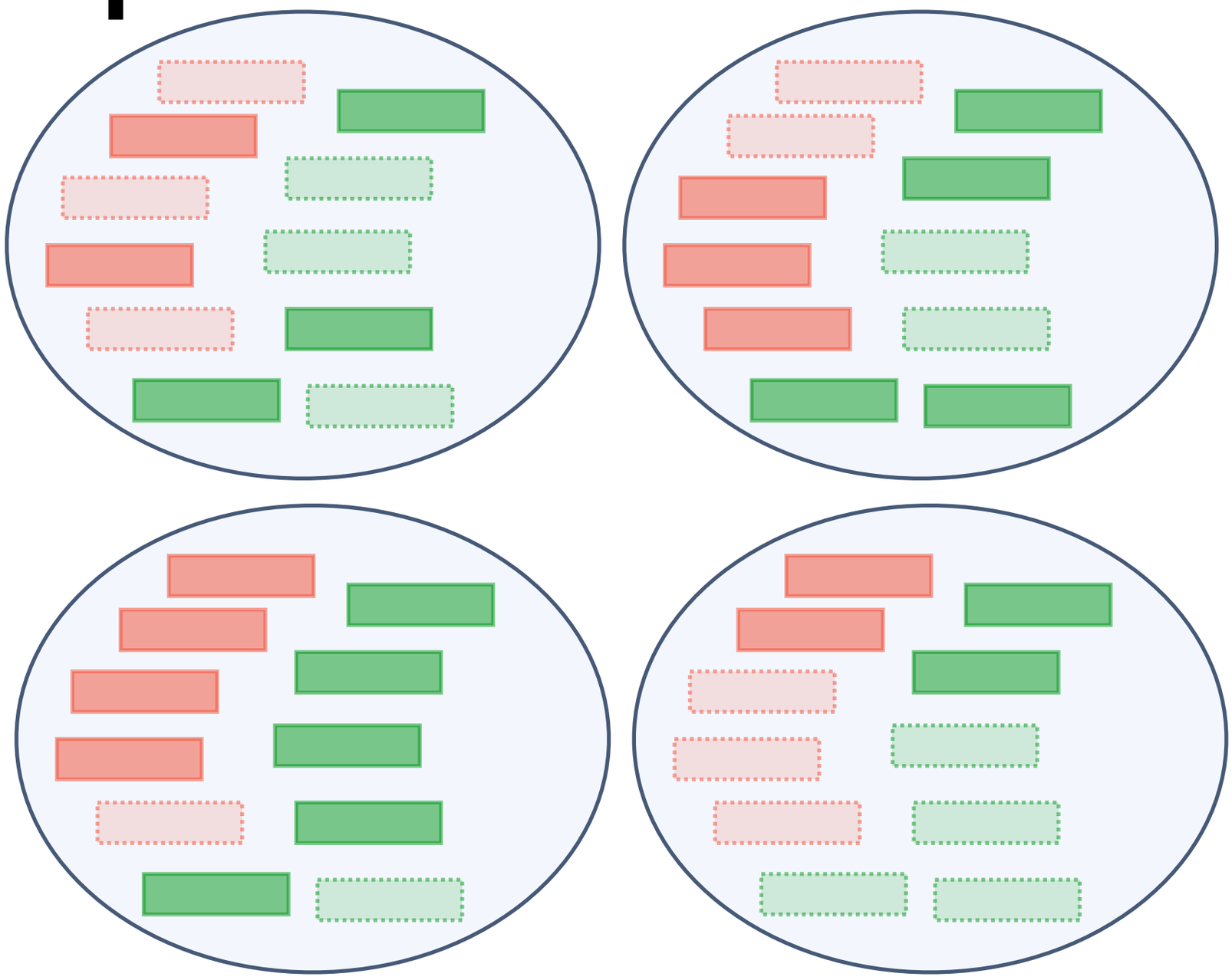
?? :: link_class(Source,Target,***CLASS1***,***CLASS2***).
?? :: word_class(***WORD***,***CLASS***).

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

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

Sampling Interpretations

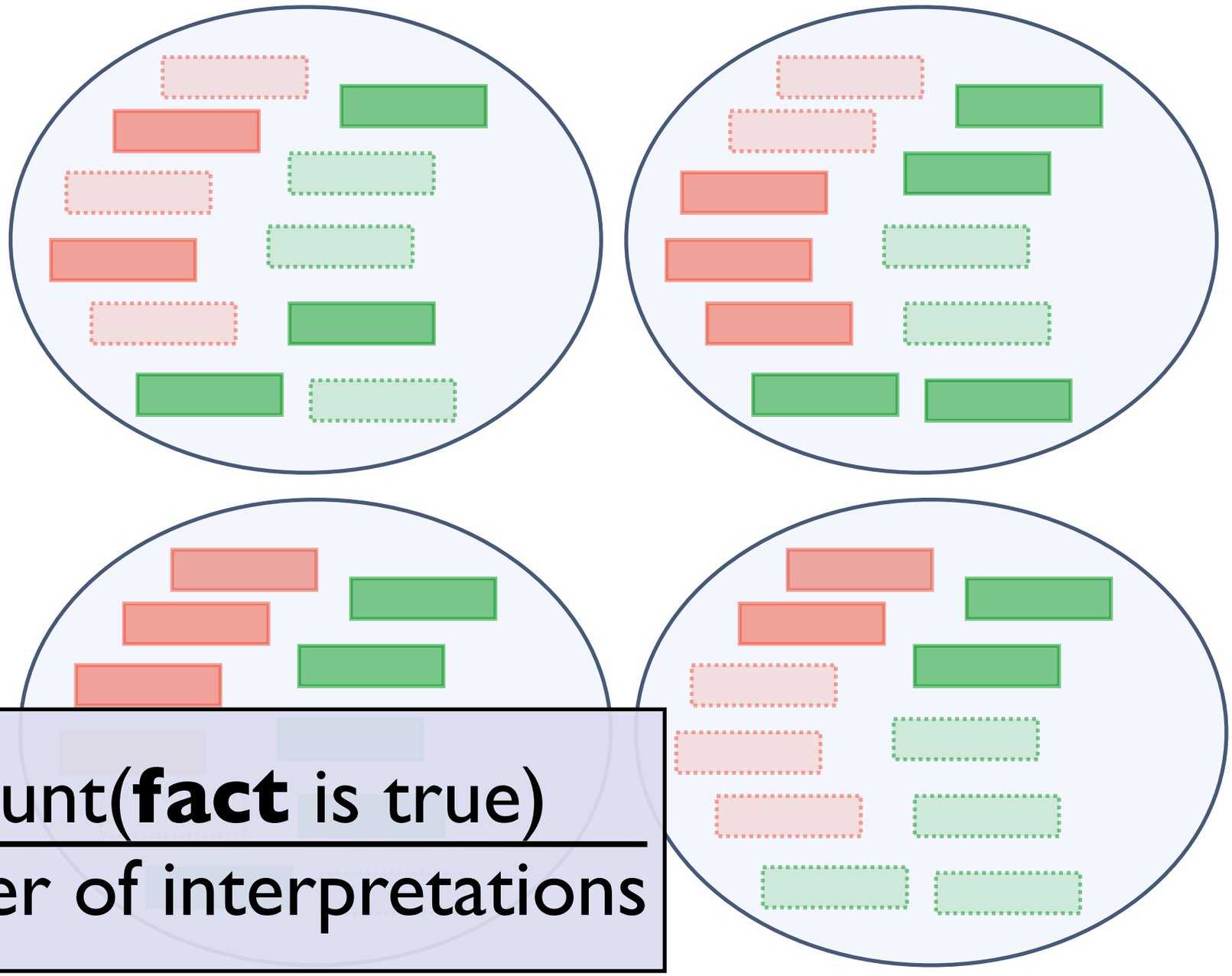






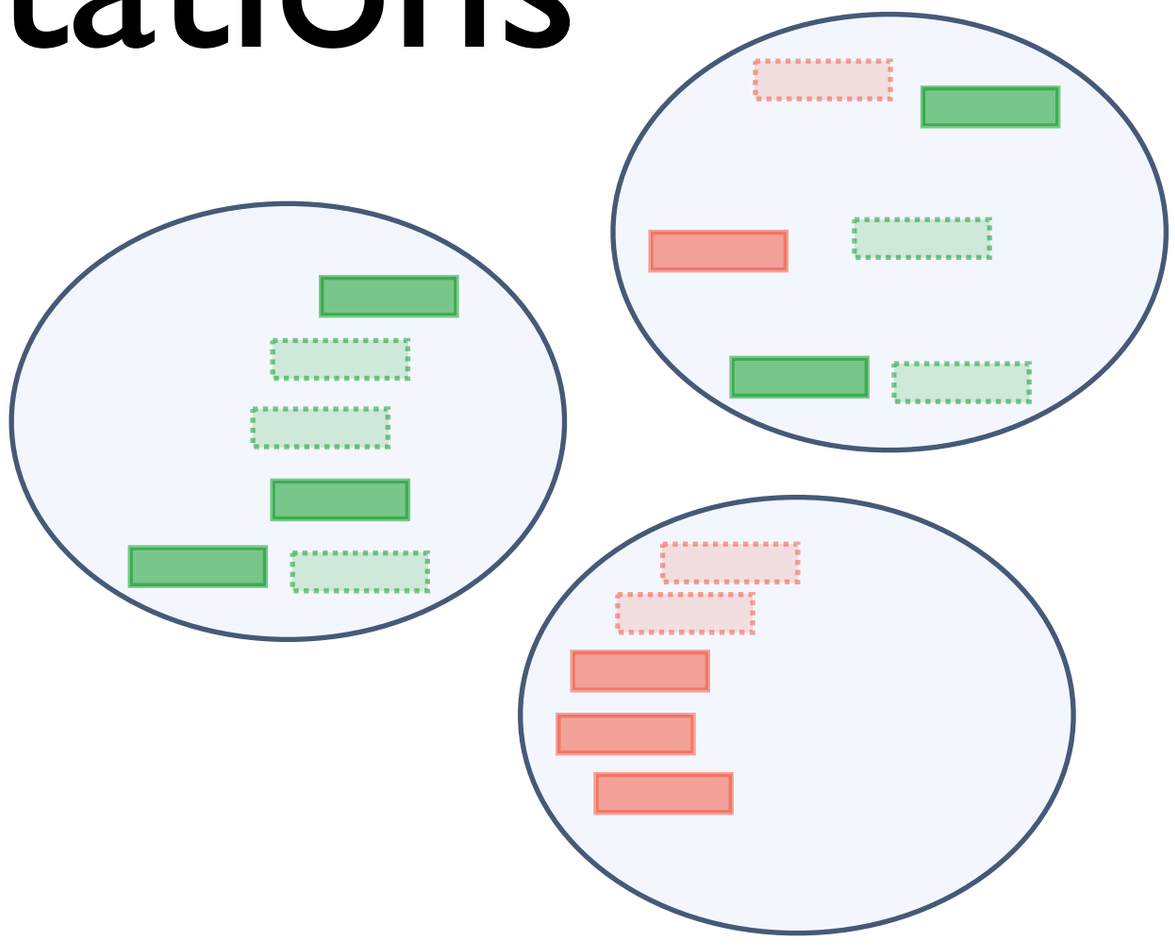
Parameter Estimation





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

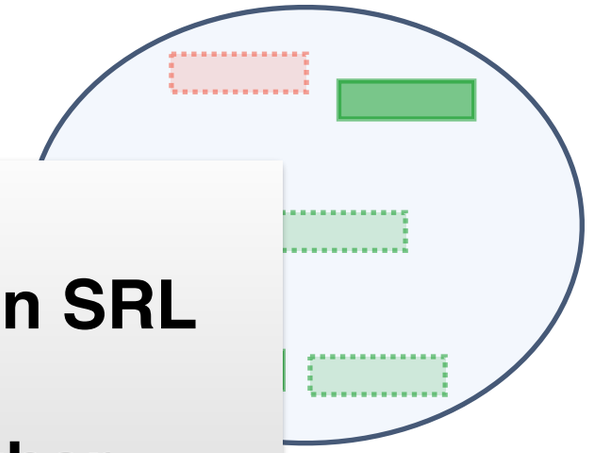
Learning from partial interpretations



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

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

Learning from partial interpretations



Key Points for parameter learning in SRL

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

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

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

Bayesian Parameter Learning

- Learning as inference
- From priors to posteriors
- Quite standard …

Part III : Learning

b. Rules / Structure

Information Extraction in NELL

Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/

Experiments

Table 4: Precision for different experimental setups and parameters (A: $m = 1, p = 0.99$, B: $m = 1000, p = 0.90$).

Setting	**athleteplaysforteam**		**athleteplayssport**		**teamplaysinleague**		**athleteplaysinleague**		**teamplaysagainstteam**	
train/test/rule	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
1: det/det/det	74.00	69.36	94.14	93.47	96.29	82.15	80.95	74.14	73.40	73.86
2: det/prob/det	73.51	69.57	97.53	94.85	96.70	87.83	90.83	77.73	73.70	73.35
3: det/prob/prob	74.67	69.82	95.86	94.74	96.35	82.57	82.26	75.29	73.84	74.34
4: det/prob/prob	77.25	73.87	96.53	96.04	98.00	90.59	84.91	79.36	77.26	77.83
5: det/prob/prob	74.76	69.97	95.85	94.69	96.44	82.51	81.99	75.07	73.90	74.16
6: prob/prob/det	75.83	73.11	93.40	93.76	94.44	93.67	79.41	79.42	80.87	80.60
7: prob/prob/prob	78.31	73.72	95.62	95.10	98.84	91.86	96.94	79.49	85.78	81.81

Table 3: Learned relational rules for the different predicates (fold 1).

0.9375::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,B).
0.9675::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,V1), teamplaysagainstteam(B,V1).
0.9375::athleteplaysforteam(A,B)	←	athleteplayssport(A,V1), teamplayssport(B,V1).
0.5109::athleteplaysforteam(A,B)	←	athleteplaysinleague(A,V1), teamplaysinleague(B,V1).
0.9070::athleteplayssport(A,B)	←	athleteledsportsteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B),teamalsoknownas(V1,V2).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V1), teamplayssport(V1,B).
0.9286::athleteplaysinleague(A,B)	←	athleteledsportsteam(A,V1), teamplaysinleague(V1,B).
0.7868::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplaysinleague(V1,B).
0.9384::athleteplaysinleague(A,B)	←	athleteplayssport(A,V2), athleteplayssport(V1,V2), teamplaysinleague(V1,B).
0.9024::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V1), teamplaysinleague(V1,B).

Rule learning in NELL

- Original approach by Cohen et al.
  - Make probabilistic data deterministic
    - run classic rule-learner (variant of FOIL)
    - re-introduce probabilities on learned rules and predict
- You can also do this directly ProbFOIL [De Raedt et al IJCAI 2015]

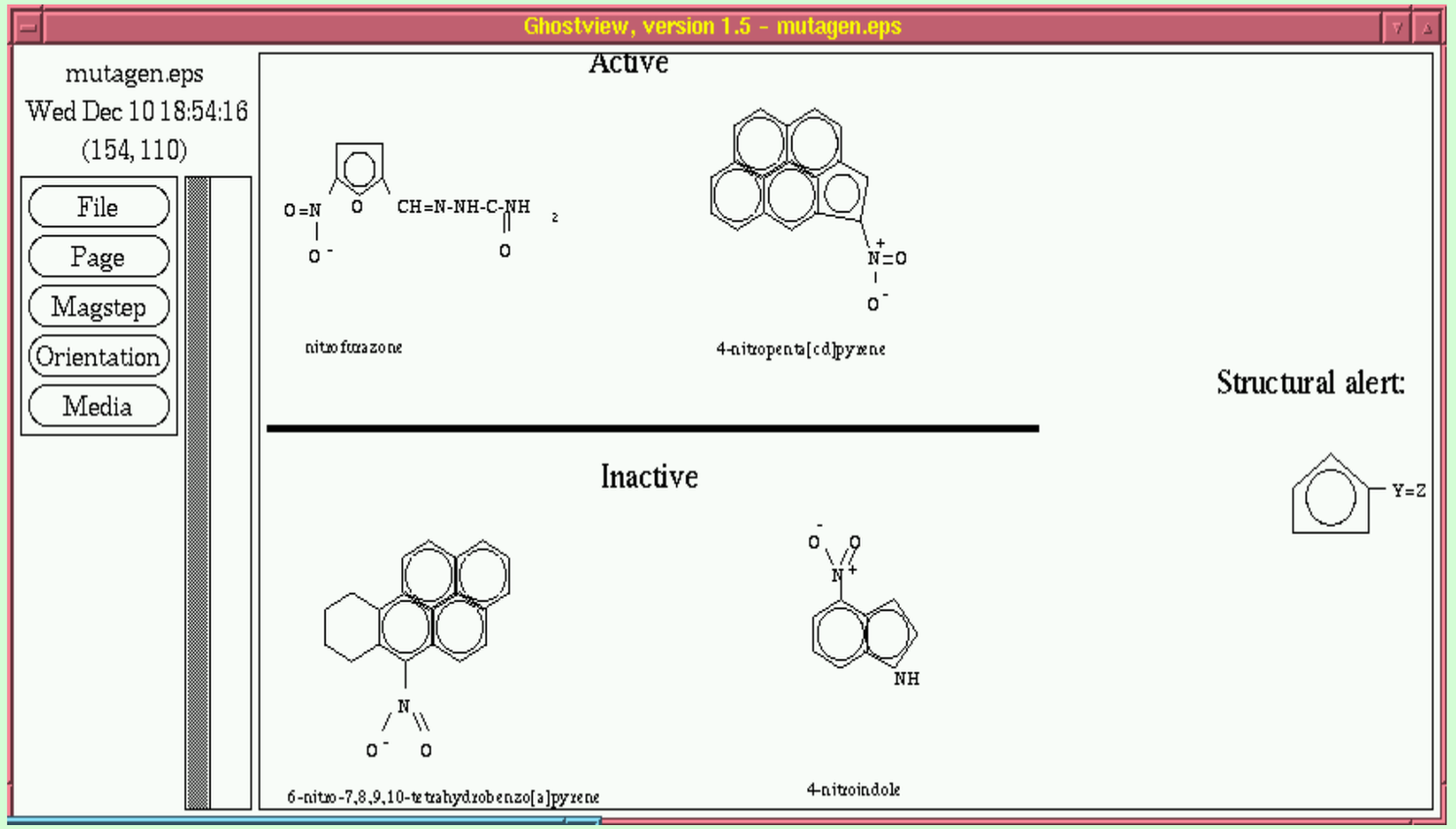
The Mutagenicity dataset

Background theory

```
molecule(225).
logmutag(225,0.64).
lumo(225,-1.785).
logp(225,1.01).
nitro(225,[f1_4,f1_8,f1_10,f1_9]).
atom(2
atom
atom
atom
...
ring
hete

hypo

mut

Exam
```

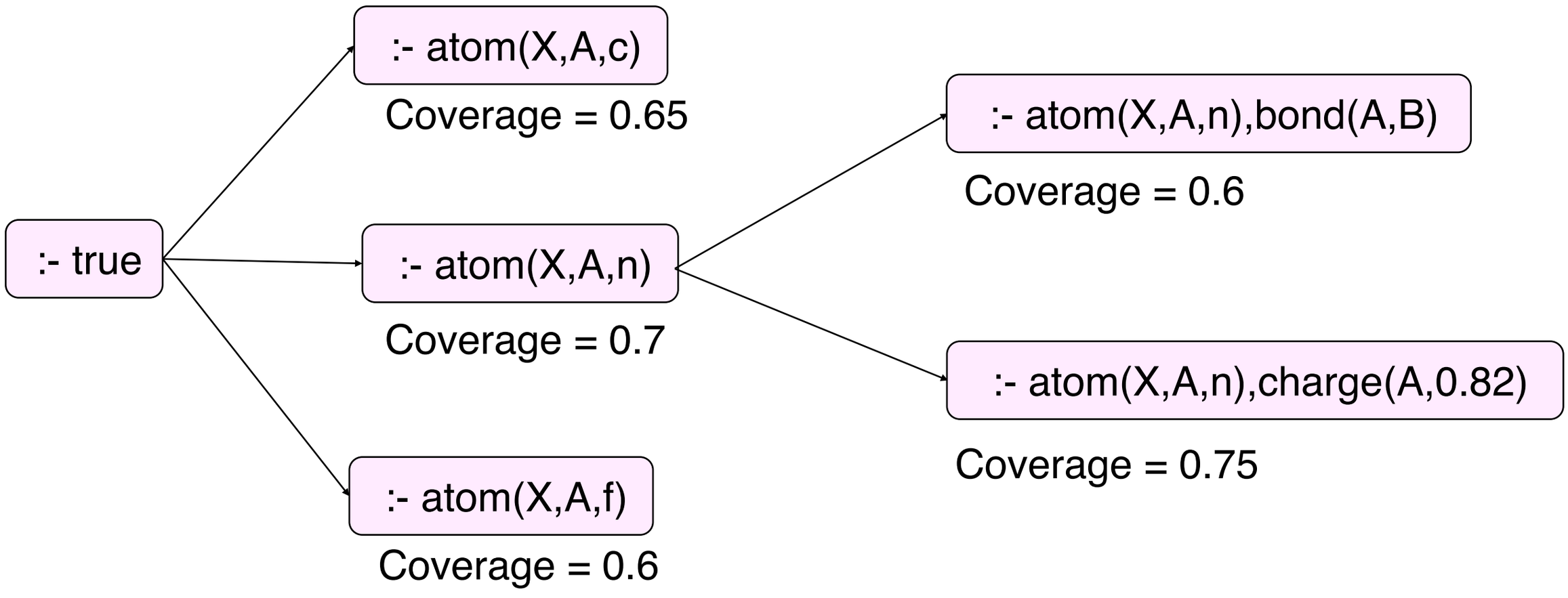
```
bond(225,f1_1,f1_2,7).
bond(225,f1_2,f1_3,7).
bond(225,f1_3,f1_4,7).
bond(225,f1_4,f1_5,7).
bond(225,f1_5,f1_1,7).
f1_9,2).
_10,2).
_11,1).
1_12,2).
1_13,1).
```

Applications



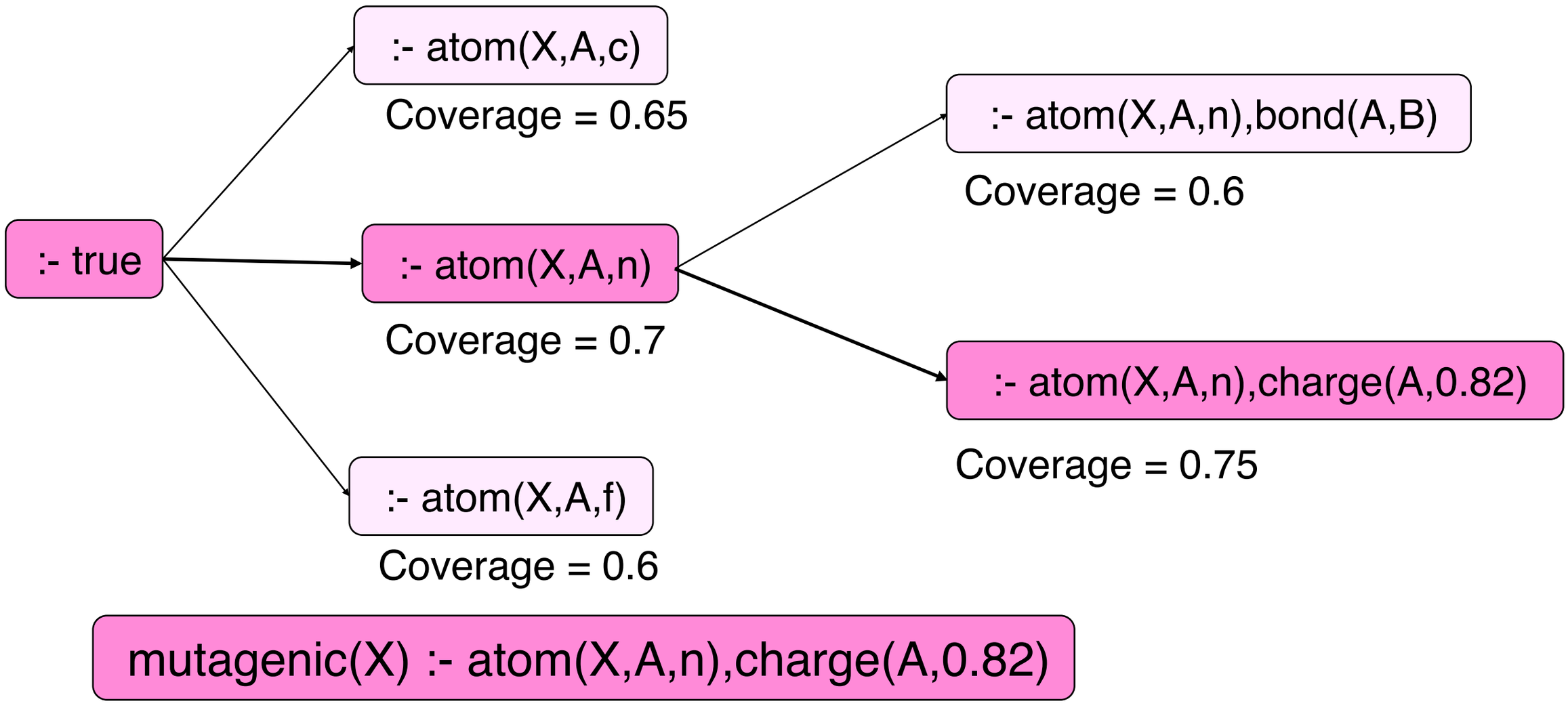
Learning: FOIL (Quinlan 1990)

- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



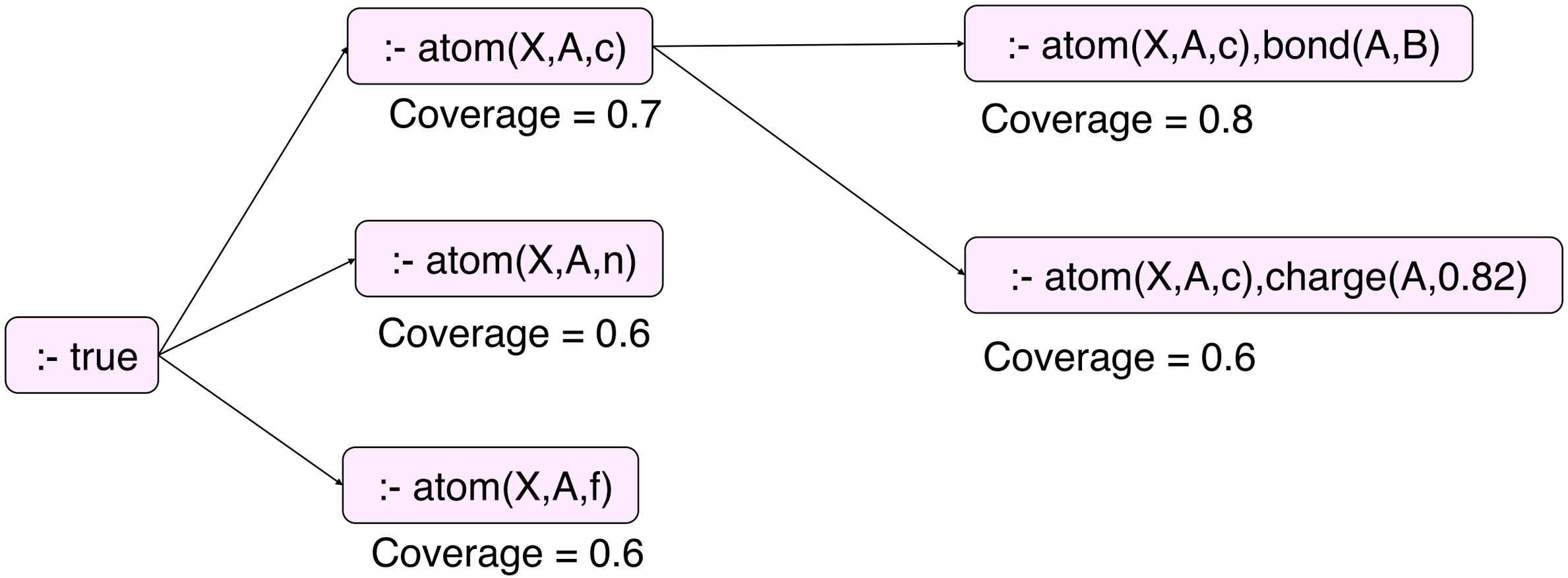
Learning: FOIL (Quinlan 1990)

- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



Learning: FOIL (Quinlan 1990)

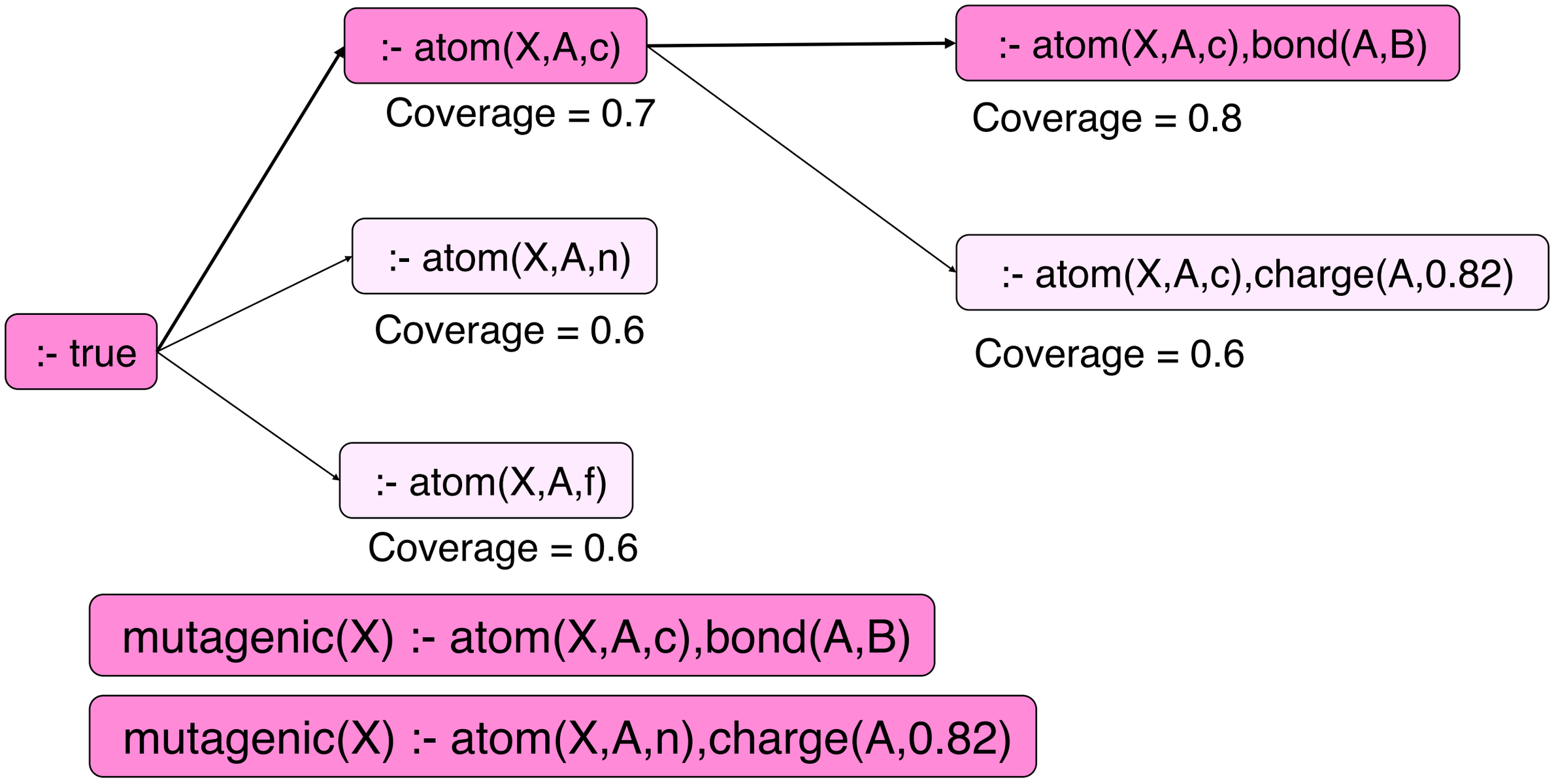
- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



mutagenic(X) :- atom(X,A,n),charge(A,0.82)

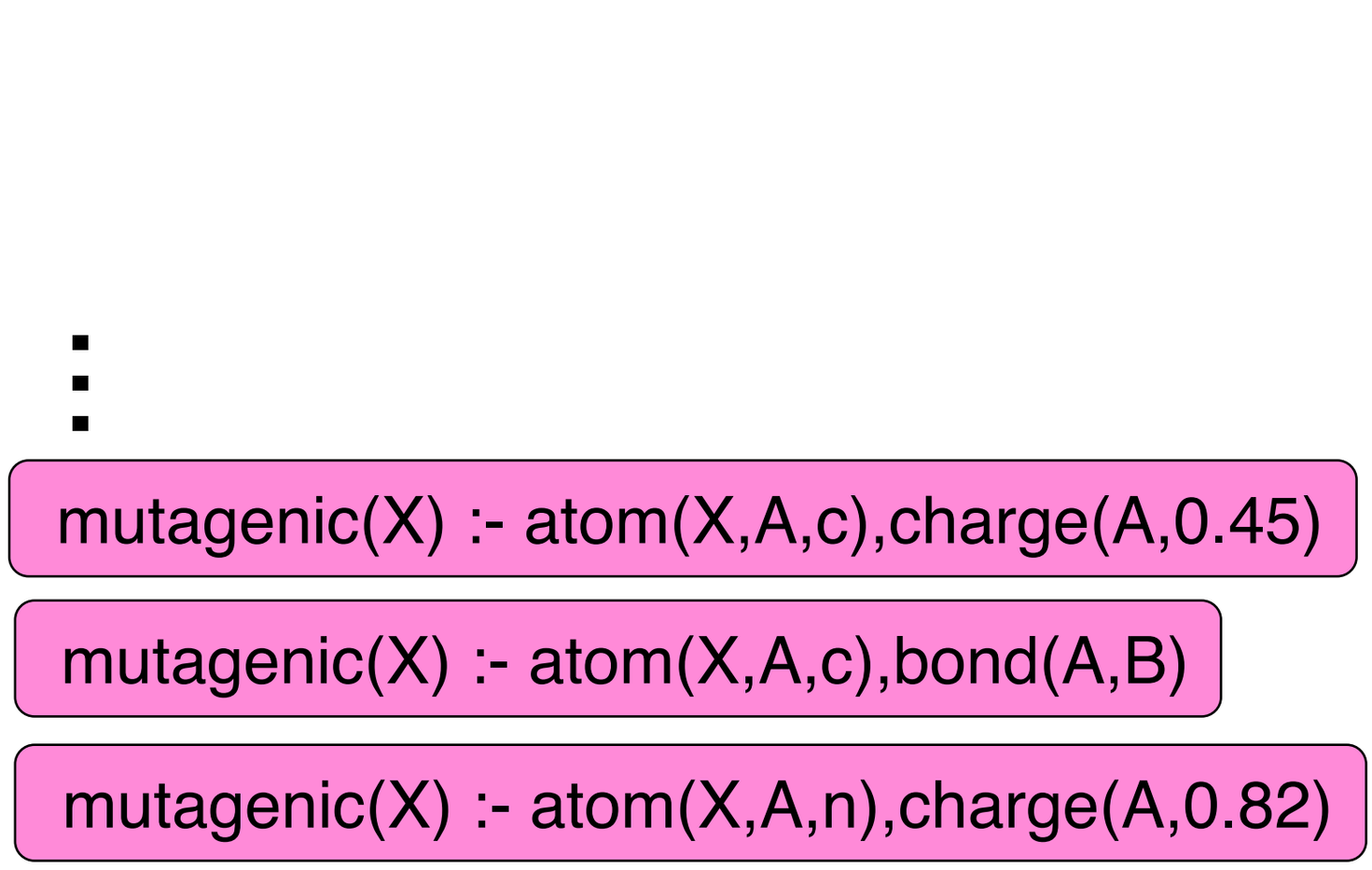
Learning: FOIL (Quinlan 1990)

- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



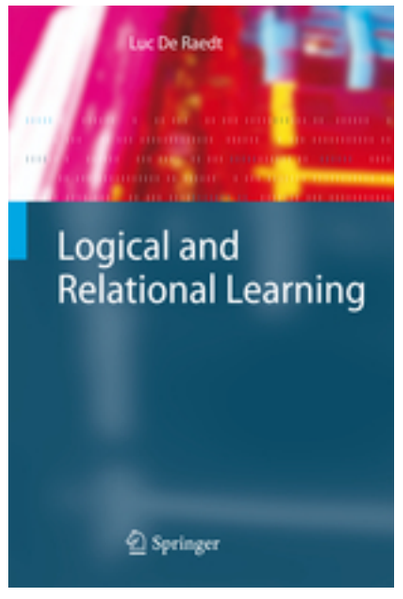
Learning: FOIL (Quinlan 1990)

- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



Structure Learning in (Statistical) Relational Learning

- StarAI / SRL structure learning models typically search such rule / clause spaces; very much in the spirit of logical and relational learning



- Variations on underlying models, scoring functions, search methods, …
- Also available for Markov Logic [Domingos et al] and Probabilistic Soft Logic [Getoor et al.]

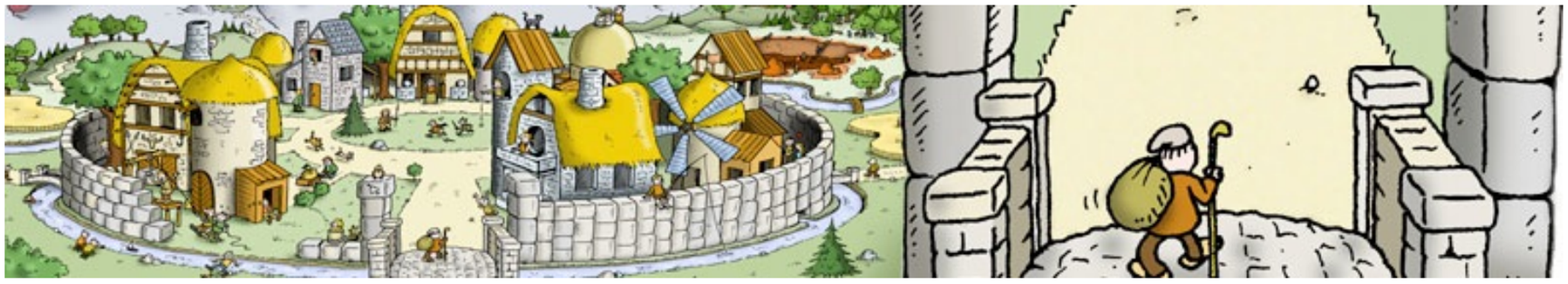
Roadmap

- Prob. Programming - Modeling
- Inference
- Learning
- Dynamics
- Logic, Probability and Neural Nets
- KBMC & Markov Logic
- Lifted Inference

... with some detours on the way

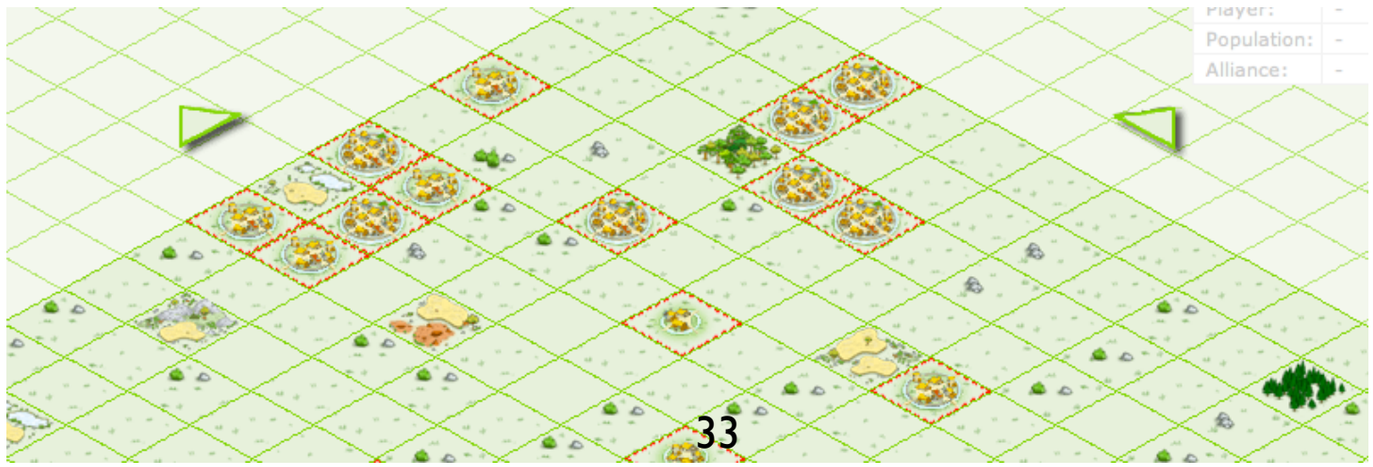
Part IV : Dynamics

Dynamics: Evolving Networks



- *Travian*: A massively multiplayer real-time strategy game
- Commercial game run by TravianGames GmbH
- ~3.000.000 players spread over different "worlds"
- ~25.000 players in one world

[Thon et al. ECML 08]



World Dynamics

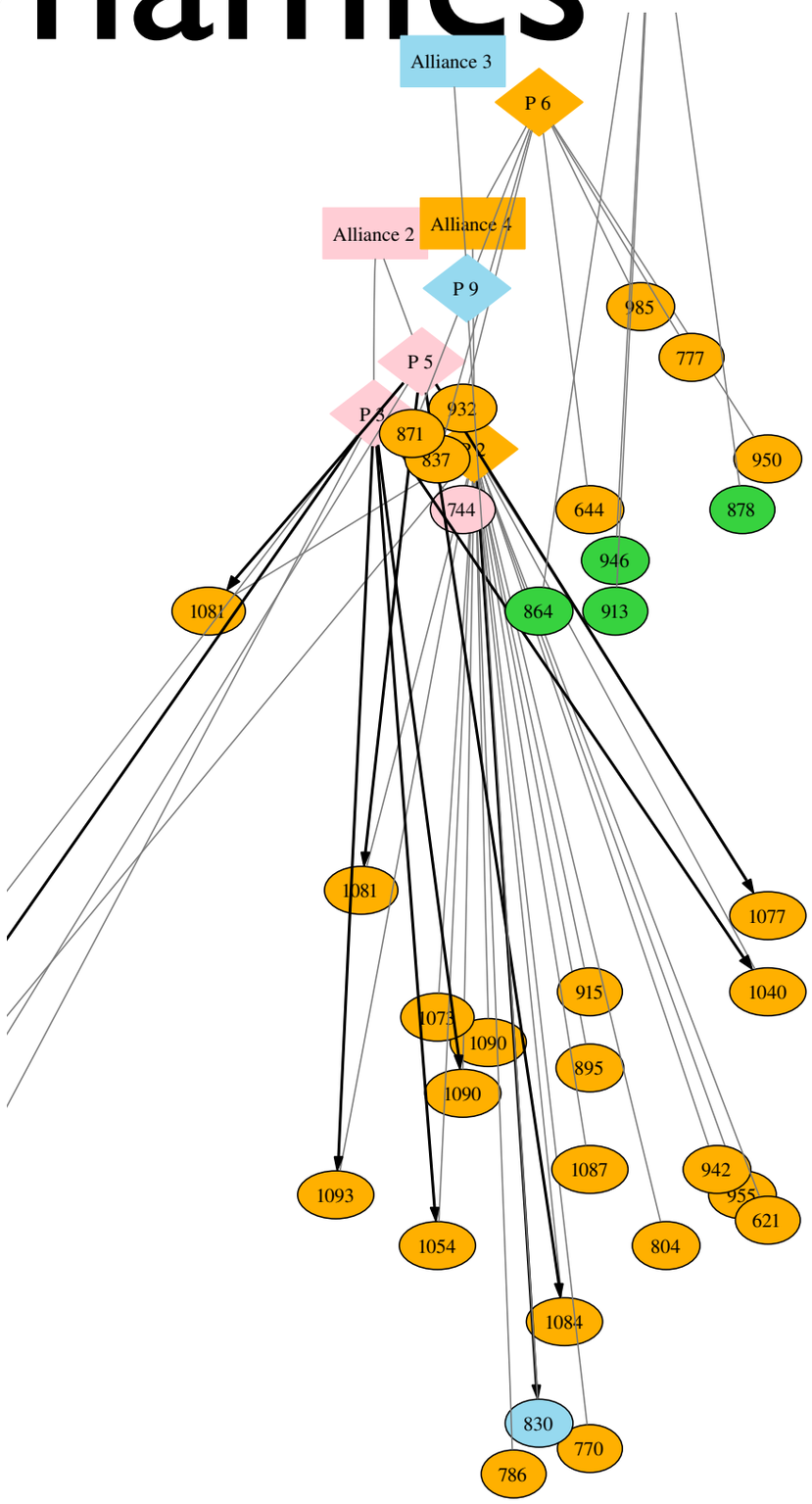
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

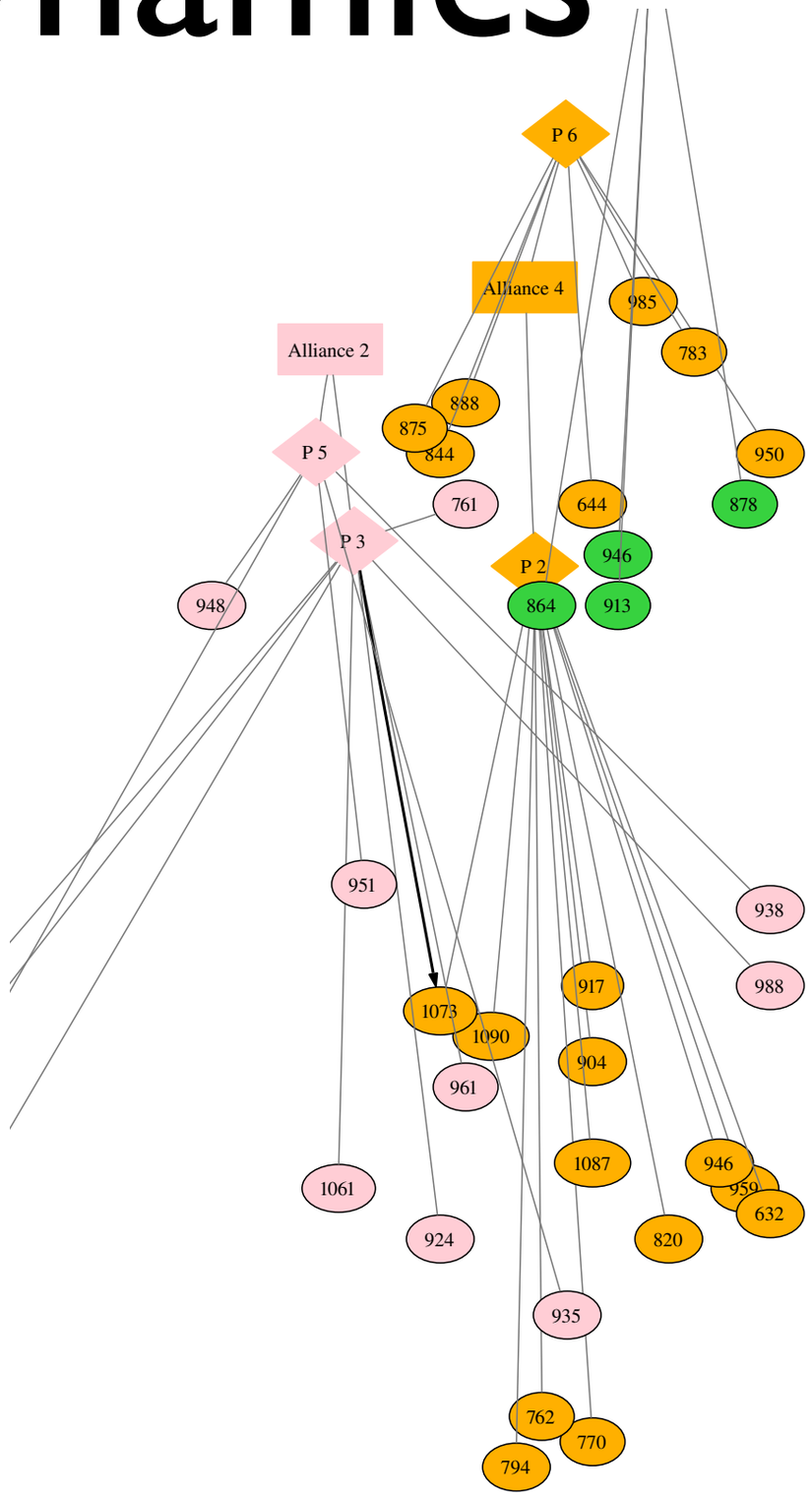
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

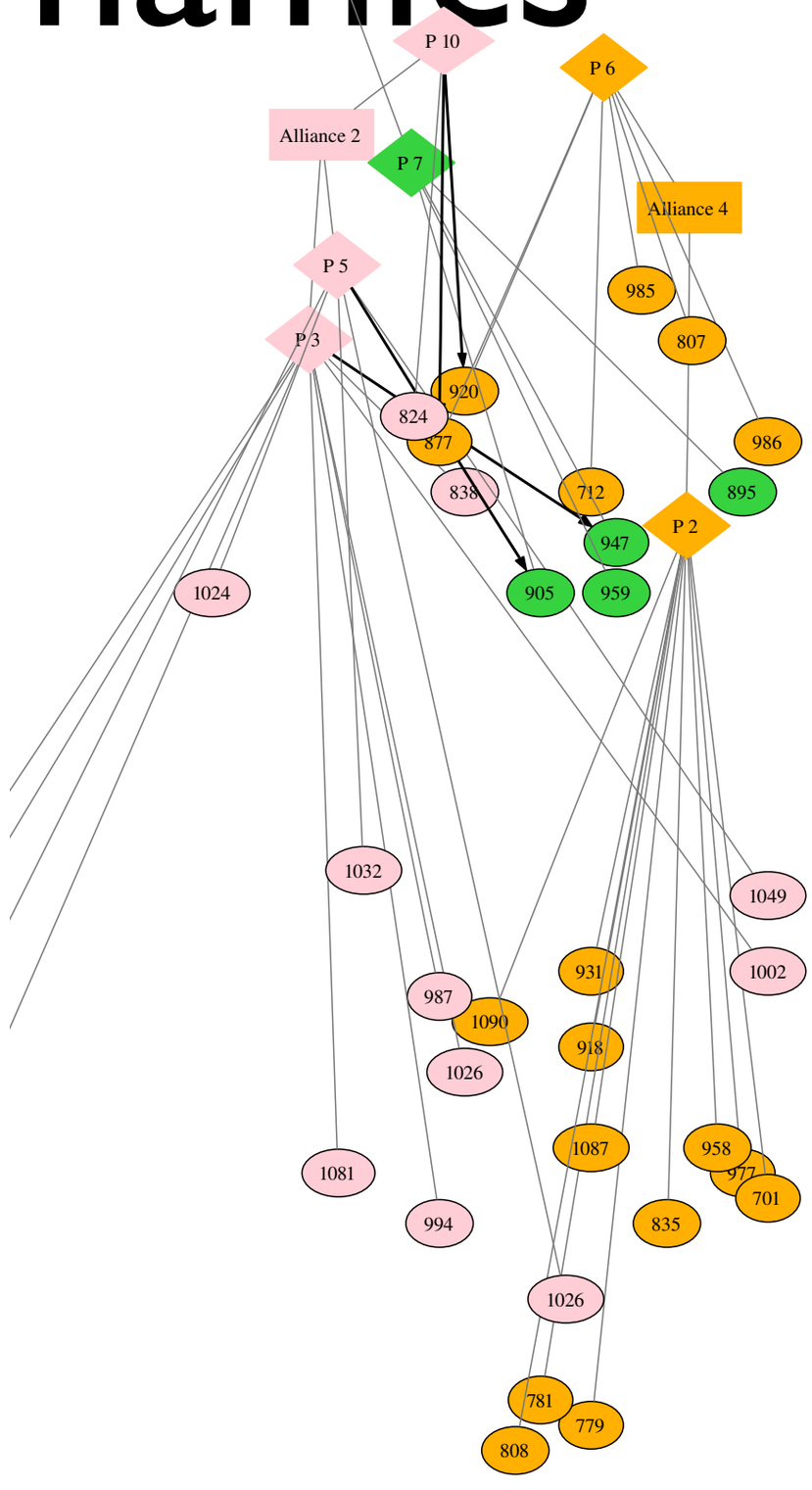
Fragment of world with

~10 alliances
~200 players
~600 cities

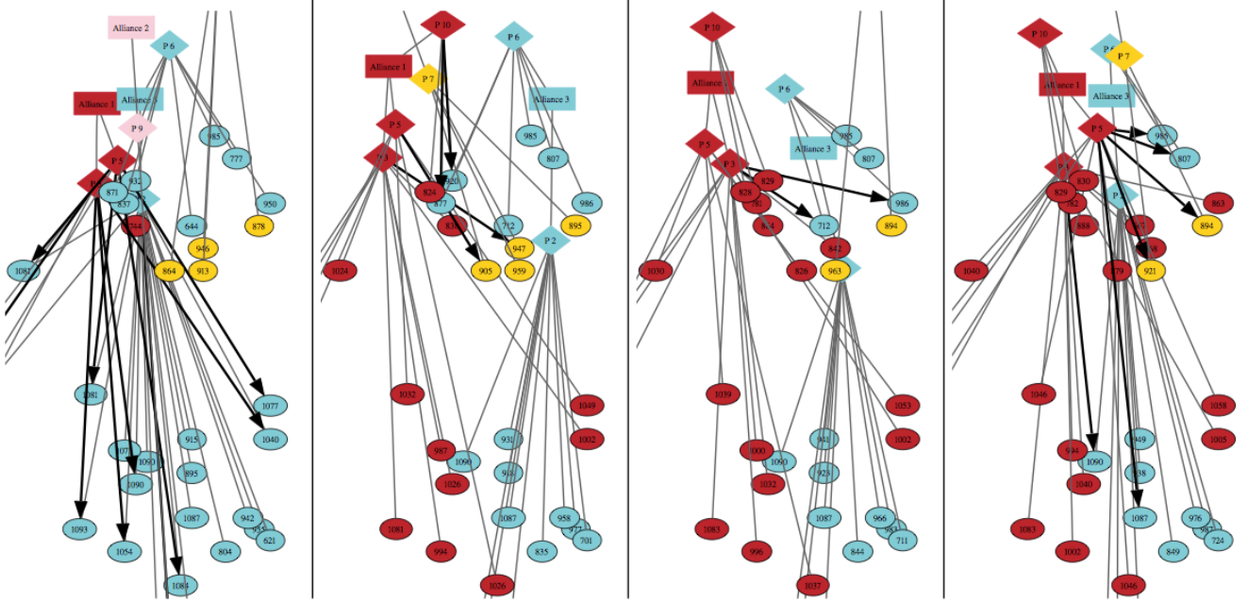
alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

one of the **effects** holds at time T+1

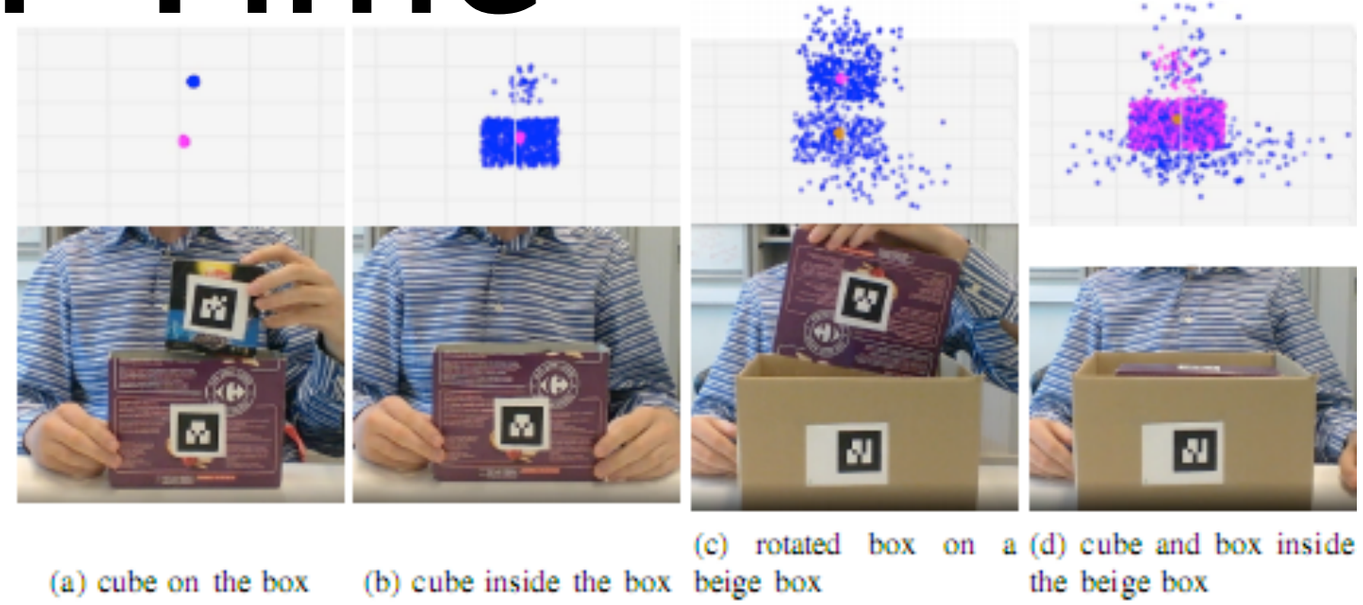
```
0.4::conquest(Attacker,C); 0.6::nil :-
```

```
city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

Relational State Estimation over Time



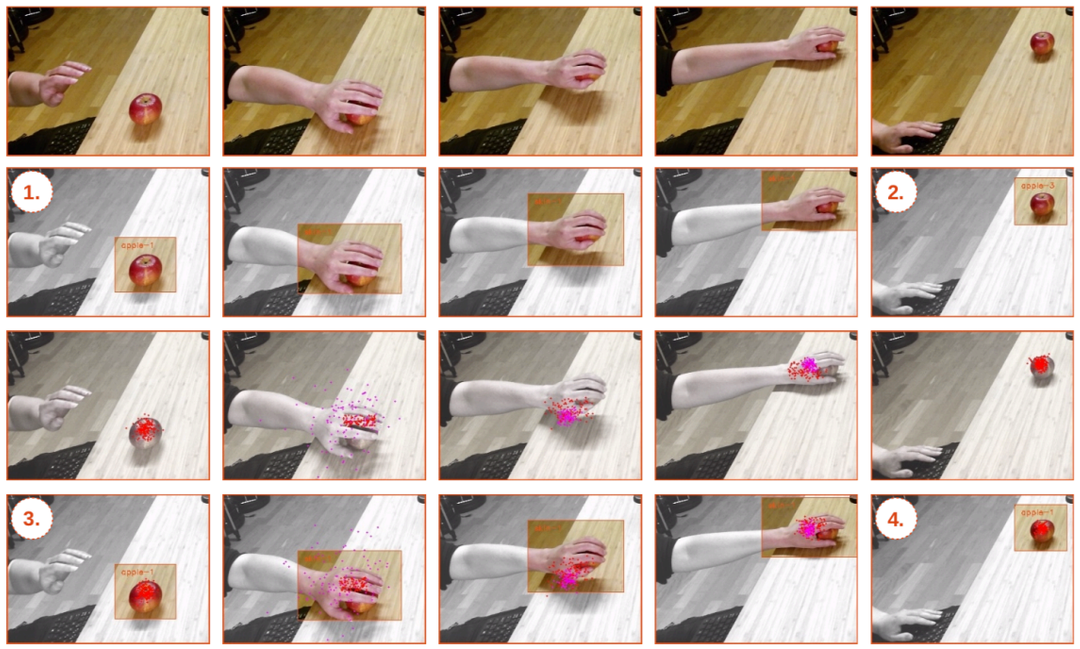
(a) cube on the box (b) cube inside the box (c) rotated box on a beige box (d) cube and box inside the beige box

Box scenario

- object tracking even when invisible
- estimate spatial relations

[Nitti et al, IROS 13]

Relational State Estimation over Time



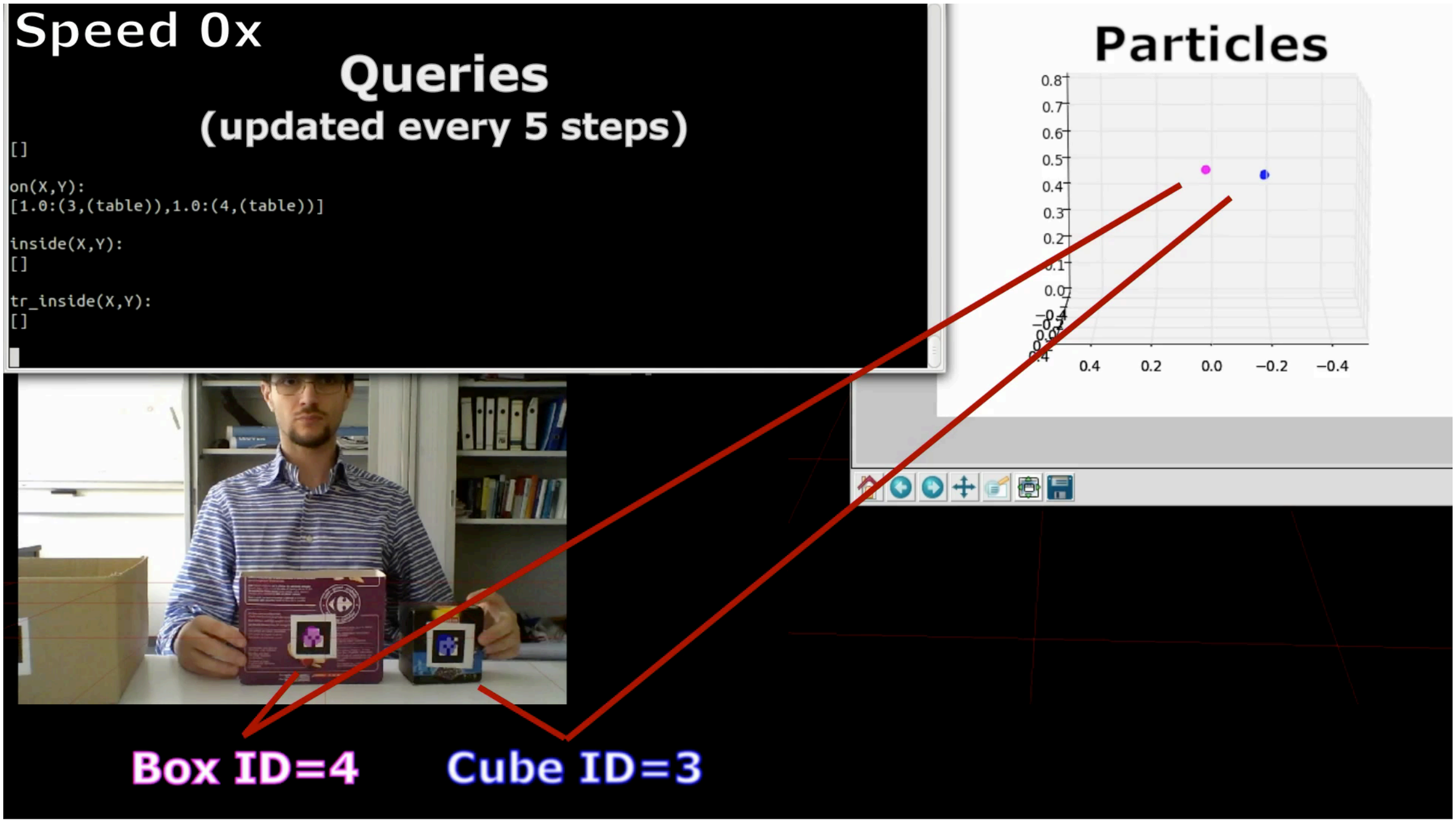


(a) cube on the box (b) cube inside the box beige box (c) rotated box on a (d) cube and box inside the beige box

Box scenario

- object tracking even when invisible
- estimate spatial relations

[Nitti et al, IROS 13]


Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

IROS 13

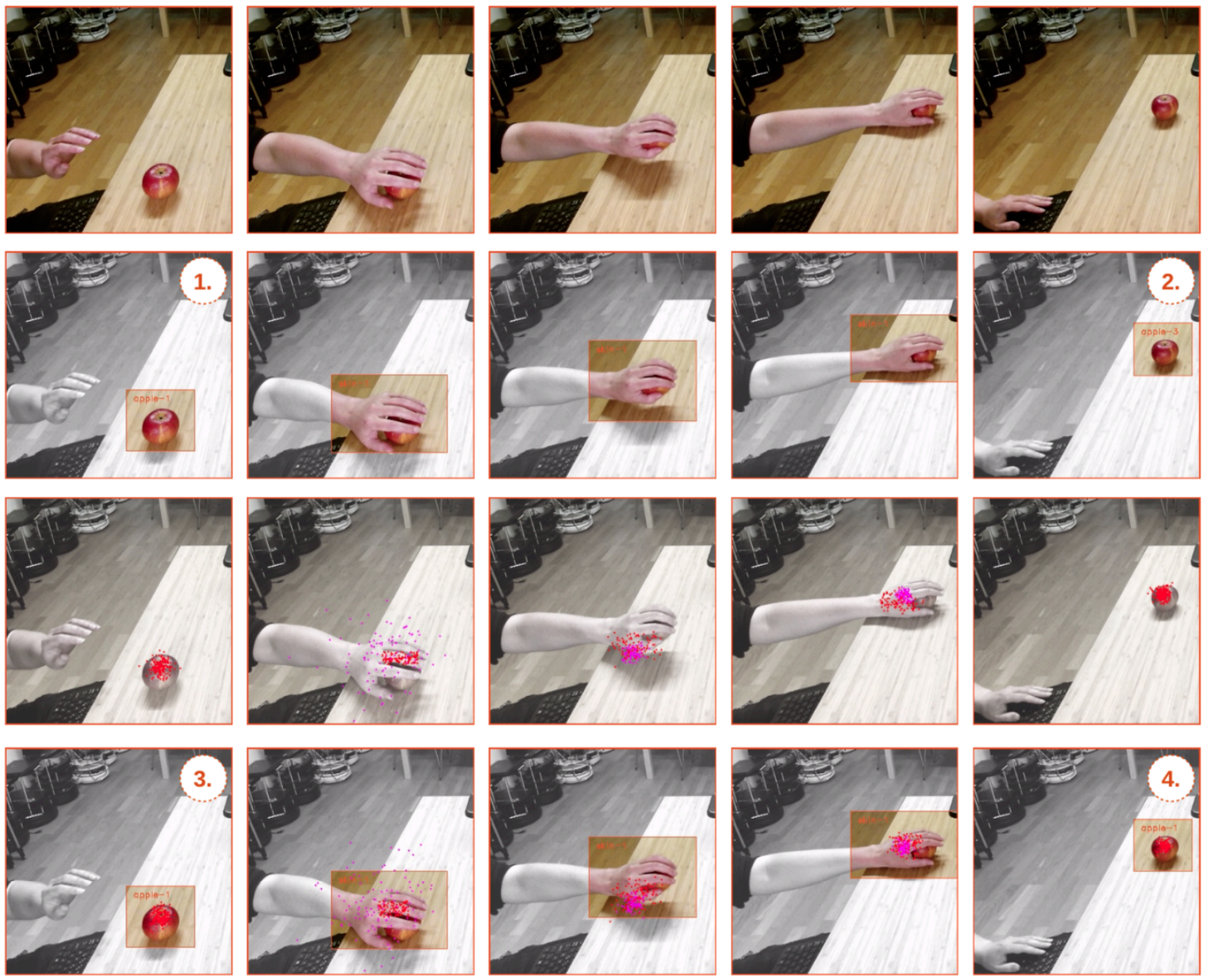


Fig. 5. A depiction of how suggested system benefits of combined object anchoring and probabilistic object tracking. *Rows in order from the top: 1st)* representing screen-shots of a scenario where a human hand is occluding an apple while the apple is moved, *2nd)* corresponding resulting anchored objects while *only* using the *anchoring system* (note that the original `'apple-1'` object is lost while it is occluded and moved by the `'skin-1'` object, and a new `'apple-3'` object is, therefore, *acquired* in the end of the scenario), *3rd)* plotted particles given by the *inference system* during execution of suggested integrated approach, and *4th)* corresponding resulting anchored objects of the *anchoring system* supported by the feed back of the *inference system* (note that in this case is the position of `'apple-1'` object tracked while it is occluded and moved by the `'skin-1'` object, and the `'apple-1'` object is, accordingly, *re-acquired* in the end of the scenario).

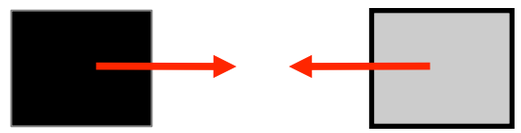
From Persson et al.,
IEEE Trans. Cognitive and Developmental Systems,
in press

Magnetic scenario

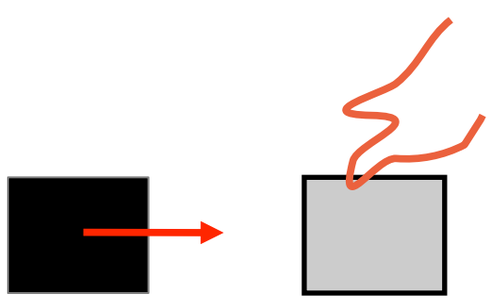
- 3 object types: magnetic, ferromagnetic, nonmagnetic

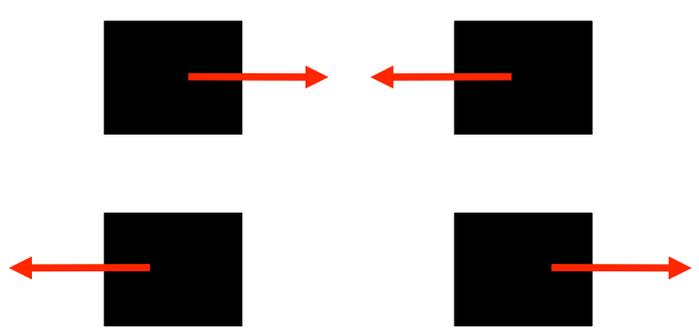


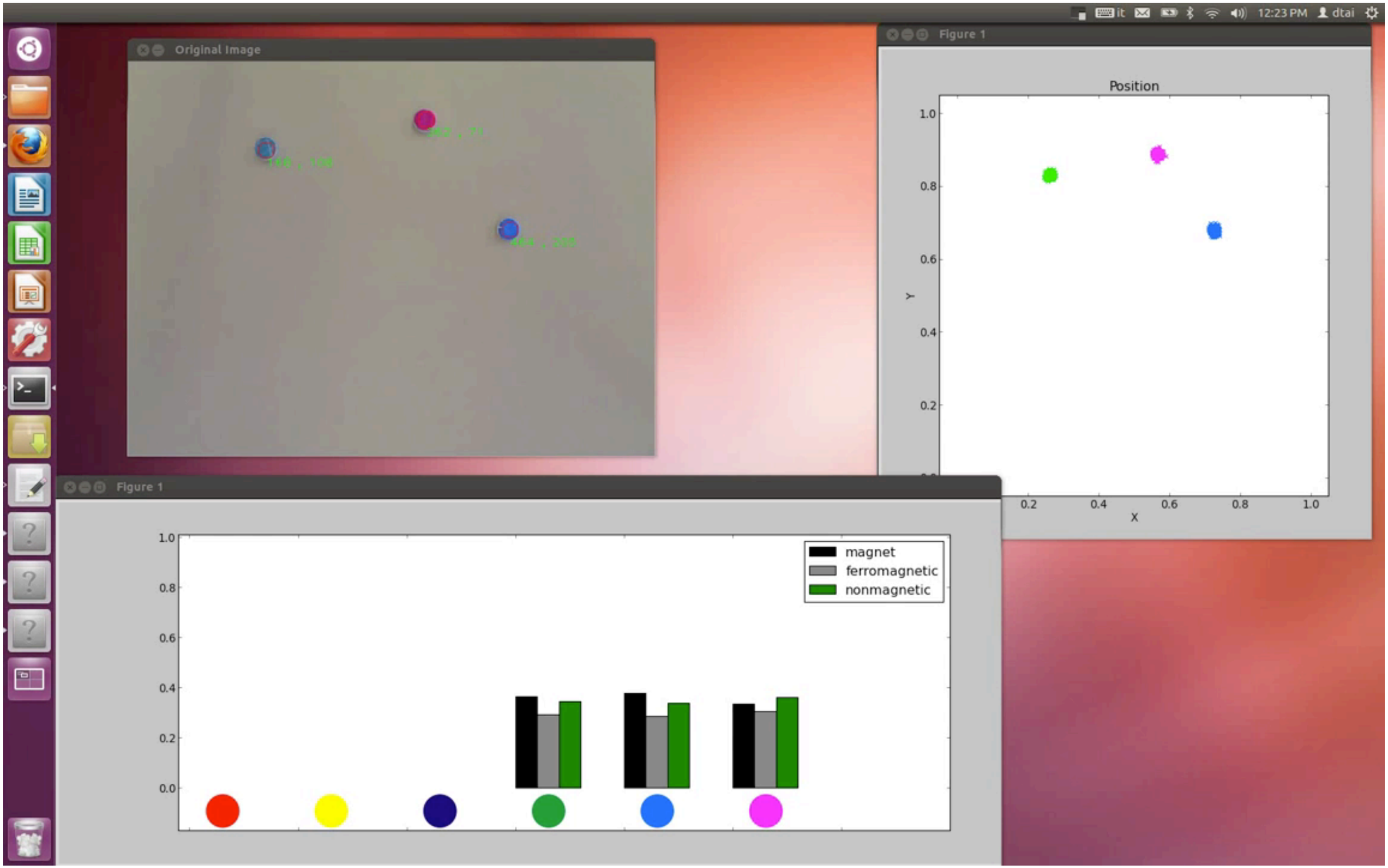
- Nonmagnetic objects do not interact
- A magnet and a ferromagnetic object attract each other



- Magnetic force that depends on the distance
- If an object is held magnetic force is compensated.






it
12:23 PM
dtai
Original Image
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
Y
0.2
0.4
0.6
0.8
1.0
X
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
magnet
ferromagnetic
nonmagnetic

Magnetic scenario

- 3 object types: magnetic, ferromagnetic, nonmagnetic

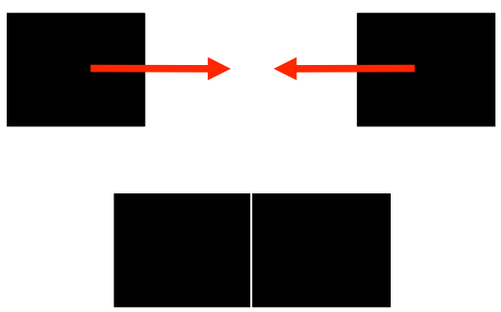
$$\text{type}(X)_t \sim \text{finite}([1/3\text{:magnet}, 1/3\text{:ferromagnetic}, 1/3\text{:nonmagnetic}]) \leftarrow \text{object}(X).$$

- 2 magnets attract or repulse

$$\text{interaction}(A,B)_t \sim \text{finite}([0.5\text{:attraction}, 0.5\text{:repulsion}]) \leftarrow \text{object}(A), \text{object}(B), A<B, \text{type}(A)_t = \text{magnet}, \text{type}(B)_t = \text{magnet}.$$

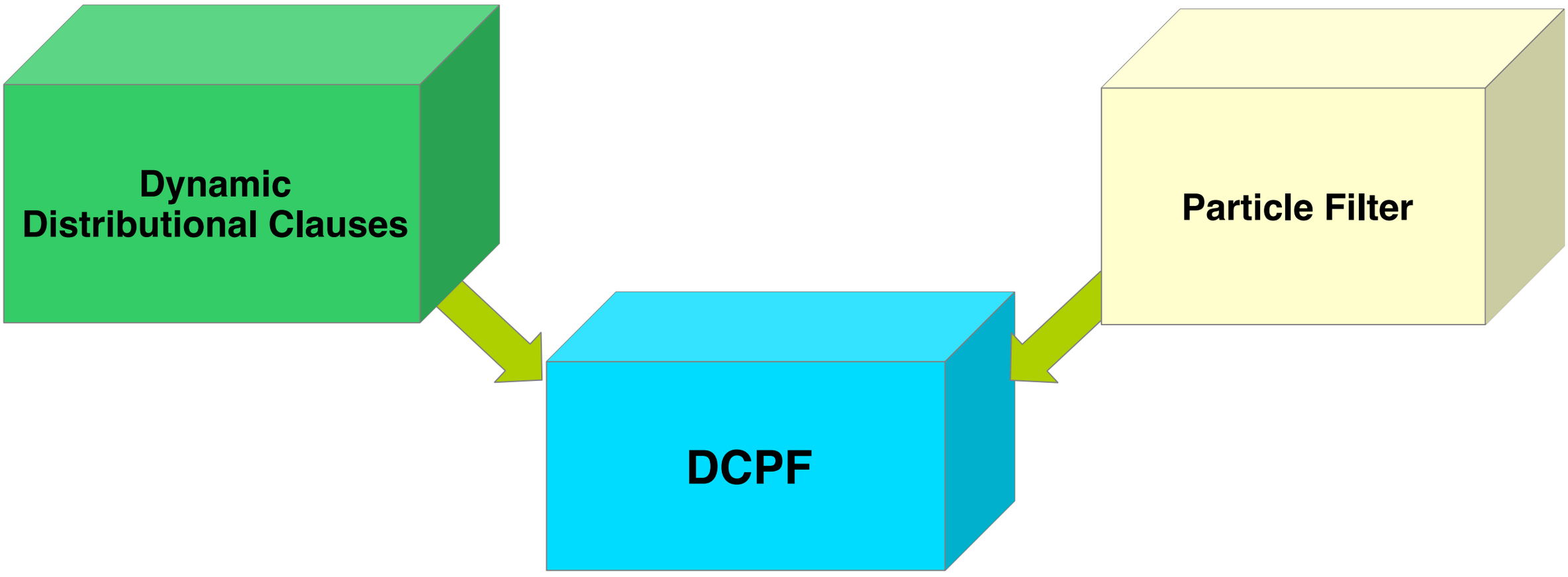
- Next position after attraction

$$\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{middlepoint}(A,B)_t, \text{Cov}) \leftarrow \text{near}(A,B)_t, \text{not}(\text{held}(A)), \text{not}(\text{held}(B)), \text{interaction}(A,B)_t = \text{attr}, c/\text{dist}(A,B)_t^2 > \text{friction}(A)_t.$$



$$\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{pos}(A)_t, \text{Cov}) \leftarrow \text{not}(\text{attraction}(A,B)).$$

DC Particle Filter (DCPF)



Goal

Flexible (relational) state representation

Fast inference (state estimation) in general models

"A particle filter for hybrid relational domains" IROS 2013
D. Nitti, T. De Laet, L. De Raedt

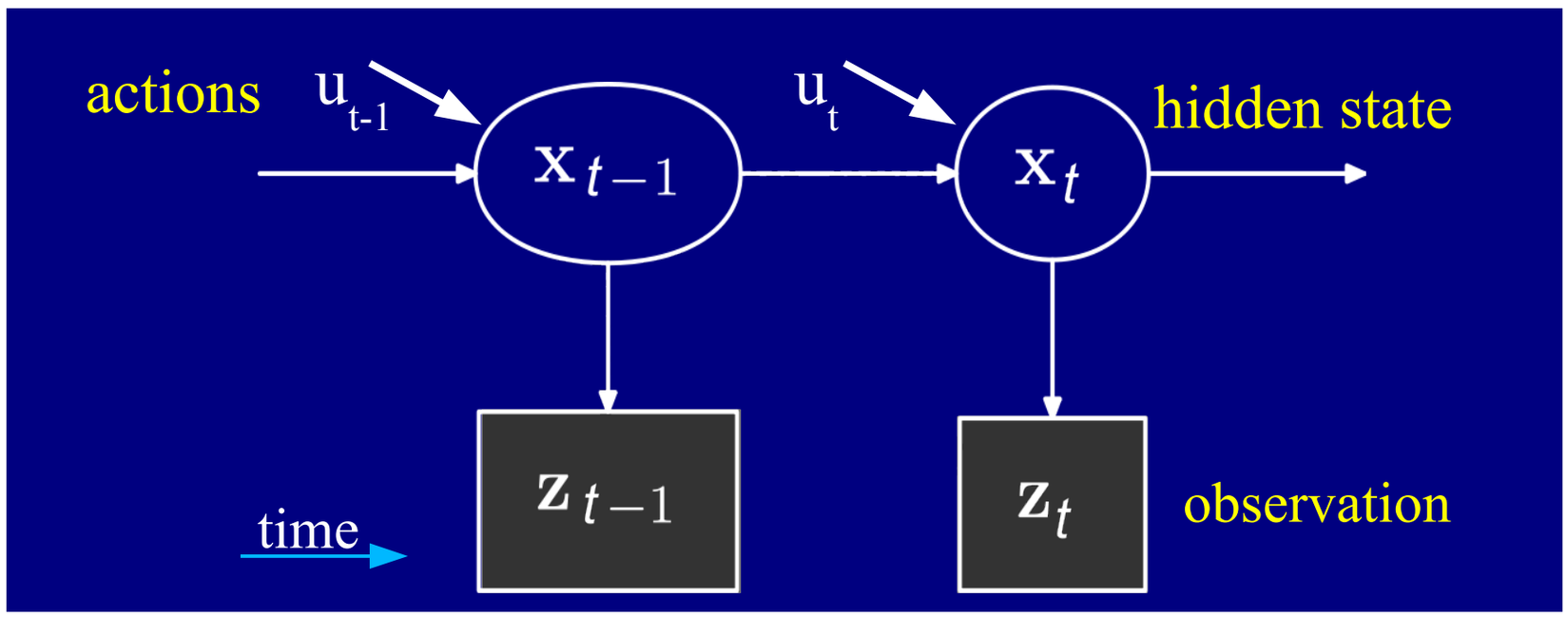
Dynamic Distributional Clauses

Prior distribution $p(x_0)$

State transition model $p(x_t|x_{t-1},u_t)$

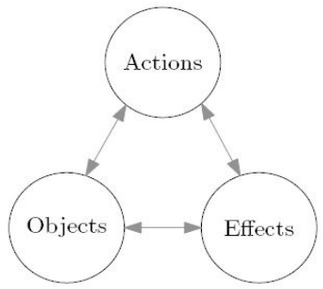
Measurement model $p(z_t|x_t)$

Other rules: $p(x'_t|x''_t)$



Learning relational affordances

Learn probabilistic model



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

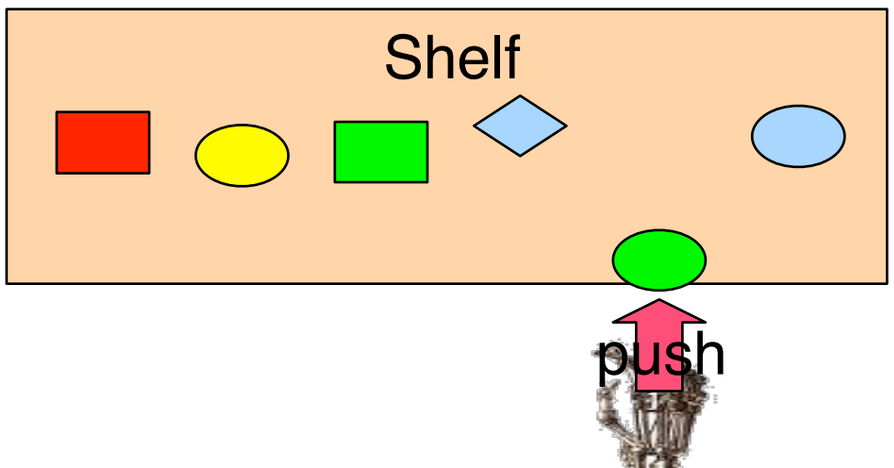
Learning relational affordances between two objects (learnt by experience)

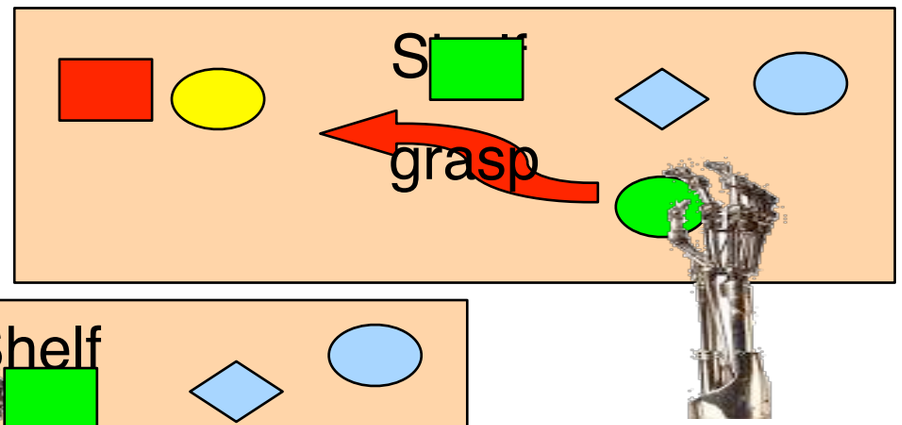
From two object interactions

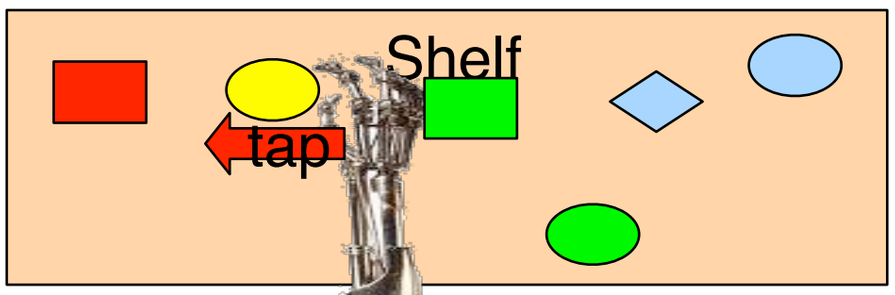
Generalize to N

Moldovan et al. ICRA 12, 13, 14

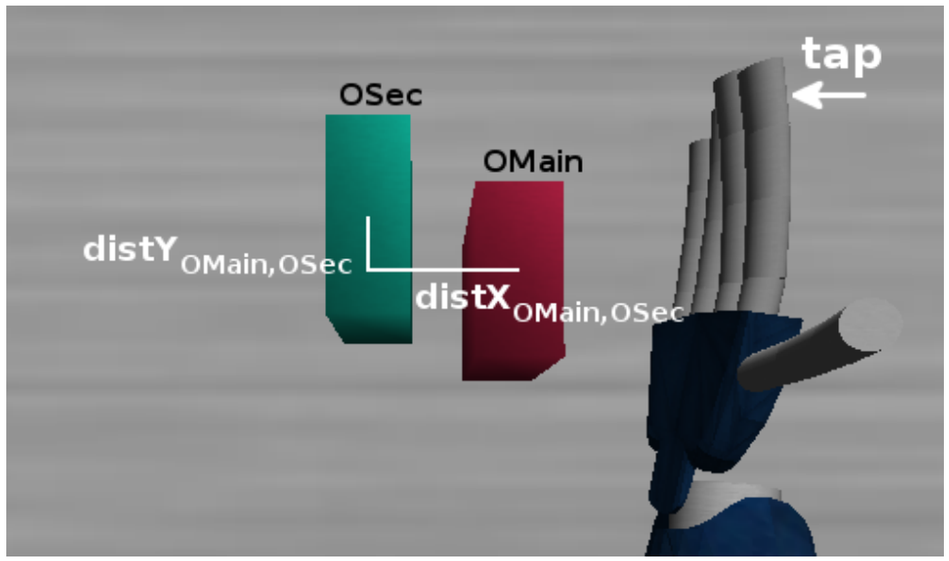
Relational MDPS

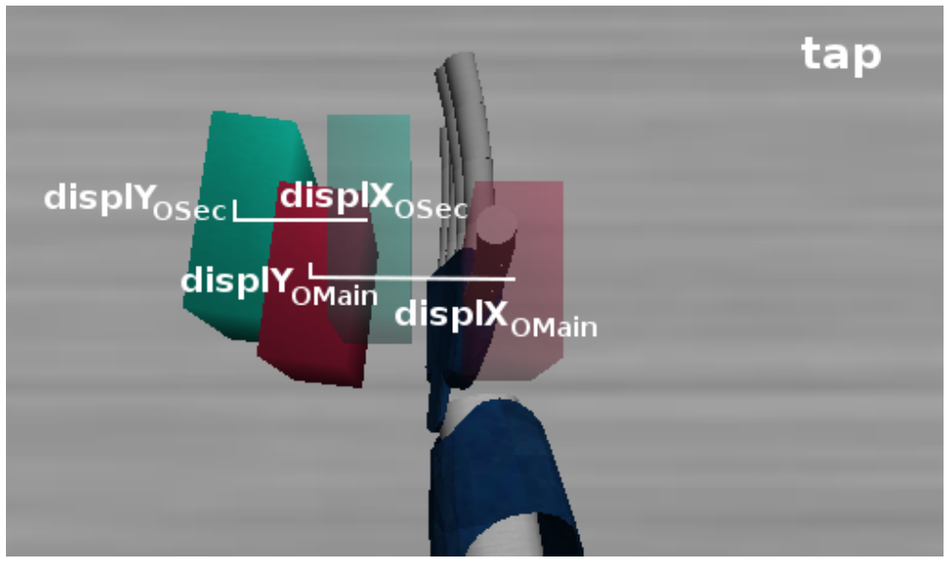






What is an affordance ?

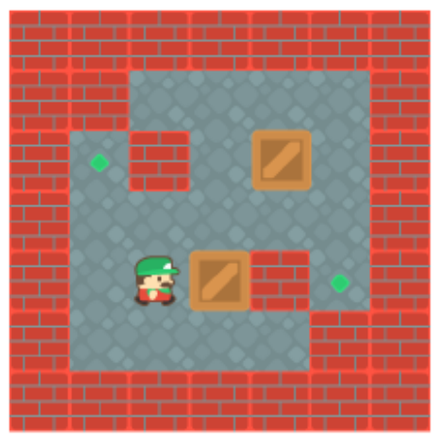




Clip 8: Relational O before (l), and E after the action execution (r).

Table 1: Example collected O, A, E data for action in Figure 8

Object Properties	Action	Effects
$shape_{O_{Main}} : sprism$ $shape_{O_{Sec}} : sprism$ $distX_{O_{Main},O_{Sec}} : 6.94cm$ $distY_{O_{Main},O_{Sec}} : 1.90cm$	$tap(10)$	$displX_{O_{Main}} : 10.33cm$ $displY_{O_{Main}} : -0.68cm$ $displX_{O_{Sec}} : 7.43cm$ $displY_{O_{Sec}} : -1.31cm$

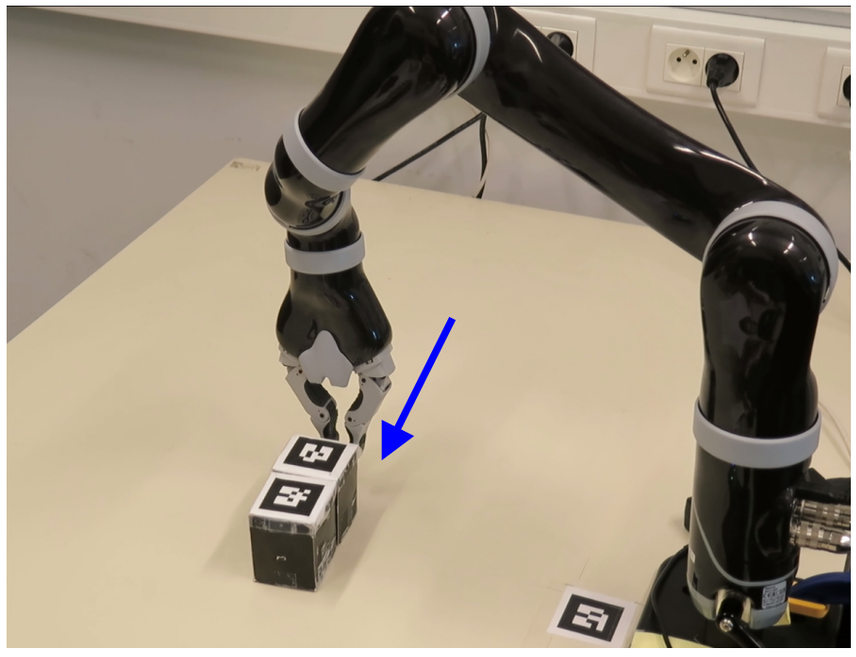


- Formalism — related to STRIPS but models delta
  - but also joint probability model over A, E, O

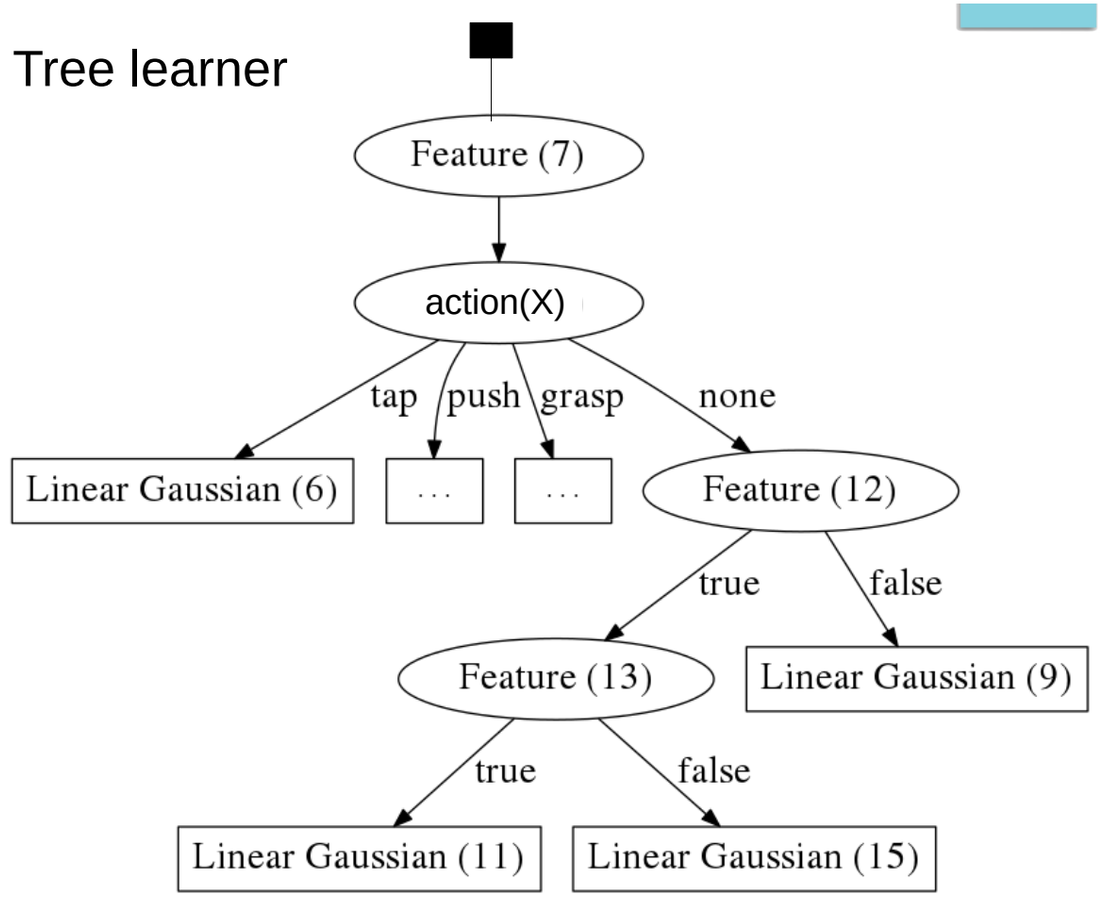
Relational MDP
From Rosman

Relational Affordance Learning

- **Learning the Structure of Dynamic Hybrid Relational Models**
  Nitti, Ravkic, et al. ECAI 2016
  - Captures relations/affordances
  - Suited to learn affordances in
    robotics set-up, continuous and discrete variables
  - Planning in hybrid robotics domain



DDC Tree learner

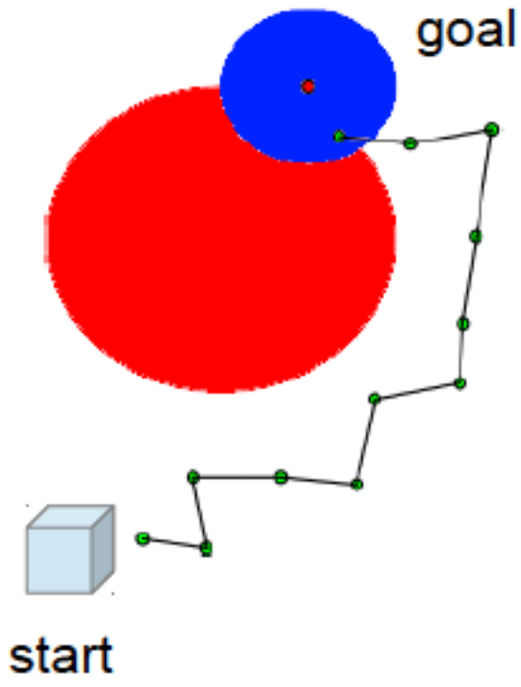


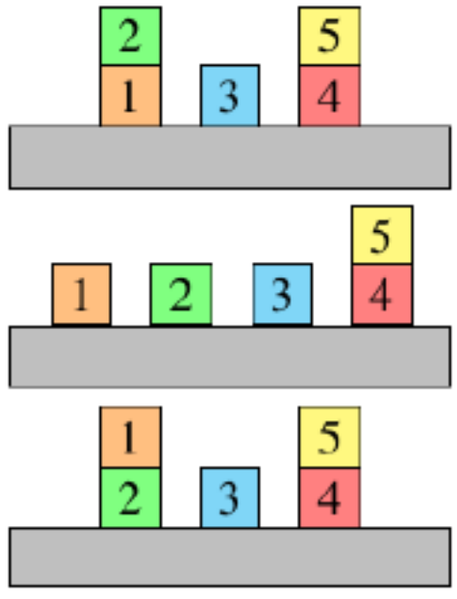
Planning

- Main task: probabilistic planning

  Find the best action to achieve the goal
- Discrete + continuous + relational representation







[Nitti et al ECML 15, MLJ]

ProbLog for activity recognition from video



CAVIAR-INRIA human activity dataset

28 videos
≈ 26.500 frames

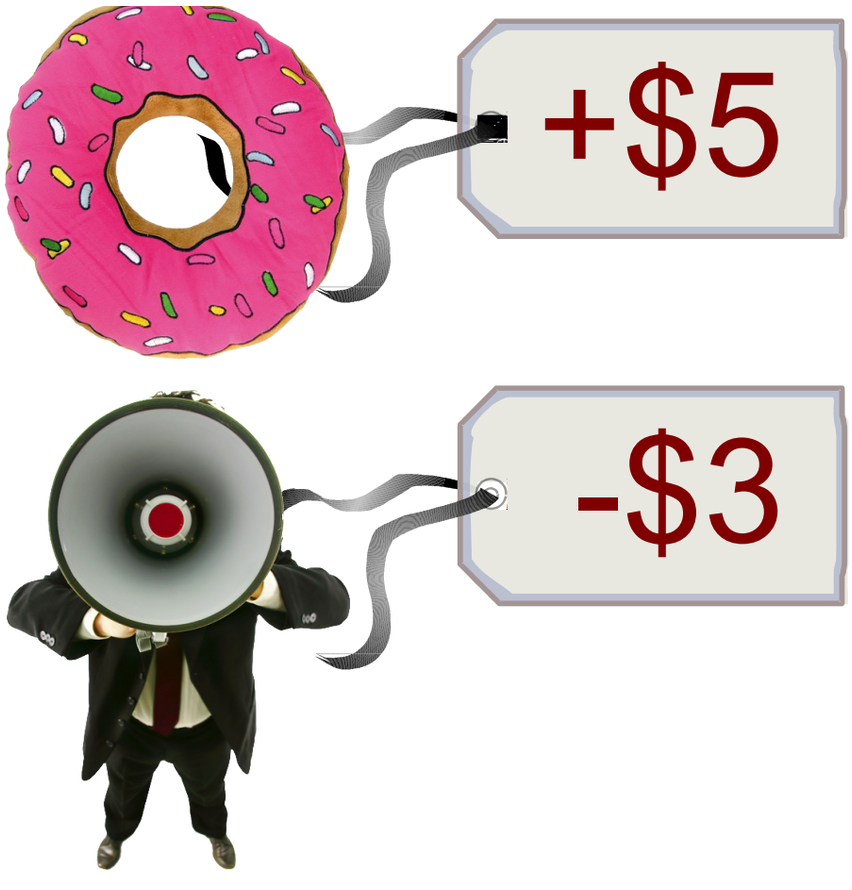
- Separation between low-level events (LLE) and high-level events (HLE)
  - LLE: *walking, running, active, inactive, abrupt*
  - HLE: *meeting, moving, fighting, leaving_object*
- Probabilistic Logic approach: *Event Calculus in ProbLog* (Prob-EC) to infer the high-level events from an **algebra** of low-level events.
- Example:

$$initiatedAt(fighting(P_1, P_2) = true, T) \leftarrow$$
$$happensAt(abrupt(P_1), T),$$
$$holdsAt(close(P_1, P_2, 44) = true, T),$$
$$not\ happensAt(inactive(P_2), T).$$

[Skarlatidis et al, TPLP 13]

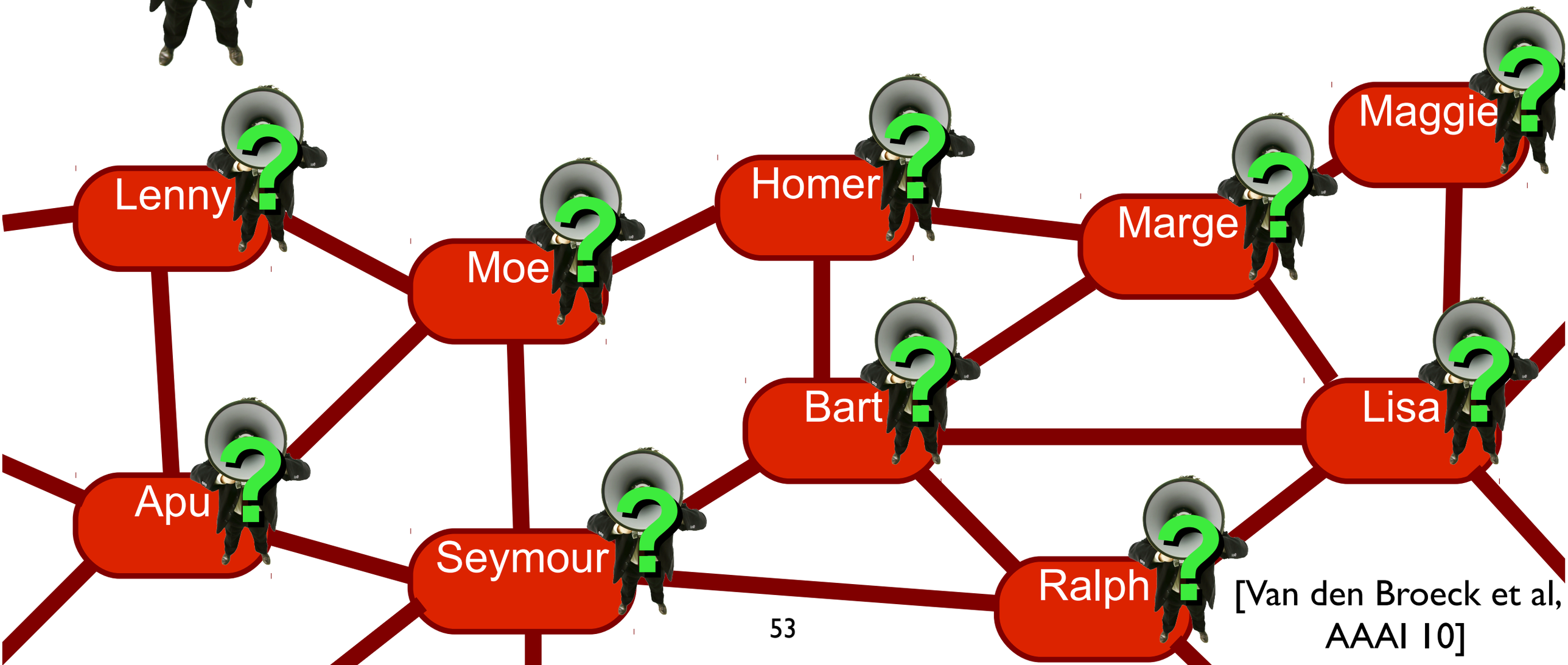
Decisions

Viral Marketing



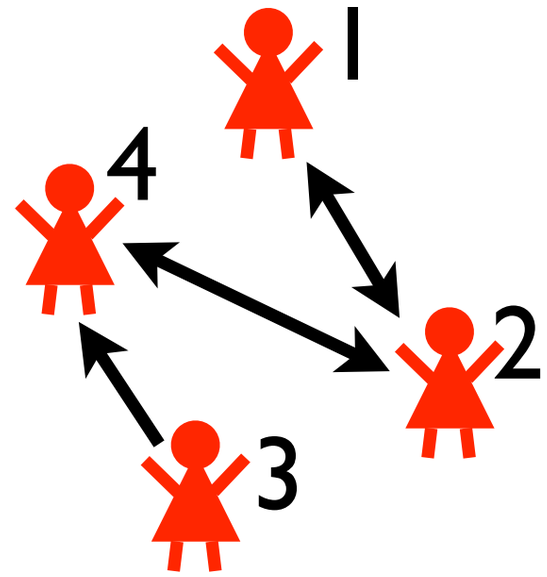
Which advertising strategy maximizes expected profit?

decide truth values of some atoms



[Van den Broeck et al, AAAI 10]

DTProbLog




```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

decision fact: true or false?

probabilistic facts + logical rules

utility facts: cost/reward if true

```
marketed(1)          marketed(3)
   bt(2,1)    bt(2,4)          bm(1)
buys(1)      buys(2)
```

world contributes 0.0032 to expected utility

probability = 0.0032

utility = 3 + 5 + 5 = 13

task: find strategy that maximizes expected utility

solution: using ProbLog technology

expected utility of strategy

Phenetic

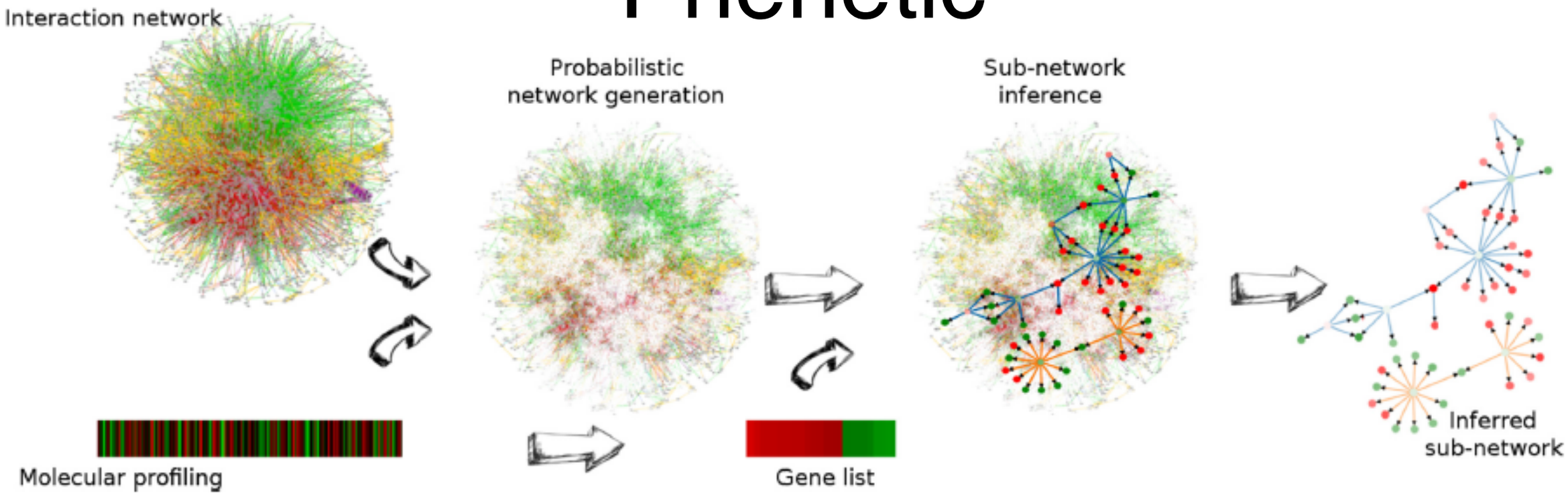


Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for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

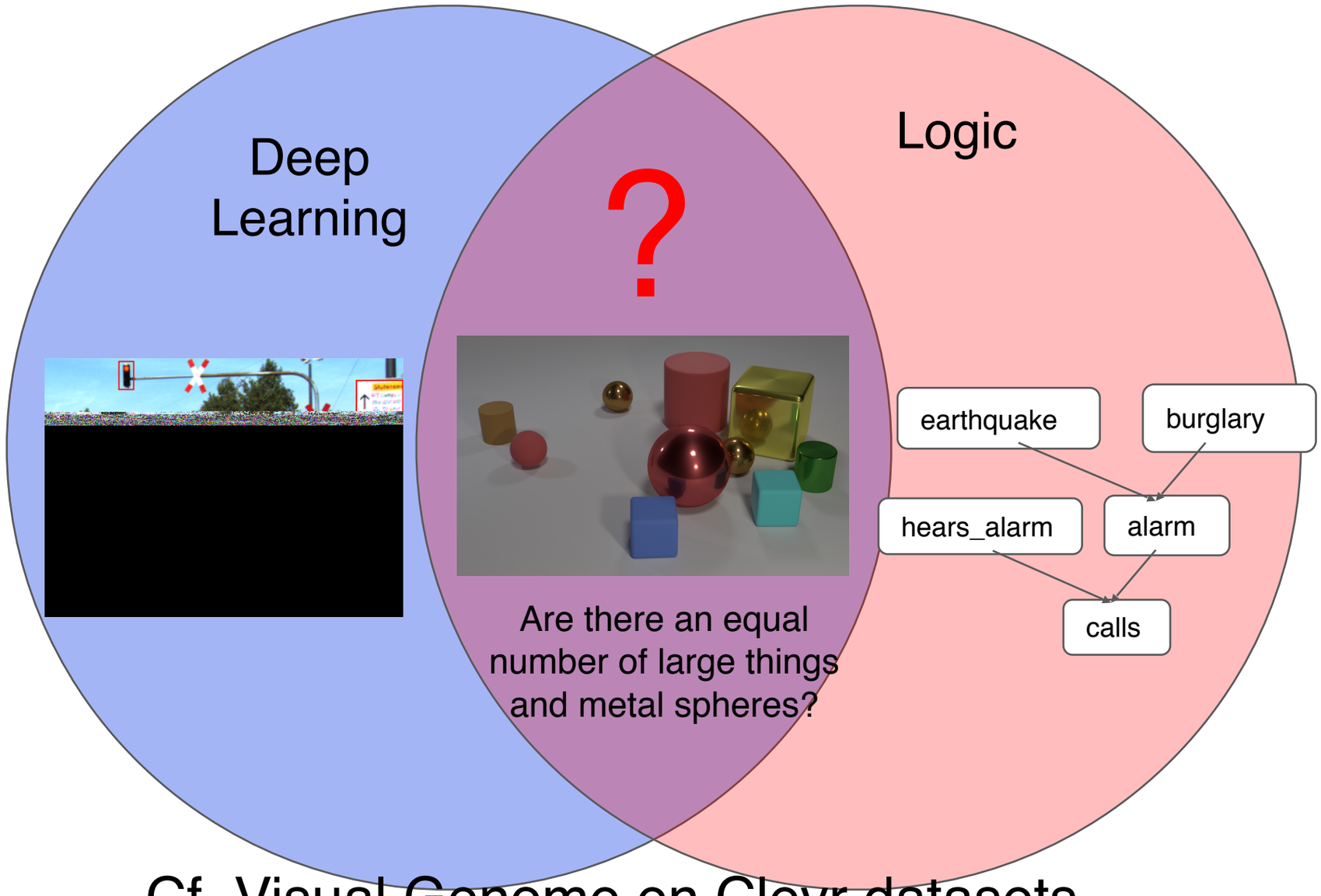
Roadmap

- Prob. Programming - Modeling
- Inference
- Learning
- Dynamics
- Logic, Probability and Neural Nets
- KBMC & Markov Logic
- Lifted Inference

... with some detours on the way

Part V: Logic, Probability and Deep Networks

Integrate Deep Learning and (Probabilistic) Logics ?



Cf. Visual Genome en Clevr datasets

Neural-symbolic learning and reasoning: A survey and interpretation.[Besold et all]

State-of-the-art

- Encoding logic in the weights of neural networks
  - *Logic Tensor Networks* (Serafini et al.)
  - *A Semantic Loss Function for Deep Learning with Symbolic Knowledge* (Xu et al.)
  - *Ontology Reasoning with Deep Neural Networks* (Hohenecker et al.)
  - *Semantic Based Regularization* (Diligenti et al.)

State-of-the-art

- Templates for neural networks (ako Knowledge Base Model Construction)
  - *Lifted Relational Neural Networks* (Šourek et al.)
  - *Neural Theorem Prover* (Rocktäschel et al.)
  - *Neural Module Networks* (Andreas et al.)

State-of-the-art

- Differentiable neural computers / program interpreters
  - *Differentiable Neural Computer* (Graves et al.)
  - *Neural Programmer-Interpreters* (Reed et al.)
  - *Differentiable Forth Interpreter* (Bošnjak et al.)

DeepProbLog NeurIPS 2018

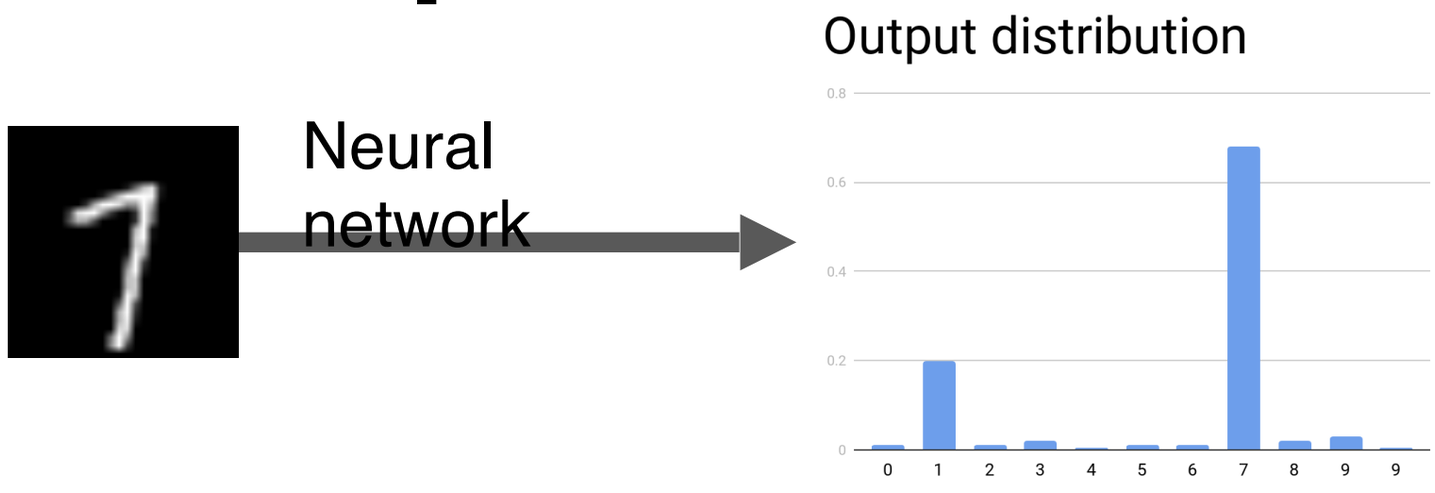
Idea: inject neural networks into logic by extending an existing PLP language

DeepProbLog = ProbLog + neural predicate

The neural predicate makes neural networks a first-class citizen

Related work	**DeepProbLog**
Logic is made less expressive	Full expressivity is retained
Logic is pushed into the neural network	Clean separation
Fuzzy logic	Probabilistic logic
Language semantics unclear	Clear semantics

Neural predicate



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

Example

Learn to classify the sum of pairs of MNIST digits

Individual digits are not labeled!

E.g. ( ,  , 8)

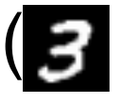
Could be done by a CNN: classify the concatenation of both images into 19 classes

However: 

in DeepProbLog

addition(Im1,Im2,Sum) :- digit(Im1, N1), digit(Im2, N2), Sum is N1 + N2.

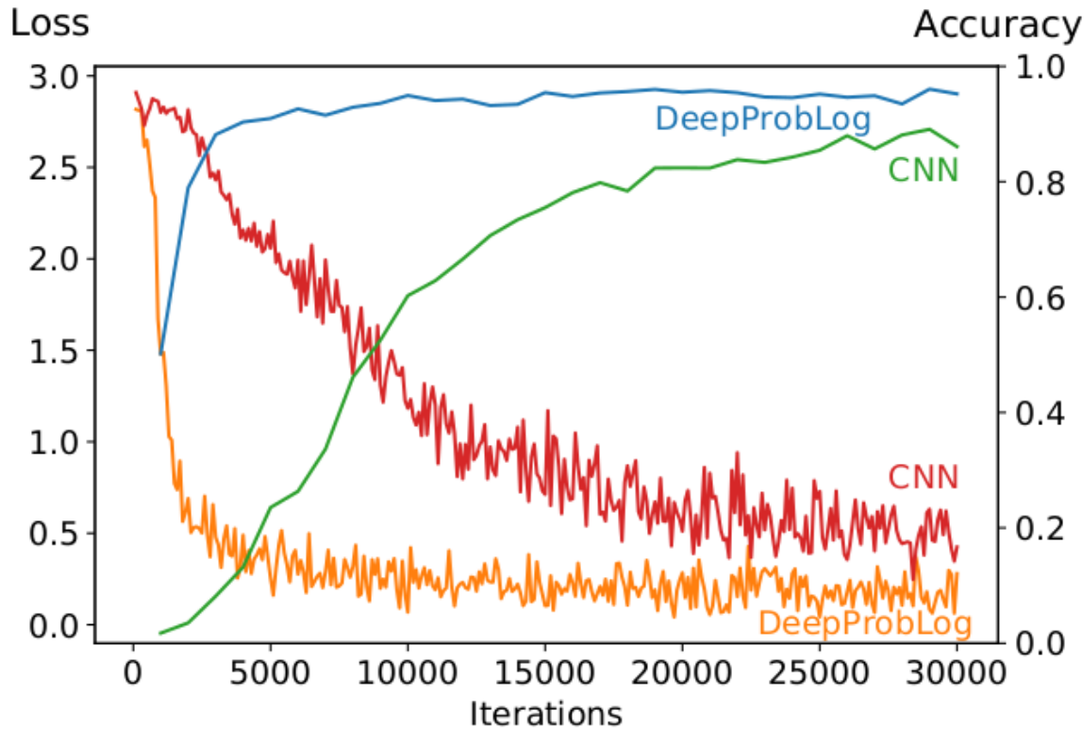
E.g:

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

digit/2 is a neural predicate that **learns** to classify MNIST digits

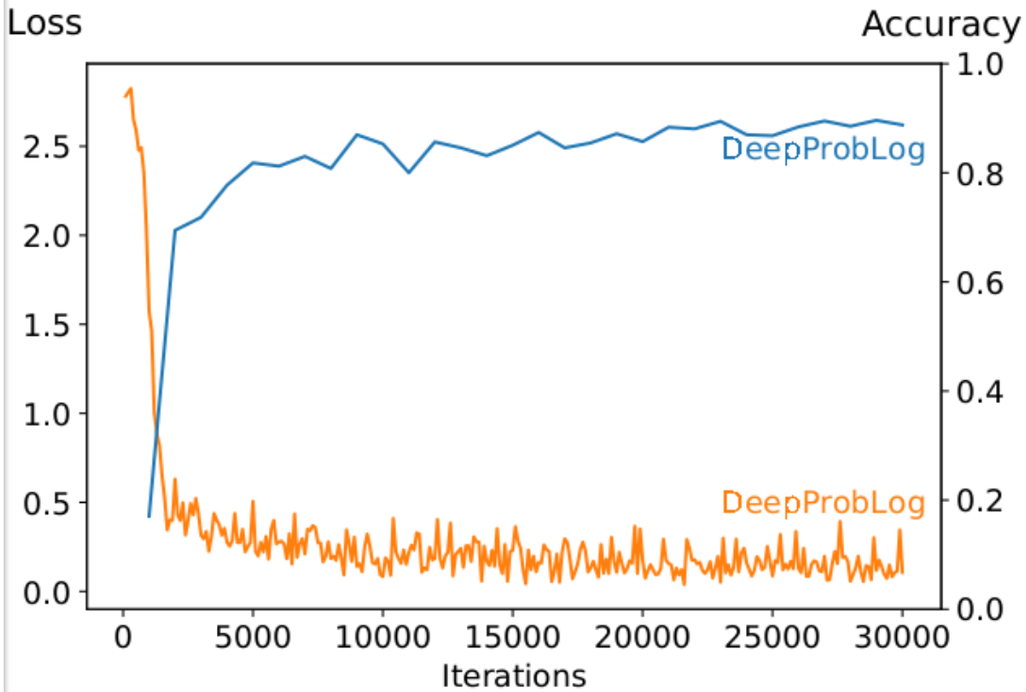
MNIST Addition

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



Multi-digit MNIST addition with MNIST

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

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



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

Program Induction

- Approach similar to that of '*Programming with a Differentiable Forth Interpreter*' [1] (∂4)
  - Partially defined Forth program with slots / holes
    - Slots are filled by neural network (encoder / decoder)
    - Fully differentiable interpreter: NNs are trained with input / output examples
- DeepProbLog program with switches
  - Switches are controlled by neural networks

[1]: Matko Bosnjak, Tim Rocktäschel, Jason Naradowsky, Sebastian Riedel: Programming with a Differentiable Forth Interpreter. ICML 2017: 547-556

Tasks[1]

- Sorting
  - Sort lists of numbers using Bubble sort
  - Hole: Swap or don’t swap when comparing two numbers
- Addition
  - Add two numbers and a carry
  - Hole: What is the resulting digit and carry on each step
  - (Note: not MNIST digits, but actual numbers)
- Word Algebra Problems
  - E.g. “Ann has 8 apples. She buys 4 more. She distributes them equally among her 3 kids. How many apples does each child receive?
  - Hole: Sequence of permuting, swapping and performing operations on the three numbers

[1]: Matko Bosnjak, Tim Rocktäschel, Jason Naradowsky, Sebastian Riedel: Programming with a Differentiable Forth Interpreter. ICML 2017: 547-556

Example DeepProbLog solution

```
hole(X,Y,X,Y):-
    swap(X,Y,0).

hole(X,Y,Y,X):-
    swap(X,Y,1).
```

Holes defined by neural predicate

```
bubble([X],[],X).
bubble([H1,H2|T],[X1|T1],X):-
    hole(H1,H2,X1,X2),
    bubble([X2|T],T1,X).

bubblesort([],L,L).

bubblesort(L,L3,Sorted) :-
    bubble(L,L2,X),
    bubblesort(L2,[X|L3],Sorted).

sort(L,L2) :- bubblesort(L,[],L2).
```

Bubble sort implementation

Result

		Sorting: Training length					Addition: training length		
	Test Length	2	3	4	5	6	2	4	8
$\partial 4$ [Bošnjak et al., 2017]	8	100.0	100.0	49.22	–	–	100.0	100.0	100.0
	64	100.0	100.0	20.65	–	–	100.0	100.0	100.0
DeepProbLog	8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	64	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

(a) Accuracy on the sorting and addition problems (results for $\partial 4$ reported by Bošnjak et al. [2017]).

Training length ⟶	2	3	4	5	6
$\partial 4$ on GPU	42 s	160 s	–	–	–
$\partial 4$ on CPU	61 s	390 s	–	–	–
DeepProbLog on CPU	11 s	14 s	32 s	114 s	245 s

(b) Time until 100% accurate on test length 8 for the sorting problem.

Table 1: Results on the Differentiable Forth experiments

Coin-ball problem

- Play a game where:
  - You flip a biased coin
  - You take a ball from each urn
    - The first urn has red and blue balls
    - The second urn has red, green and blue balls
  - You win if:
    - The two balls have the same colour
    - The coin comes up heads and at least one ball is red
- What needs to be learned:
  - Classify the coin as heads / tails (even / uneven MNIST digits are used for the coins)
  - Classify the color (RGB triplet) as either red, green or blue
  - The bias of the coin
  - The distribution of the balls in the urns

Coin-ball problem

- This example showcases join learning of:
  - Neural networks
  - Probabilistic parameters in the logic program
- Result: after 5 epochs of 256 examples we managed to:
  - Learn to classify the coins
  - Learn to classifty the colours
  - Learned the bias of the coin

Related work

	First-order logic	Logic program	Train neural network	Learn logic parameters	Program induction	Probabilistic logic	Embeddings
Differentiabel Forth Interpreter			V		V		V
End-to-end differentiable proving		V			V		V
Semantic Loss	V		V				
Logic Tensor Networks	V						V
Semantic Based Regularization	V		V				
DeepProbLog		V	V	V	V	V	V

How does it work ?

- We use our algebraic ProbLog and use the gradient semi-ring
- What is aProbLog ?
  - a version of ProbLog where the probabilistic semi-ring is replaced by an arbitrary semiring structure
  - labels on facts are elements of the semiring
  - cf. the different semi-rings for the WMC, #SAT, …

more examples

semiring label function

task	$\mathcal{A}$	$e^{\oplus}$	$e^{\otimes}$	$a \oplus b$	$a \otimes b$	$\alpha(f)$	$\alpha(\neg f)$
PROB	$\mathbb{R}_{\geq 0}$	0	1	$a+b$	$a \cdot b$	$\alpha(f_i) \in [0,1]$	$1-\alpha(f_i)$
MPE	$\mathbb{R}_{\geq 0}$	0	1	$\max(a,b)$	$a \cdot b$	$\alpha(f_i) \in [0,1]$	$1-\alpha(f_i)$
MPE State	$\mathbb{R}_{\geq 0} \times 2^{\{J \mid J \subseteq I \in \mathcal{I}(\mathrm{F})\}}$	$(0,\emptyset)$	$(1,\{\emptyset\})$	Eq. (8)	Eq. (7)	$\alpha(f_i) = (p_i, \{\{f_i\}\})$ with $p_i \in [0,1]$	$(1-p_i, \{\{\neg f_i\}\})$
SAT	$\{true, false\}$	$false$	$true$	$a \vee b$	$a \wedge b$	$\alpha(f_i) = true$	$true$
#SAT	$\mathbb{N}$	0	1	$a+b$	$a \cdot b$	$\alpha(f_i) = 1$	1
BDD	$\mathcal{BDD}(\mathcal{V})$	$bdd(0)$	$bdd(1)$	$a \vee_{bdd} b$	$a \wedge_{bdd} b$	$\alpha(f_i) = bdd(b_i)$	$\neg_{bdd} bdd(b_i)$
sensitivity	$\mathbb{R}[X]$	0	1	$a+b$	$a \cdot b$	$\alpha(f_i) = x$ or $\alpha(f_i) \in [0,1]$	$1-\alpha(f_i)$
gradient	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	$(0,0)$	$(1,0)$	Eq. (9)	Eq. (10)	Eq. (11)	Eq. (12)

$$(p,S) \otimes (q,T) = (p \cdot q, \{I \cup J \mid I \in S, J \in T\}) \quad (7)$$

$$(p,S) \oplus (q,R) = \begin{cases} (p,S) & \text{if } p > q \\ (q,R) & \text{if } p < q \\ (p, S \cup R) & \text{if } p = q \end{cases} \quad (8)$$

$$(a_1,a_2) \oplus (b_1,b_2) = (a_1+b_1, a_2+b_2) \quad (9)$$

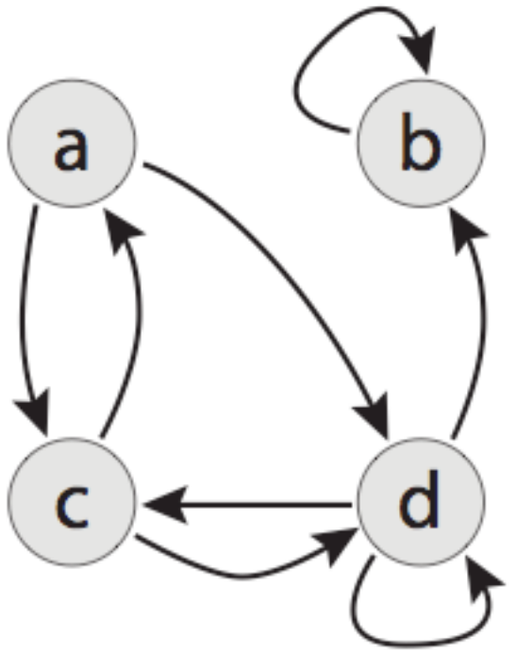
$$(a_1,a_2) \otimes (b_1,b_2) = (a_1 \cdot b_1, a_1 \cdot b_2 + a_2 \cdot b_1) \quad (10)$$

$$\alpha(f_i) = \begin{cases} (p_i, 1) & \text{if } i = k \\ (p_i, 0) & \text{if } i \neq k \end{cases} \quad (11)$$

$$\alpha(\neg f_i) = \begin{cases} (1-p_i, -1) & \text{if } i = k \\ (1-p_i, 0) & \text{if } i \neq k \end{cases} \quad (12)$$

```
reachable(Q) :- initial(Q).
reachable(Q) :- reachable(P), edge(P, Q).
```

Fig. 1. A simple bottom-up logic program for graph reachability



$$
\begin{aligned}
&\texttt{initial(a)} = \mathrm{T} \quad \texttt{edge(c, d)} = \mathrm{T} \\
&\texttt{edge(a, c)} = \mathrm{T} \quad \texttt{edge(d, b)} = \mathrm{T} \\
&\texttt{edge(a, d)} = \mathrm{T} \quad \texttt{edge(d, c)} = \mathrm{T} \\
&\texttt{edge(b, b)} = \mathrm{T} \quad \texttt{edge(d, d)} = \mathrm{T} \\
&\texttt{edge(c, a)} = \mathrm{T}
\end{aligned}
$$

Fig. 2. A directed graph and the corresponding initial database

[Figures: Cohen et al, ICLP 08]

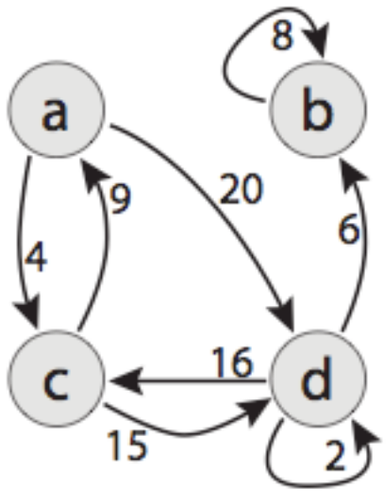
$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{initial}(\texttt{Q}).$$

$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{reachable}(\texttt{P}) \otimes \texttt{edge}(\texttt{P}, \texttt{Q}).$$

$\oplus \rightarrow \min$

$\otimes \rightarrow +$

shortest path



$\texttt{initial}(\texttt{a}) = 0$
$\texttt{edge}(\texttt{a}, \texttt{c}) = 4$
$\texttt{edge}(\texttt{a}, \texttt{d}) = 20$
$\texttt{edge}(\texttt{b}, \texttt{b}) = 8$
$\texttt{edge}(\texttt{c}, \texttt{a}) = 9$
$\texttt{edge}(\texttt{c}, \texttt{d}) = 15$
$\texttt{edge}(\texttt{d}, \texttt{b}) = 6$
$\texttt{edge}(\texttt{d}, \texttt{c}) = 16$
$\texttt{edge}(\texttt{d}, \texttt{d}) = 2$

Fig. 3. A cost graph and the corresponding initial database

$\oplus \rightarrow \max$

$\otimes \rightarrow \cdot$

most likely path

$\oplus \rightarrow +$

$\otimes \rightarrow \cdot$

PRISM

[Figures: Cohen et al, ICLP 08]

Algebraic Prolog (aProbLog)

- commutative semiring $(A,\oplus,\otimes,e^{\oplus},e^{\otimes})$
- algebraic ground literals
  $L(F) = \{f_1,...,f_n\} \cup \{\neg f_1,...,\neg f_n\}$
- background knowledge clauses
- labeling function $\alpha: L(F) \rightarrow A$
- disjoint sum if $l(f) \oplus l(f) = l(f)$
- neutral sum if $l(f) \oplus f(\neg f) = e^{\otimes}$

$$L(\text{query}) = \bigoplus_{\text{worlds}} \bigotimes_{\text{literals}} \alpha(l)$$

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

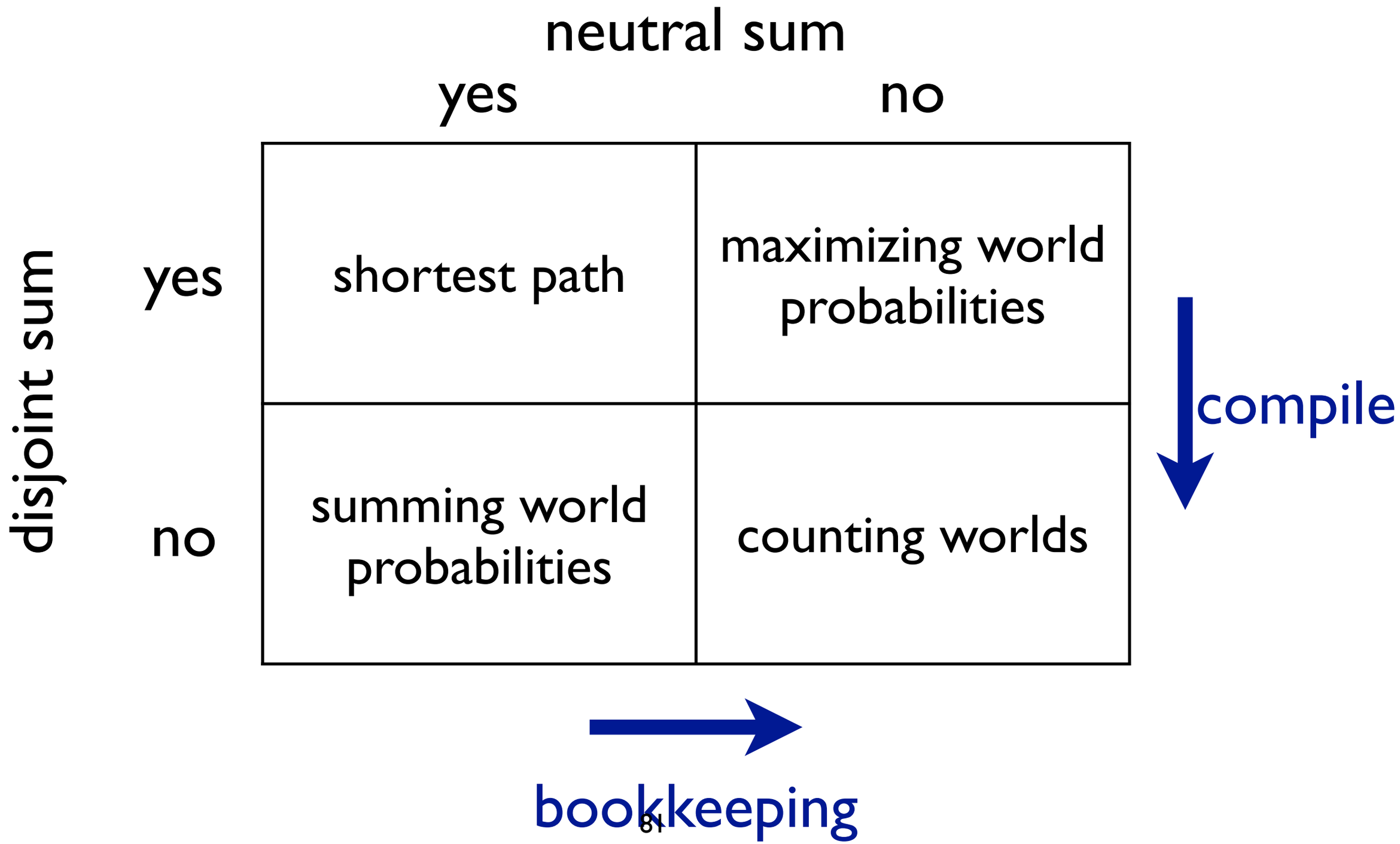
- Prolog + labeled facts
- labels from commutative semiring (e.g. probabilities, costs, polynomials, Boolean functions, datastructures, ...)
- inference based on ProbLog principles

one proof - several worlds

one world - several proofs

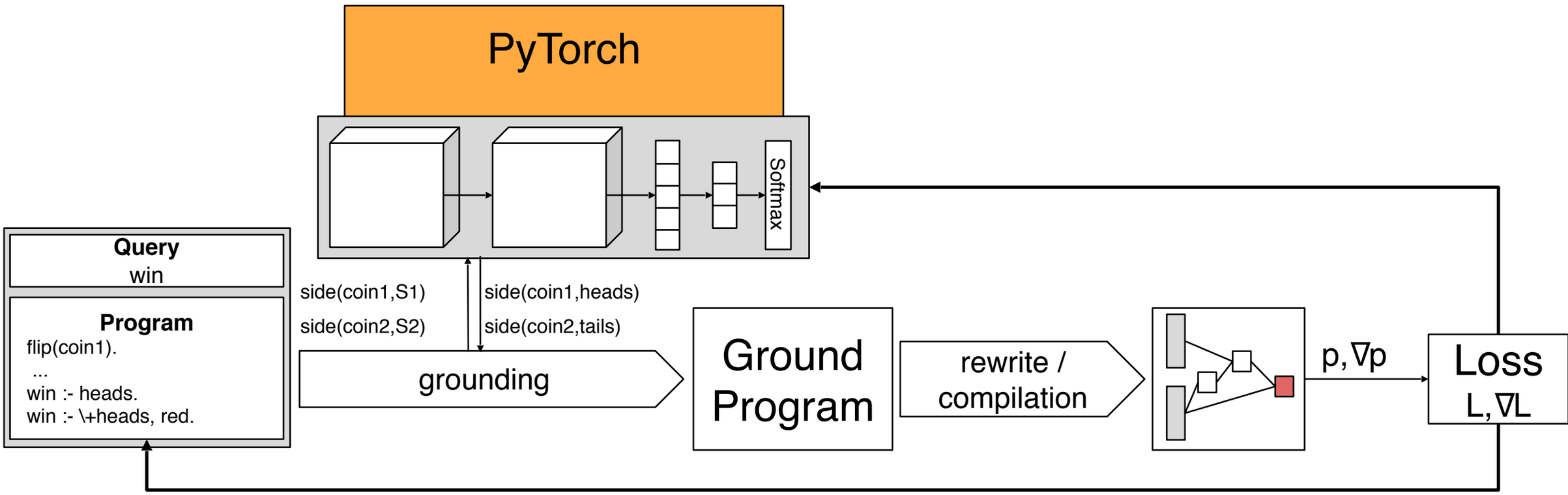
	neutral sum	NSP
disjoint sum	**SAT**	**MPE**
DSP	**PROB**	**#SAT**

Inference settings



Architecture

- ProbLog 2
- Deep Learning Framework: PyTorch



Roadmap

- Prob. Programming - Modeling
- Inference
- Learning
- Dynamics
- Logic, Probability and Neural Nets
- KBMC & Markov Logic
- Lifted Inference

... with some detours on the way

Part VI: KBMC, Markov Logic

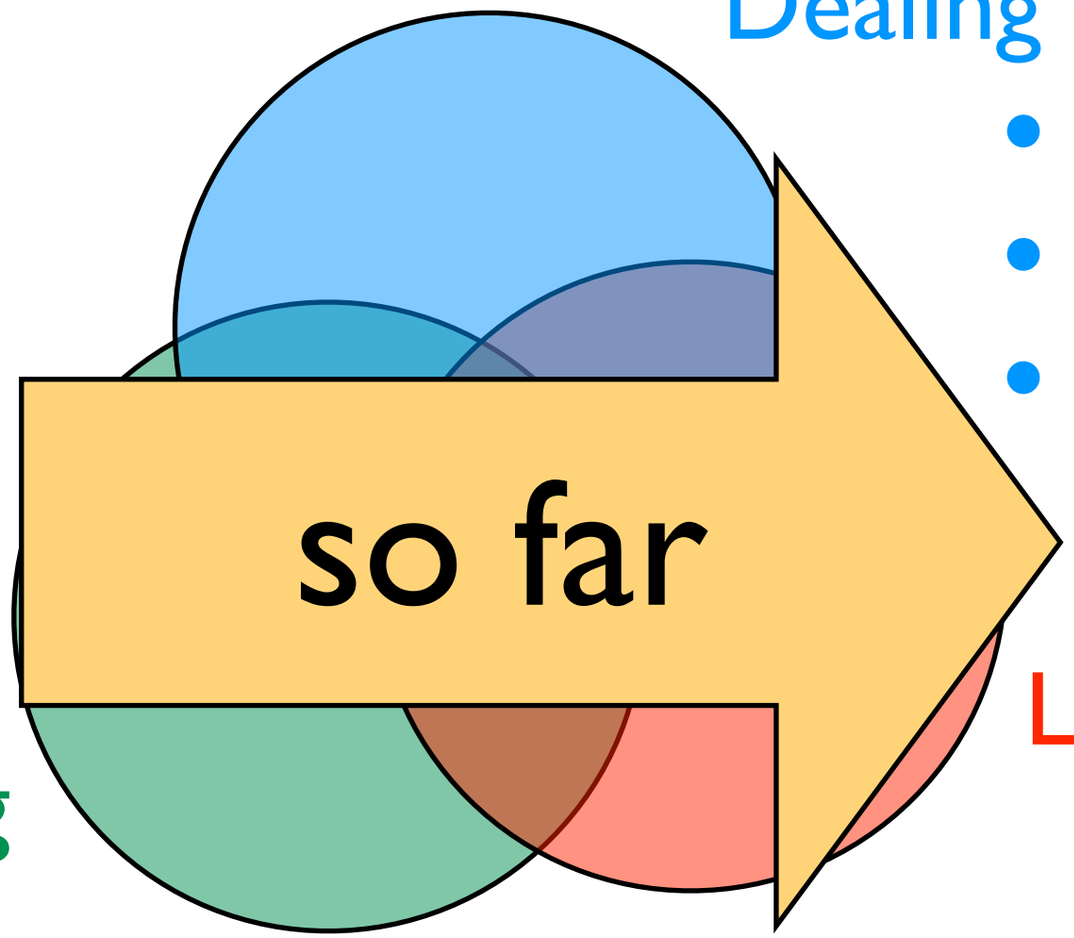
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

- parameters
- structure

Statistical relational learning
& Probabilistic programming, ...

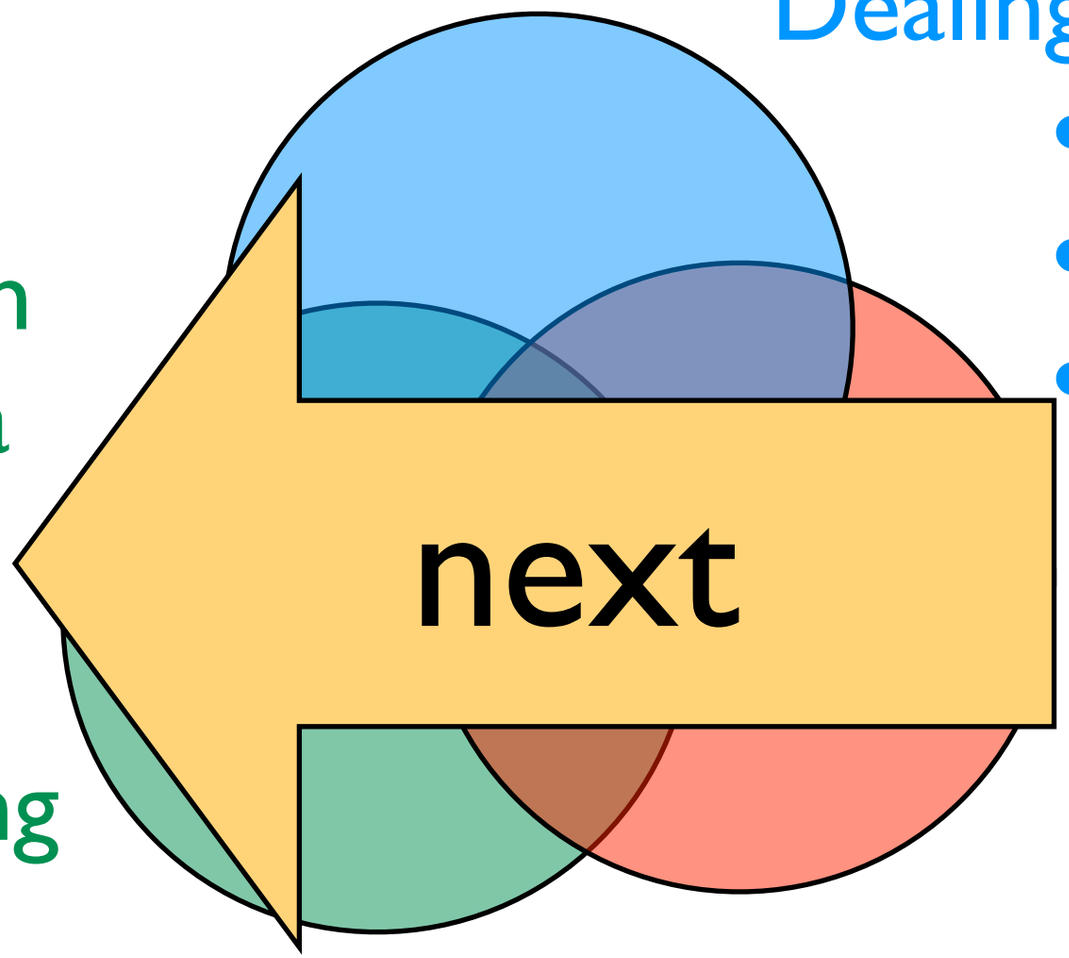
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...

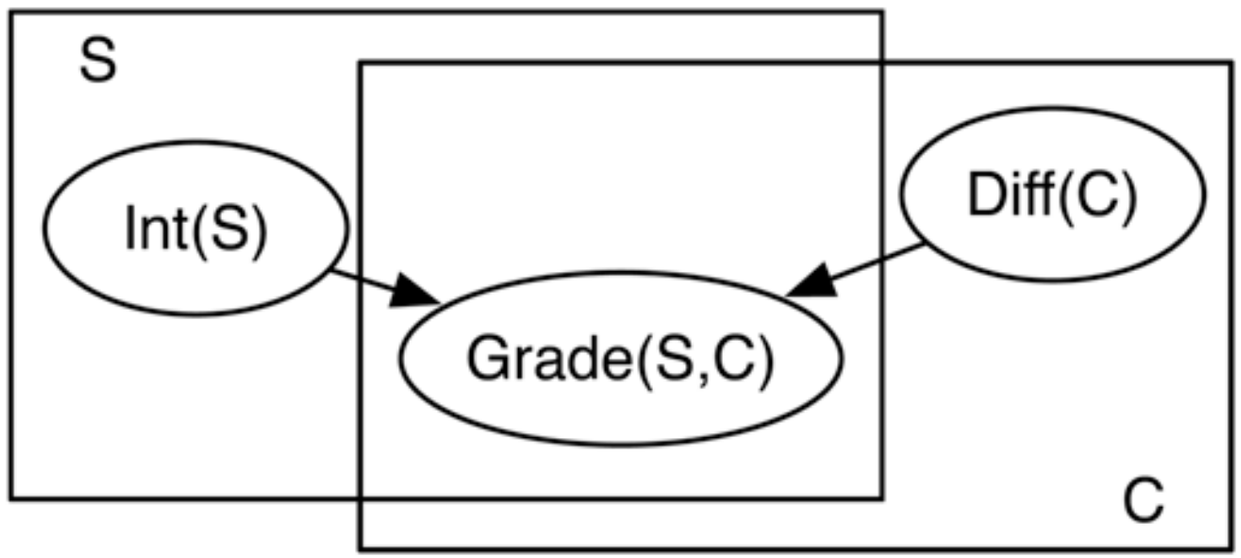


Learning

- parameters
- structure

Statistical relational learning
& Probabilistic programming, ...

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

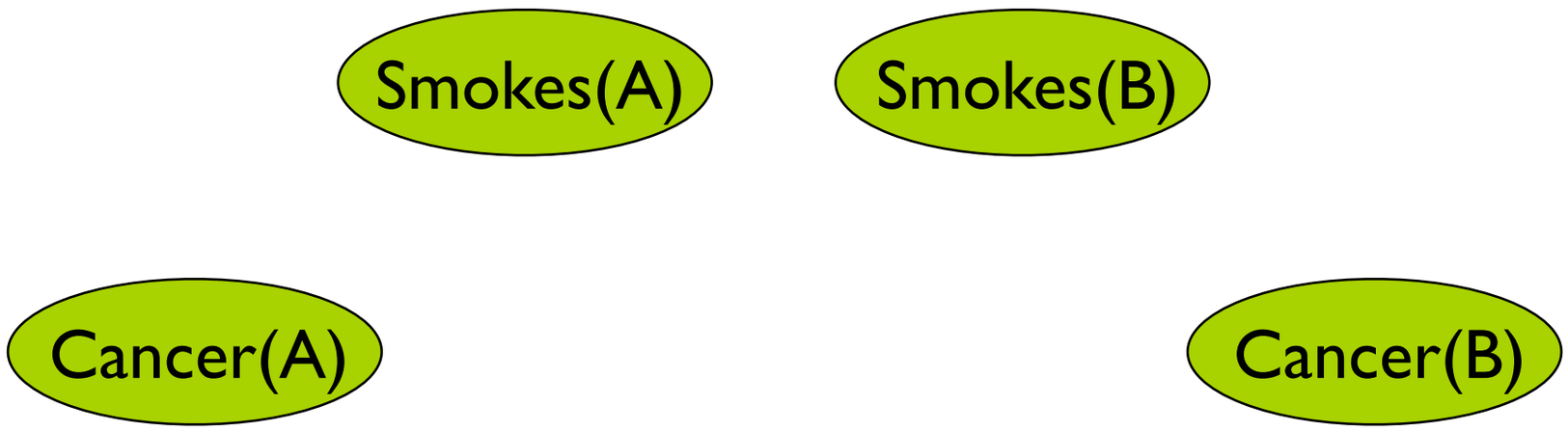
Lots of proposals in the literature, e.g.

- relational Markov networks (RMNs) [Taskar et al 2002]
- **Markov logic networks (MLNs)** [Richardson & Domingos 2006]
- *probabilistic soft logic (PSL) [Broecheler et al 2010]*
- FACTORIE [McCallum et al 2009]
- Bayesian logic programs (BLPs) [Kersting & De Raedt 2001]
- relational Bayesian networks (RBNs) [Jaeger 2002]
- logical Bayesian networks (LBNs) [Fierens et al 2005]
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- and many more ...

Markov Logic

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

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

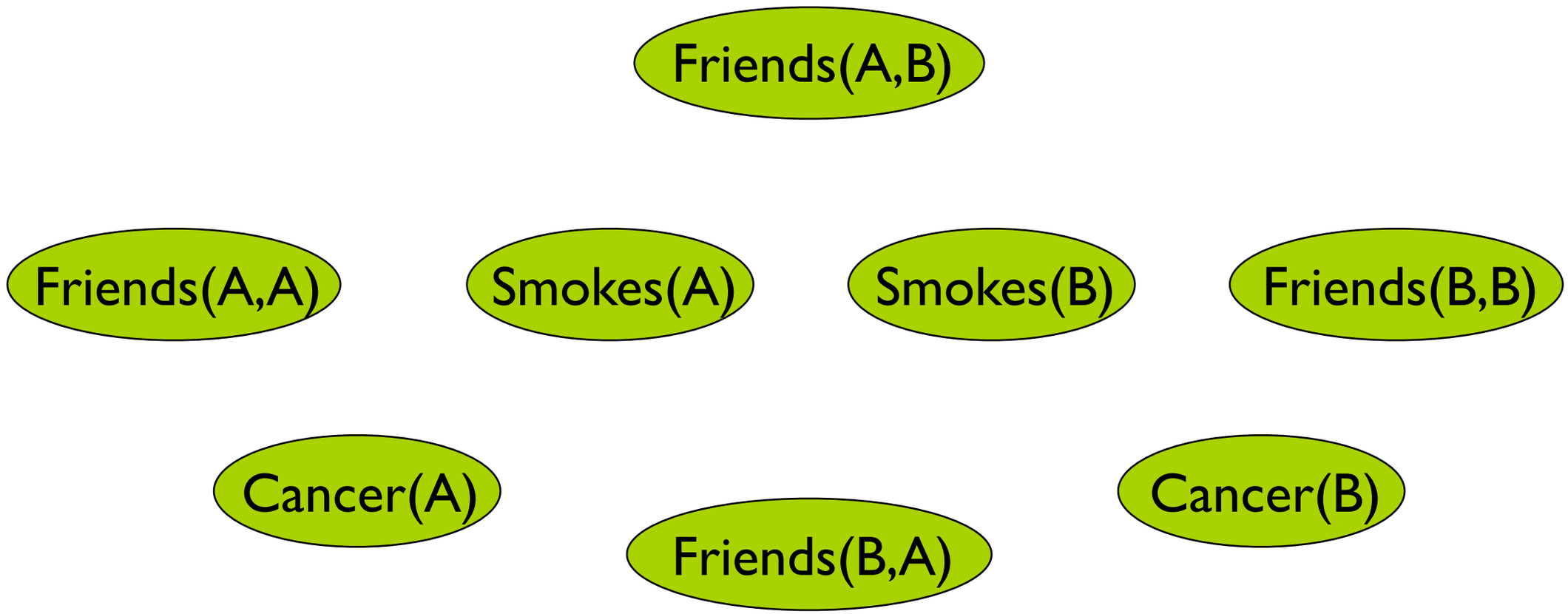


slides by Pedro Domingos

Markov Logic

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

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

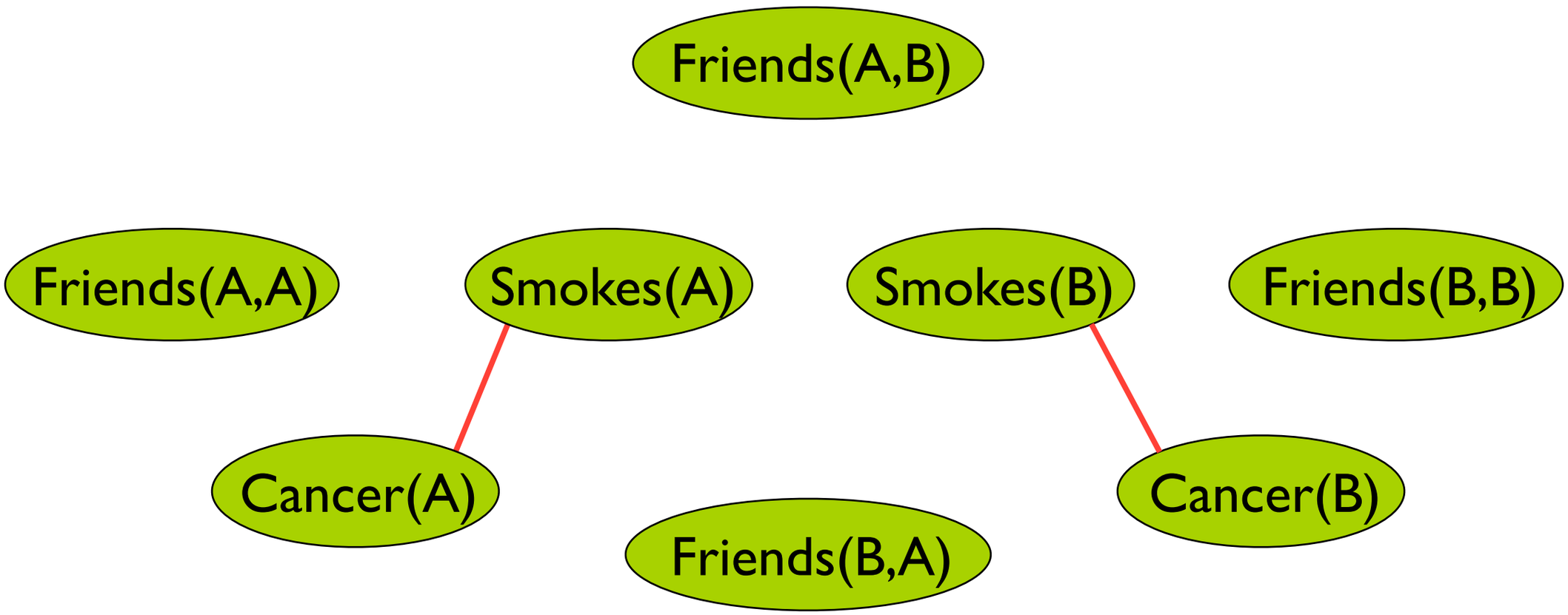


slides by Pedro Domingos

Markov Logic

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

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

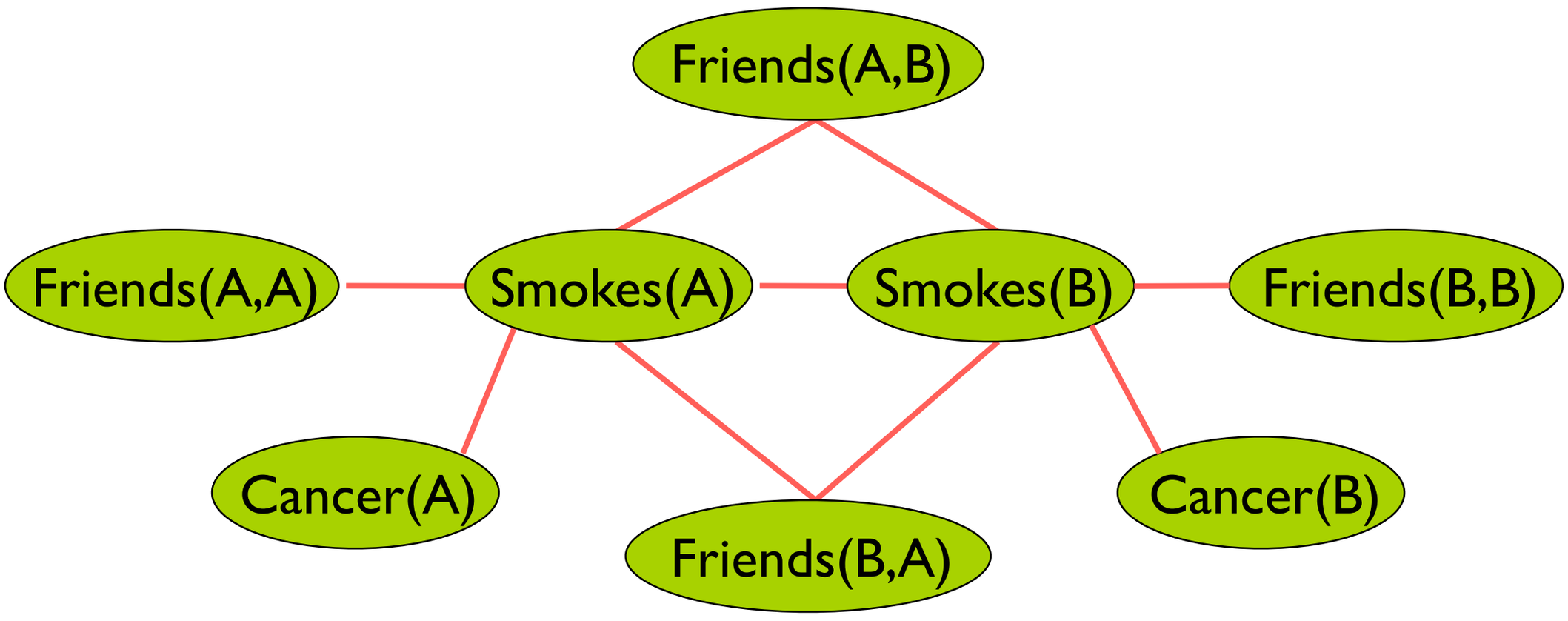


slides by Pedro Domingos

Markov Logic

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

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



slides by Pedro Domingos

Possible Worlds

A vocabulary

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

Possible worlds
Logical interpretations

Slides adapted from Guy Van den Broeck

Possible Worlds

A logical theory

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

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

Interpretations that satisfy the theory
Models

Slides adapted from Guy Van den Broeck

First-Order Model Counting

A logical theory

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

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

Σ

First-order model count
~#SAT

Slides Guy Van den Broeck

Markov Logic

- MLNs are a template for ground Markov Networks
- Probability of a world/interpretation
- If $n_i = 0$ then $P(x) = \frac{1}{Z}$

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

Weight of formula *i*

No. of true groundings of formula *i* in *x*

Markov Logic

A Markov Logic theory

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

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

Slides adapted from Guy Van den Broeck

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

Markov Logic

A Markov Logic theory

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

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

Σ

Z
partition function

Slides adapted from Guy Van den Broeck

Weighted First-Order Model Counting

A logical theory and a weight function for predicates

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

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

Σ

Weighted first-order model count

Related to ProbLog Inference !

Parameter Learning

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

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

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

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

Roadmap

- Prob. Programming - Modeling
- Inference
- Learning
- Dynamics
- Logic, Probability and Neural Nets
- KBMC & Markov Logic
- Lifted Inference

... with some detours on the way

PART VII :

Lifted Inference

Lifted Inference

- avoid the grounding step where possible
- exploiting symmetries & repeated structure
- reasoning on first order level as much as possible
- aiming at independence from number of objects
- approximation: grouping similar computations
- very active research area

Example:
First-Order Model Counting

1. Logical sentence

$$\text{Stress}(\text{Alice}) \Rightarrow \text{Smokes}(\text{Alice})$$

Domain

Alice

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

Stress(Alice) ⇒ Smokes(Alice)

Domain

Alice

Stress(Alice)	Smokes(Alice)	Formula
0	0	1
0	1	1
1	0	0
1	1	1

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

Stress(Alice) ⇒ Smokes(Alice)

Domain

Alice

→ 3 models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

Stress(Alice) ⇒ Smokes(Alice)

Domain

Alice

→ 3 models

2. Logical sentence

∀x, Stress(x) ⇒ Smokes(x)

Domain

Alice

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

$$\text{Stress}(\text{Alice}) \Rightarrow \text{Smokes}(\text{Alice})$$

Domain

Alice

→ 3 models

2. Logical sentence

$$\forall x,\ \text{Stress}(x) \Rightarrow \text{Smokes}(x)$$

Domain

Alice

→ 3 models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

2. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

$\rightarrow$ 3 models

Example:
First-Order Model Counting

2. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

$\rightarrow$ 3 models

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

Slides Guy Van den Broeck

Example:
First-Order Model Counting

2. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

$\rightarrow$ 3 models

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

3. Logical sentence

$\forall x$, Stress(x) $\Rightarrow$ Smokes(x)

Domain

n people

$\rightarrow 3^n$ models

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

if Female:

$\forall y, \text{Parent}(y) \Rightarrow \text{Mother}(y)$

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

if not Female:

True

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

Slides Guy Van den Broeck

Example: First-Order Model Counting

4. Logical sentence

$\forall y,\ \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

Example: First-Order Model Counting

4. Logical sentence

$\forall y,\ \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

5. Logical sentence

$\forall x,y,\ \text{ParentOf}(x,y) \wedge \text{Female}(x) \Rightarrow \text{MotherOf}(x,y)$

Domain

n people

Slides Guy Van den Broeck

Example: First-Order Model Counting

4. Logical sentence

$\forall y,\ \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

5. Logical sentence

$\forall x,y,\ \text{ParentOf}(x,y) \wedge \text{Female}(x) \Rightarrow \text{MotherOf}(x,y)$

Domain

n people

$\rightarrow (3^n+4^n)^n$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

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

Domain

n people

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

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

Domain

n people

Slides Guy Van den Broeck

Example:
First-Order Model Counting

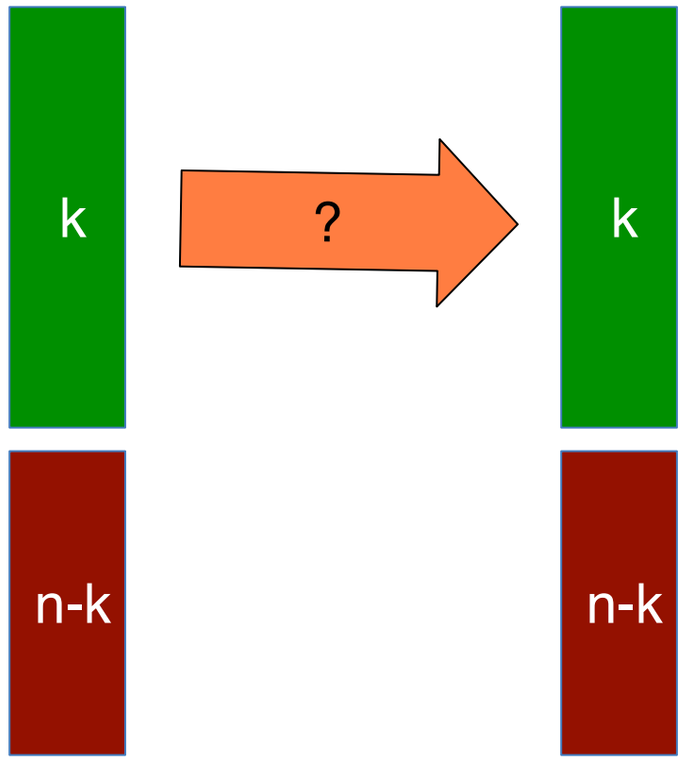
6. Logical sentence

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

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

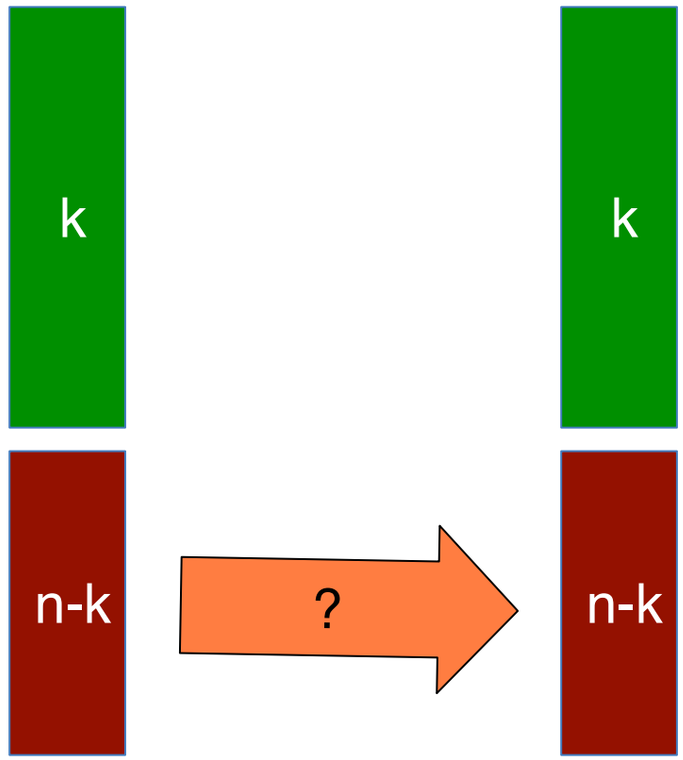
6. Logical sentence

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

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

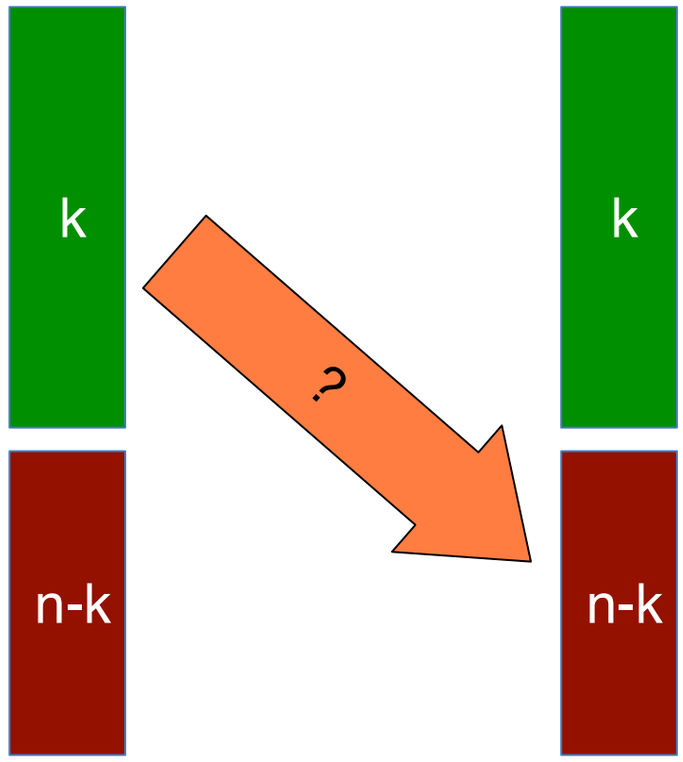
6. Logical sentence

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

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

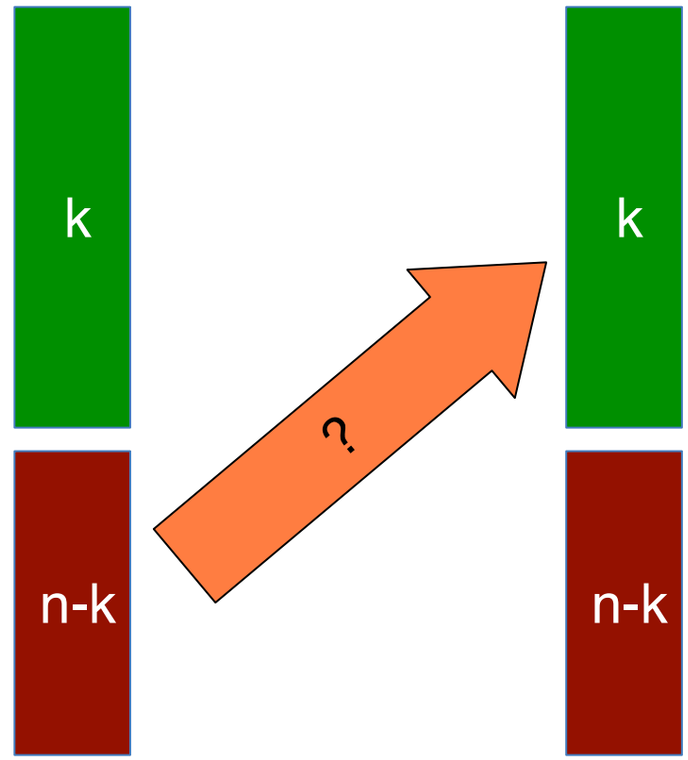
6. Logical sentence

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

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

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

Domain

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

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

Domain

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

Domain

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

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

$\rightarrow \quad \binom{n}{k} 2^{n^2 - k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

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

Domain

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

$\rightarrow \quad \binom{n}{k} 2^{n^2 - k(n-k)}$ models

In total $\sum_{k=0}^{n} \binom{n}{k} 2^{n^2 - k(n-k)}$

Slides Guy Van den Broeck

The Full Pipeline

MLN

3.14 $\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Slides Guy Van den Broeck

The Full Pipeline

MLN

3.14 $\quad \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$



$\forall x,y,\ F(x,y) \Leftrightarrow [\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)\ ]$

Relational Logic

The Full Pipeline

MLN

3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

Relational Logic

Smokes → 1
¬Smokes → 1
Friends → 1
¬Friends → 1
F → exp(3.14)
¬F → 1

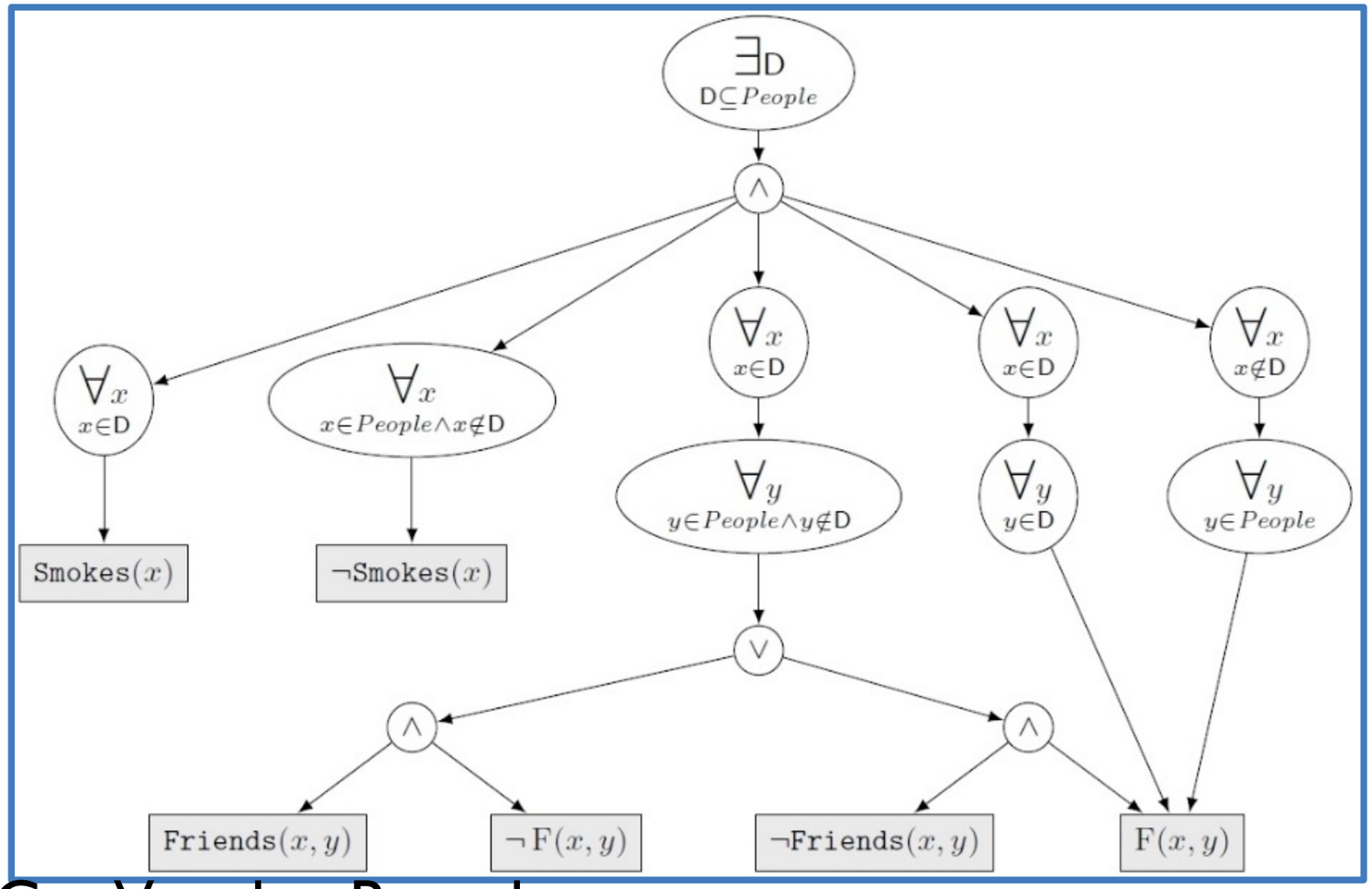
Weight Function

Slides Guy Van den Broeck

The Full Pipeline

$$\forall x,y,\ F(x,y) \Leftrightarrow [\ Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y)\ ]$$

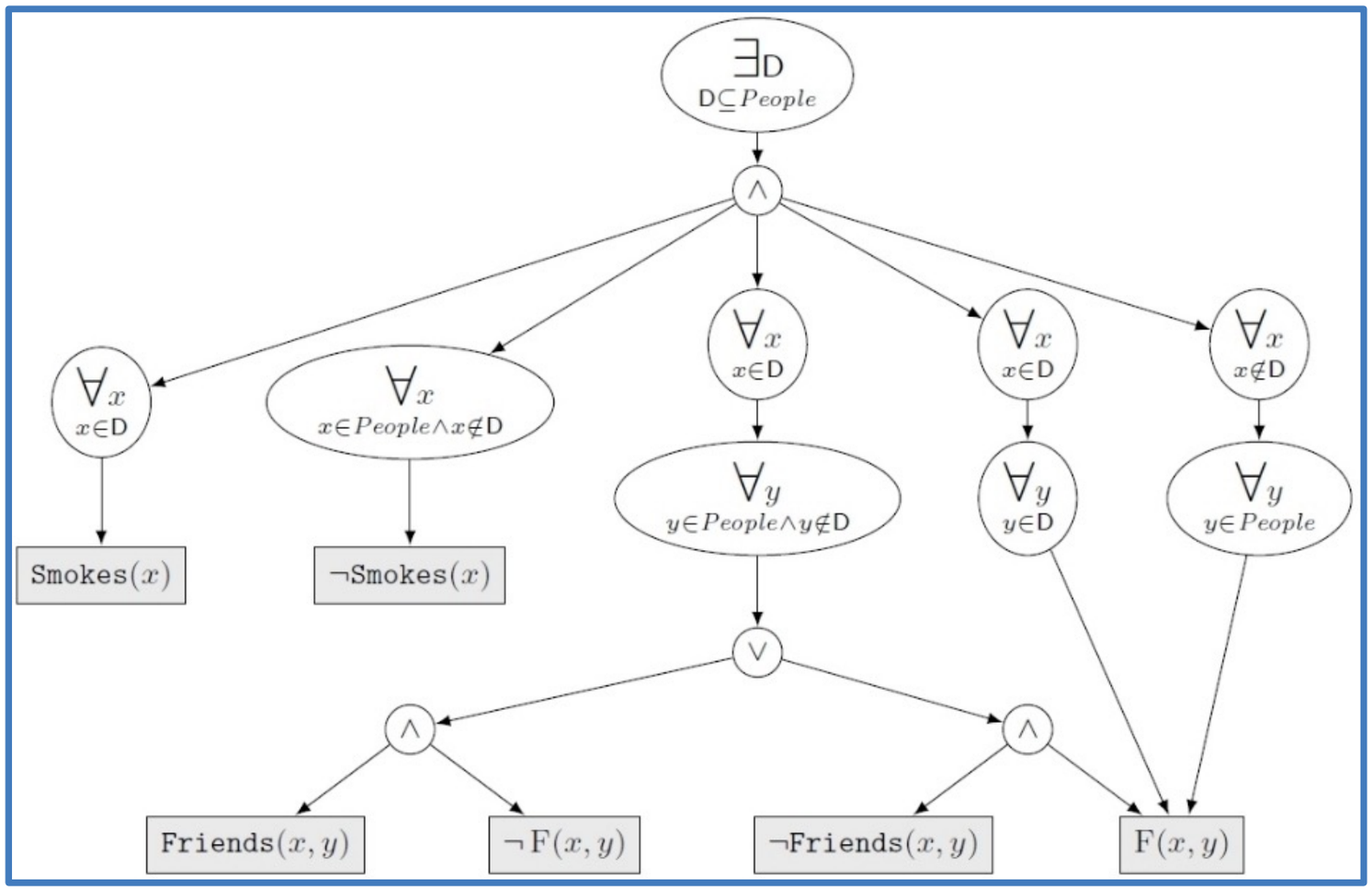
Relational Logic



First-Order
d-DNNF Circuit

Slides Guy Van den Broeck

The Full Pipeline



First-Order d-DNNF Circuit

Smokes → 1
¬Smokes → 1
Friends → 1
¬Friends → 1
F → exp(3.14)
¬F → 1

Weight Function

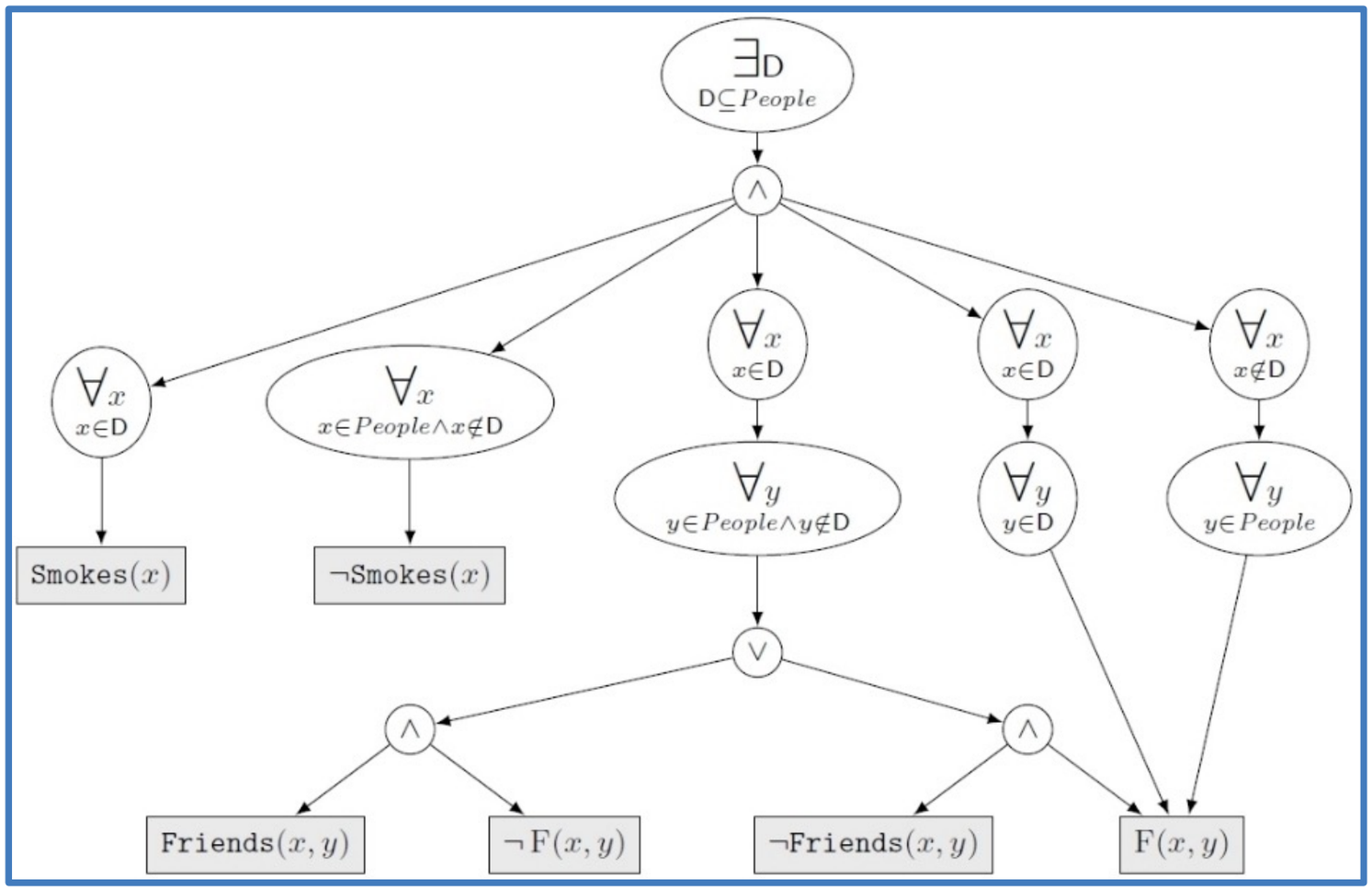
Alice
Bob
Charlie

Domain

Weighted First-Order Model Count is 1479.85

Slides Guy Van den Broeck

The Full Pipeline



First-Order d-DNNF Circuit

Smokes → 1
¬Smokes → 1
Friends → 1
¬Friends → 1
F → exp(3.14)
¬F → 1

Weight Function

Alice
Bob
Charlie

Domain

Weighted First-Order Model Count is 1479.85

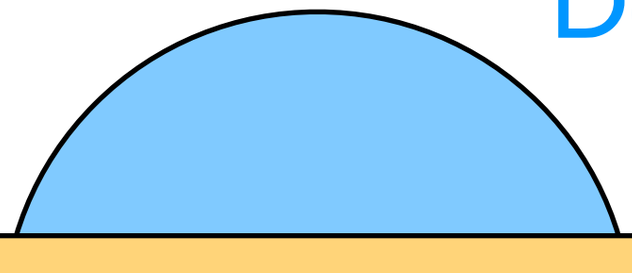
Circuit evaluation is polynomial in domain size!

Slides Guy Van den Broeck

Why are we doing this ?

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

A key question in AI:



Dealing with uncertainty
- probability theory

Reas
relat
- log
- da
- pr
- ...

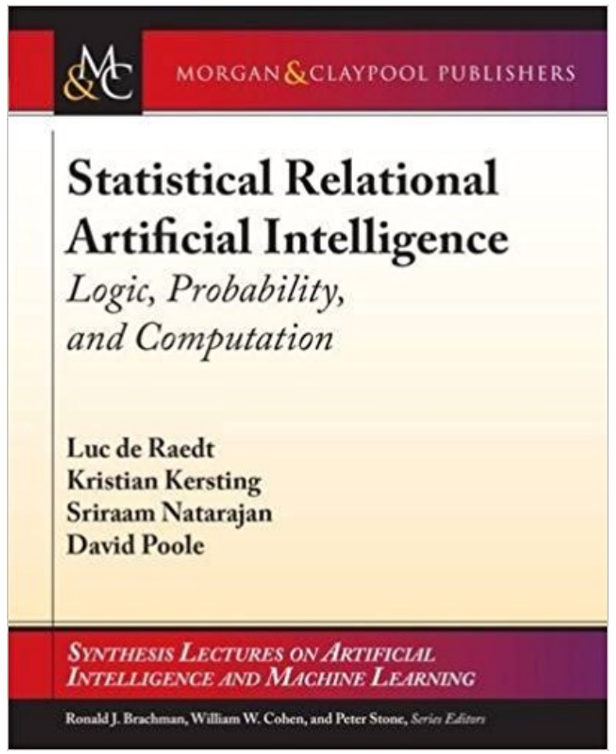
dels

- Our answer: probabilistic (logic) programming
  = probabilistic choices + (logic) program
- Many languages, systems, applications, ...
- ... and much more to do!

structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Further Reading

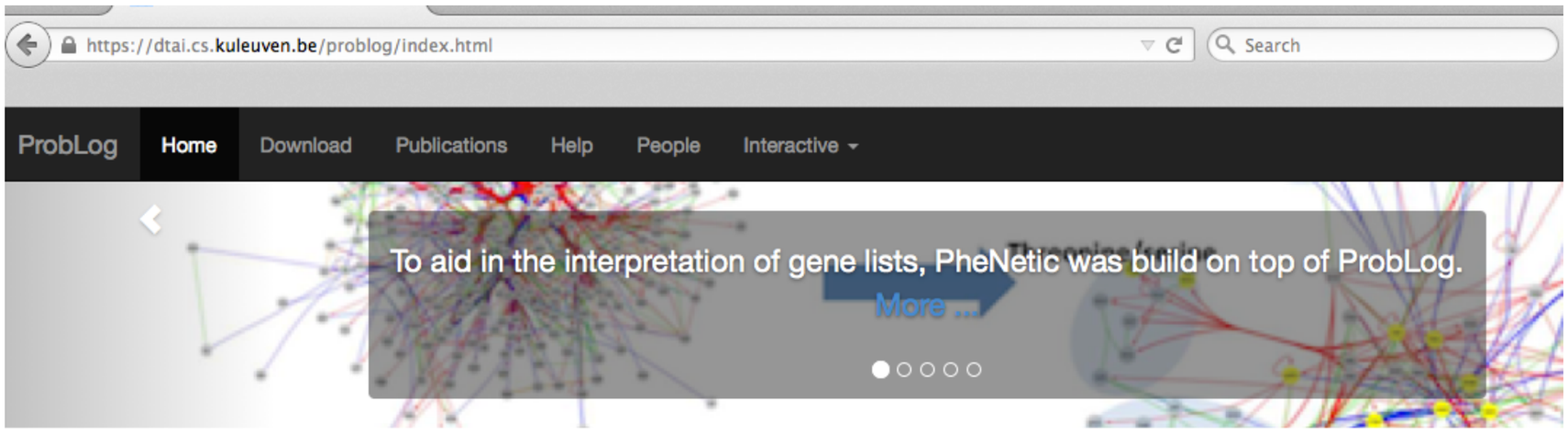


- One book
- Three websites to start
  - http://probmods.org/ Probabilistic Models of Cognition — Church
  - http://dtai.cs.kuleuven.be/problog/ — check also [DR & Kimmig, MLJ 15]
  - http://alchemy.cs.washington.edu/ —Markov Logic, check also [Domingos & Lowd] Markov Logic, Morgan Claypool.

Maurice Bruynooghe
Bart Demoen
Anton Dries
Daan Fierens
Jason Filippou
Bernd Gutmann
Manfred Jaeger
Gerda Janssens
Kristian Kersting
Angelika Kimmig
Theofrastos Mantadelis
Wannes Meert
Bogdan Moldovan
Siegfried Nijssen
Davide Nitti
Joris Renkens
Kate Revoredo
Ricardo Rocha
Vitor Santos Costa
Dimitar Shterionov
Ingo Thon
Hannu Toivonen
Guy Van den Broeck
Mathias Verbeke
Jonas Vlasselaer

Thanks !

http://dtai.cs.kuleuven.be/problog



PLP Systems

- **PRISM** http://sato-www.cs.titech.ac.jp/prism/
- **ProbLog2** http://dtai.cs.kuleuven.be/problog/
- Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  - **ProbLog1**
  - **cplint** https://sites.google.com/a/unife.it/ml/cplint
  - **CLP(BN)**
  - **LP2**
- **PITA** in XSB Prolog http://xsb.sourceforge.net/
- **AILog2** http://artint.info/code/ailog/ailog2.html
- **SLPs** http://stoics.org.uk/~nicos/sware/pepl
- **contdist** http://www.cs.sunysb.edu/~cram/contdist/
- **DC** https://code.google.com/p/distributional-clauses
- **WFOMC** http://dtai.cs.kuleuven.be/ml/systems/wfomc