ECML PKDD 2005
porto › portugal › 3 to 7 october
16th european conference on machine learning
9th european conference on principles and practice of knowledge discovery in databases

Probabilistic Inductive Logic Programming

* L. De Raedt, K. Kersting. "Probabilistic inductive Logic Programming". In S. Ben-David, J. Case and A. Maruoka, editors, Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT-2004), pages 19-36. Padova, Italy, October 2-5, 2004.

* L. De Raedt, K. Kersting. "Probabilisitc Logic Learning". In ACM-SIGKDD Explorations, special issue on Multi-Relational Data Mining, S. Dzeroski and L. De Raedt, editors, Vol. 5(1), pp. 31-48, July 2003.

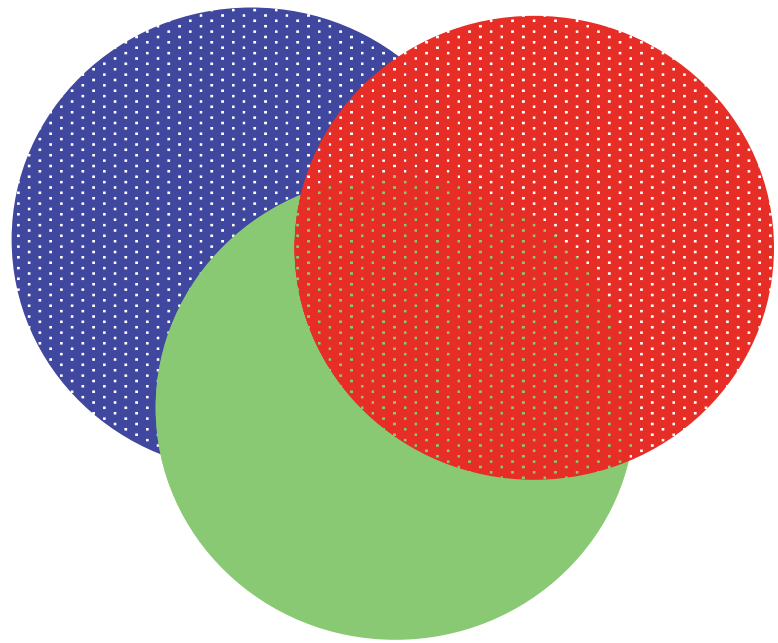
Luc De Raedt, Kristian Kersting
Machine Learning Lab, Institute for Computer Science
University of Freiburg, Germany

Probabilistic Logic Learning*

One of the key open questions of artificial intelligence concerns

"probabilistic logic learning",

i.e. the integration of probabilistic reasoning with first order logic representations and machine learning.



*In the US, sometimes called Statistical Relational Learning

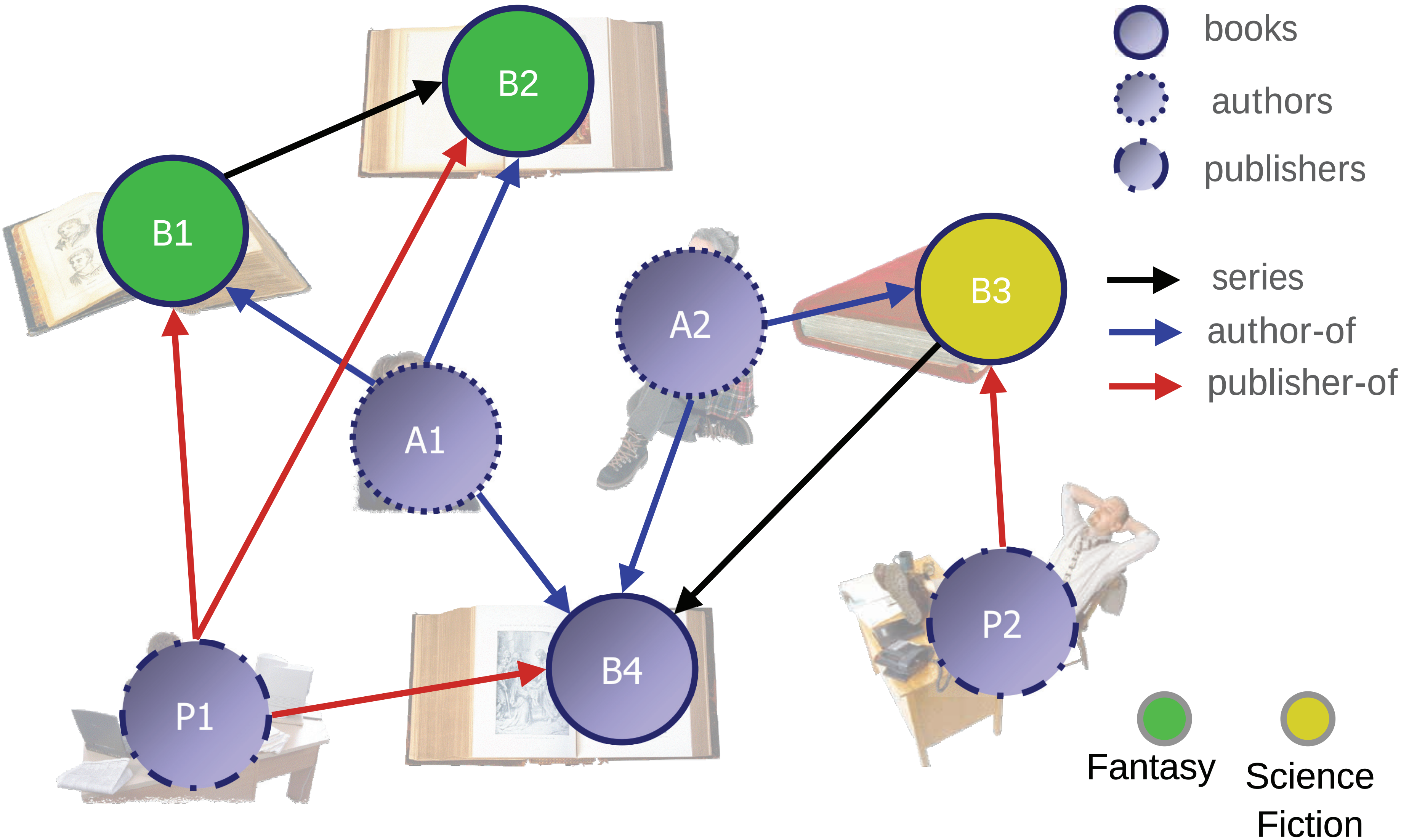
Outline

1. Motivation / Introduction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. Discriminative ILP
6. Conclusions

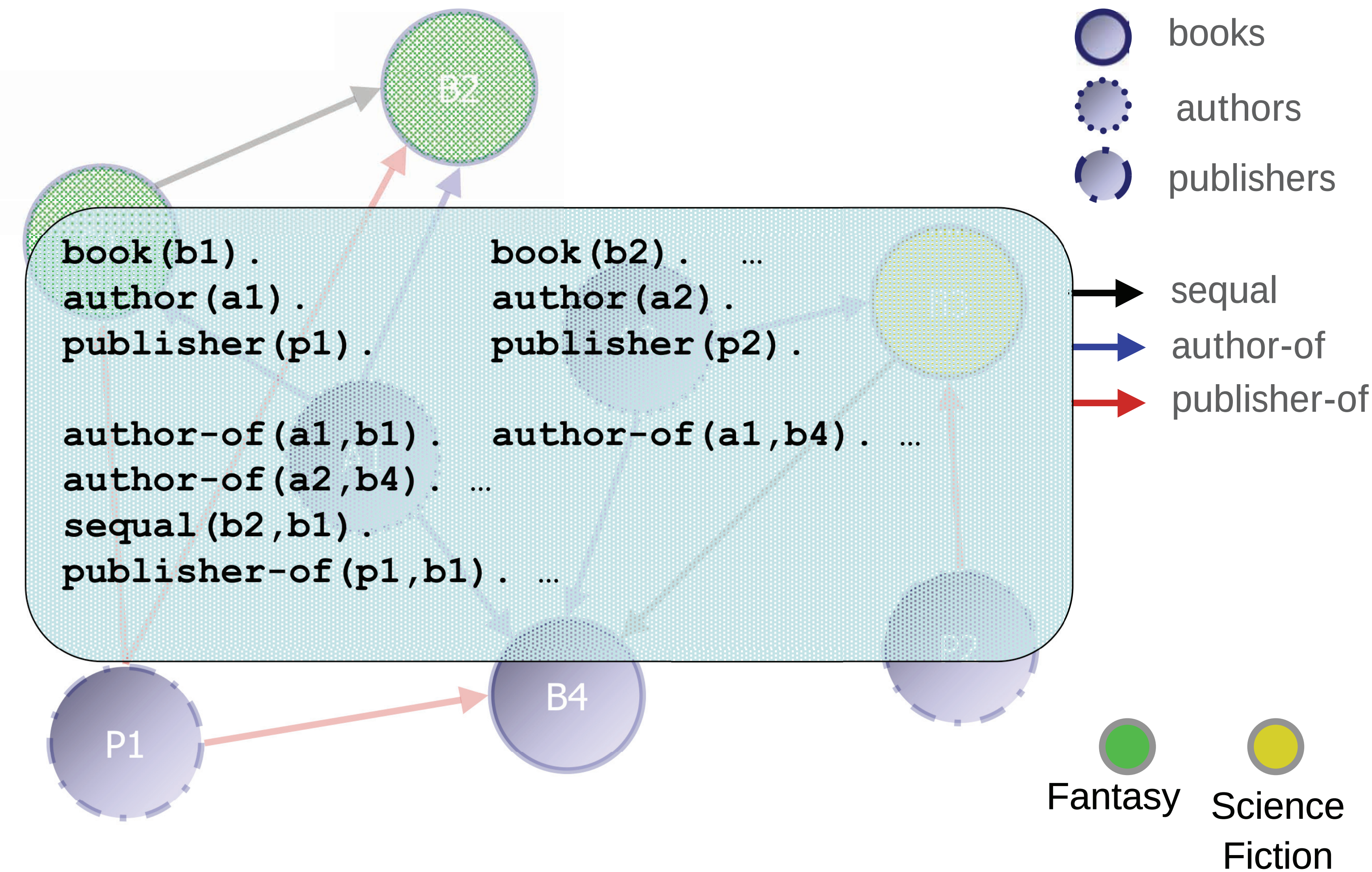
Web Mining / Linked Bibliographic Data / Recommendation Systems / …

[illustration inspired by Lise Getoor]

Web Mining / Linked Bibliographic Data / Recommendation Systems / ...

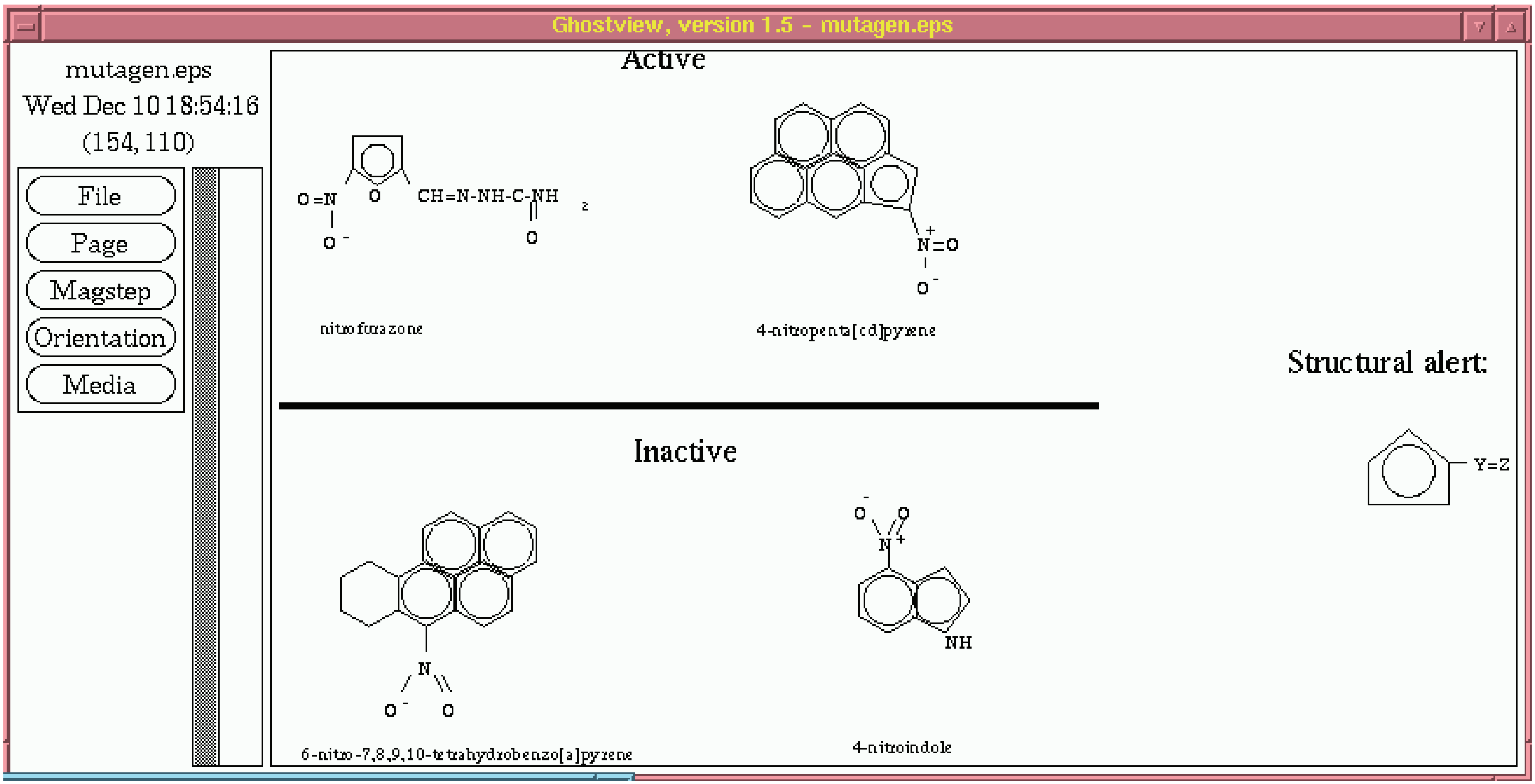


Web Mining / Linked Bibliographic Data / Recommendation Systems / …




```
book(b1).               book(b2).   …
author(a1).             author(a2).
publisher(p1).          publisher(p2).

author-of(a1,b1).       author-of(a1,b4).  …
author-of(a2,b4).  …
sequal(b2,b1).
publisher-of(p1,b1).  …
```

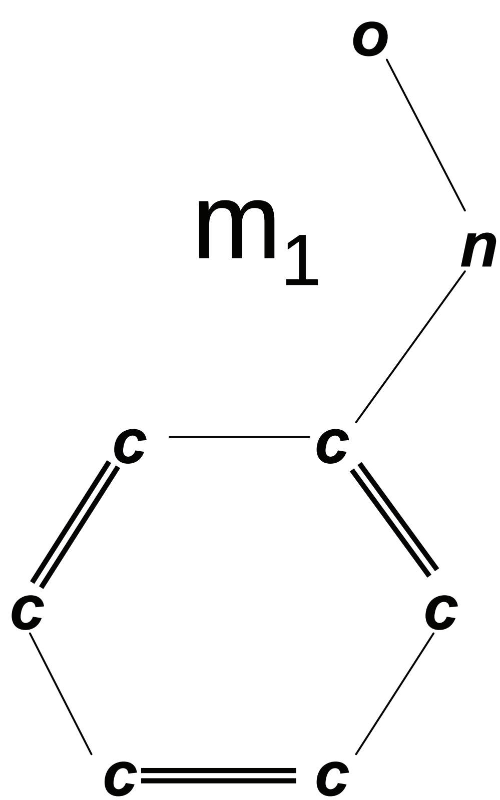
Structure Activity Relationship Prediction



Scientific Applications

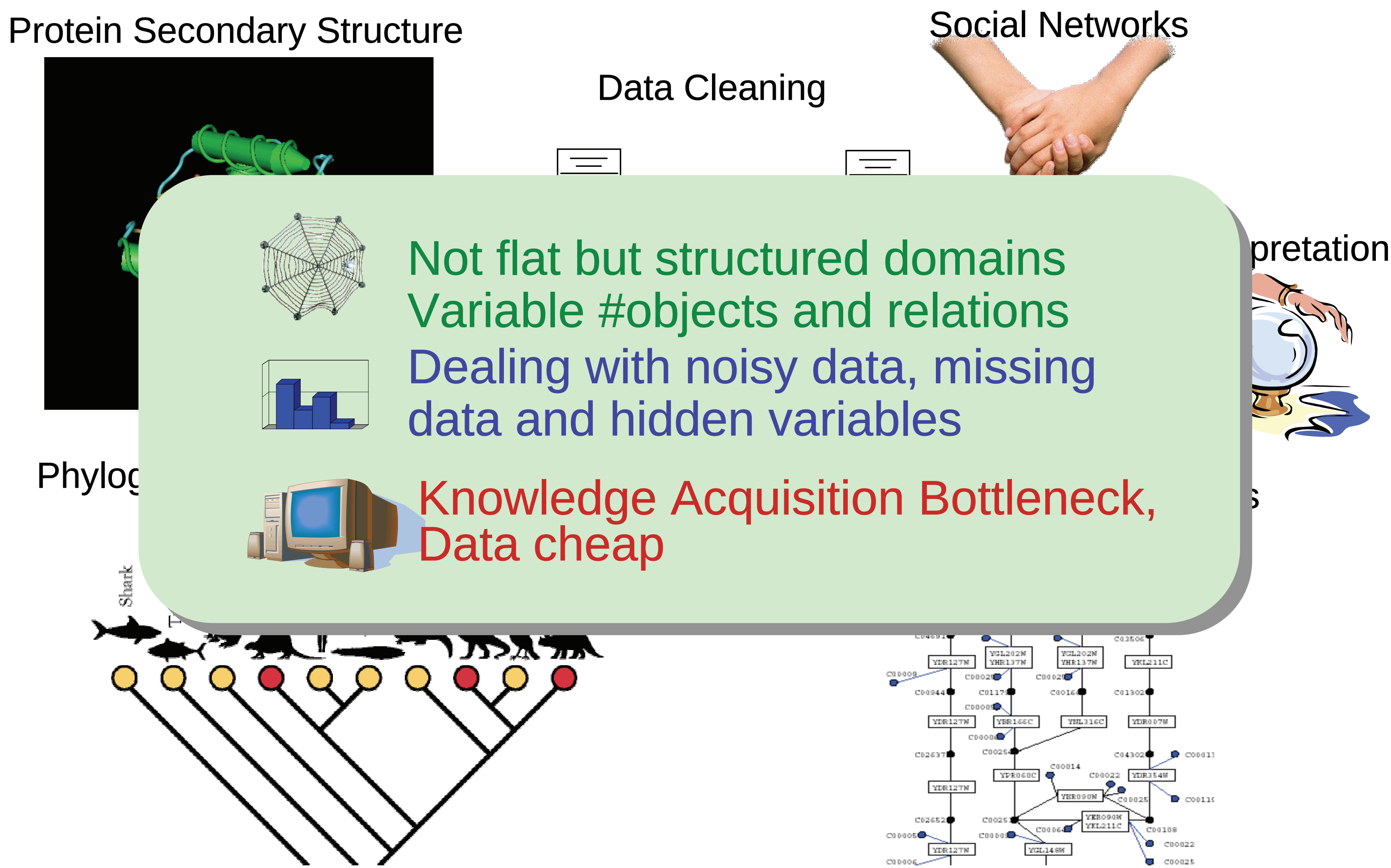
- Discovering
  - New knowledge (readily interpretable)
  - With general purpose relational learning or inductive logic programming sytems
  - Published in journals of the scientific application domain
  - Use of domain knowledge

Molecules




```
molecule(m1)
atom(m1,a11,c)
atom(m1,a12,n)
bond(m1,a11,a12)
charge(m1,a11,0.82)
...
```

... other Real World Applications



- Not flat but structured domains
- Variable #objects and relations
- Dealing with noisy data, missing data and hidden variables
- Knowledge Acquisition Bottleneck, Data cheap

Why do we need Probabilistic ILP* ?

*sometimes called statistical relational learning (SRL)

Real World Applications

Uncertainty

Probabilistic Logics

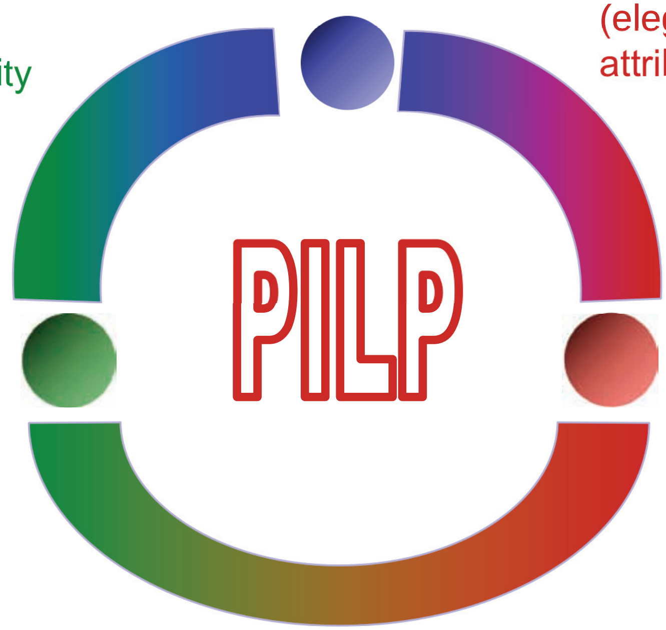
- no learning: too expensive to handcraft models

+ soft reasoning, expressivity

Statistical Learning (SL)

- attribute-value representations: some learning problems cannot (elegantly) be described using attribute value representations

+ soft reasoning, learning



Structured Domains

Machine Learning

Inductive Logic Programming (ILP)

Multi-Relational Data Mining (MRDM)

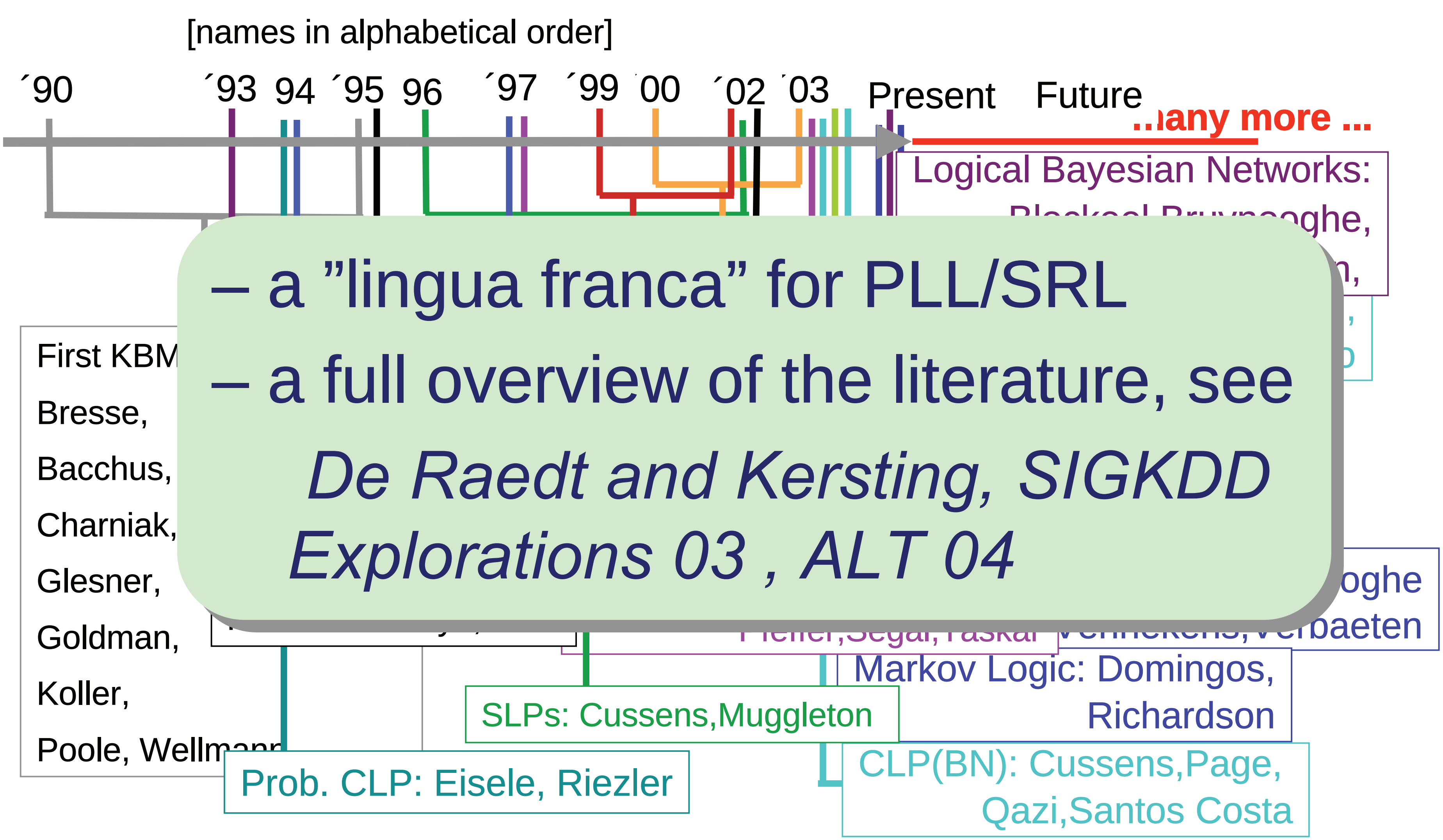
- crisp reasoning: some learning problems cannot (elegantly) be described without explicit handling of uncertainty

+ expressivity, learning

The Tutorial's Approach is …

- Start from ILP settings + extend them with probabilistic methods
  - Learning from entailment
  - Learning from interpretations
  - Learning from traces or proofs
- Hence, probabilistic ILP
- Provide insight in some logical issues
- Focus on learning … but also relevant to KR
  - Probabilities on facts, interpretations, proofs

... but NOT...



– a "lingua franca" for PLL/SRL

– a full overview of the literature, see *De Raedt and Kersting, SIGKDD Explorations 03 , ALT 04*

Outline

1. Motivation / Introduction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. Discriminative ILP
6. Conclusions

Propositional Logic

Program

```
blackbird.
normal :- blackbird.
flies :- bird, normal.
bird :- ostrich.
bird :- blackbird.
```

Clause · head · atom · body

Clauses: **IF** **`bird`** and **`normal`** are true **THEN** **`flies`** is true

Herbrand Base (HB) = all atoms in the program

`blackbird, normal, flies, ostrich, bird`

Model Theoretic Semantics
- Restrictions on Possible Worlds -

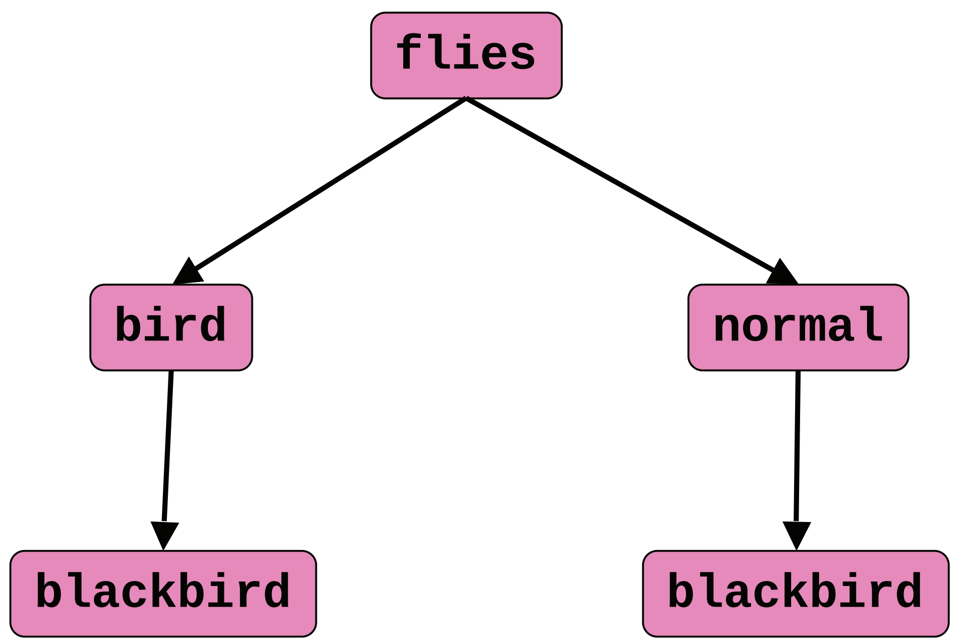
- Herbrand Interpretation
  - Truth assignments to all elements of HB

- An interpretation is a model of a clause C
  ⇔ If the body of C holds then the head holds, too.

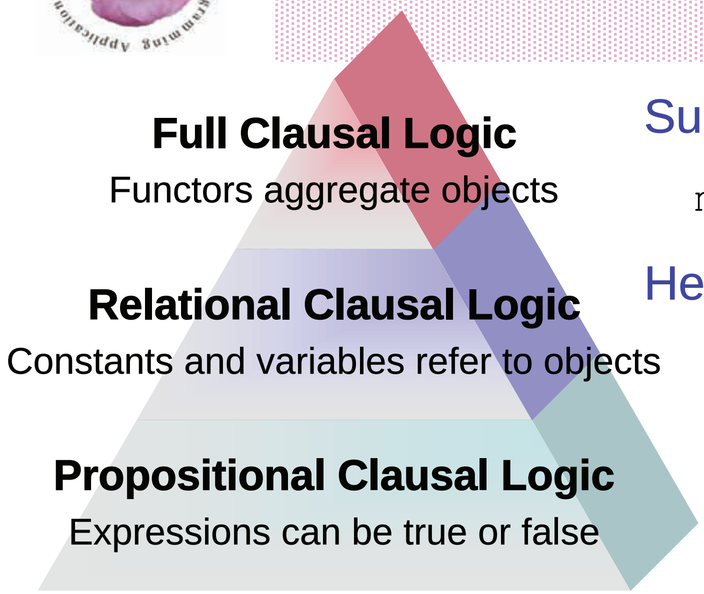
```
blackbird.
normal :- blackbird.
flies :- bird, normal.
bird :- ostrich.
bird :- blackbird.

{blackbird,normal,bird,flies}
```

Proof Theoretic Semantics

```
blackbird.
normal :- blackbird.
flies :- bird, normal.
bird :- ostrich.
bird :- blackbird.
```



Upgrading - continued



Substitution: Maps variables to terms: {M / ann}:

`mc(P,a) :- mother(ann,P),pc(ann,a),mc(ann,a).`

Herbrand base: set of ground atoms (no variables):

`{mc(fred,fred),mc(rex,fred),…}`

constant

head

body

`nat(0).`

clause `nat(succ(X)) :- nat(X).`

functor variable

term

atom

Interpretations can be infinite !

`nat(0),nat(succ(0)),`

`nat(succ(succ(0))), ...`

Motivation

- We shall start from logic + learning
  – Inductive logic programming and relational learning
- Methodological practice in ILP
  – Many ILP systems obtained by upgrading propositional or attribute-value learning systems
  – Is the methodology also applicable to Probabilistic ILP ?
- Inductive logic programming has studied several settings for learning
  – Do they also apply to Probabilistic ILP ?

Traditional ILP Problem

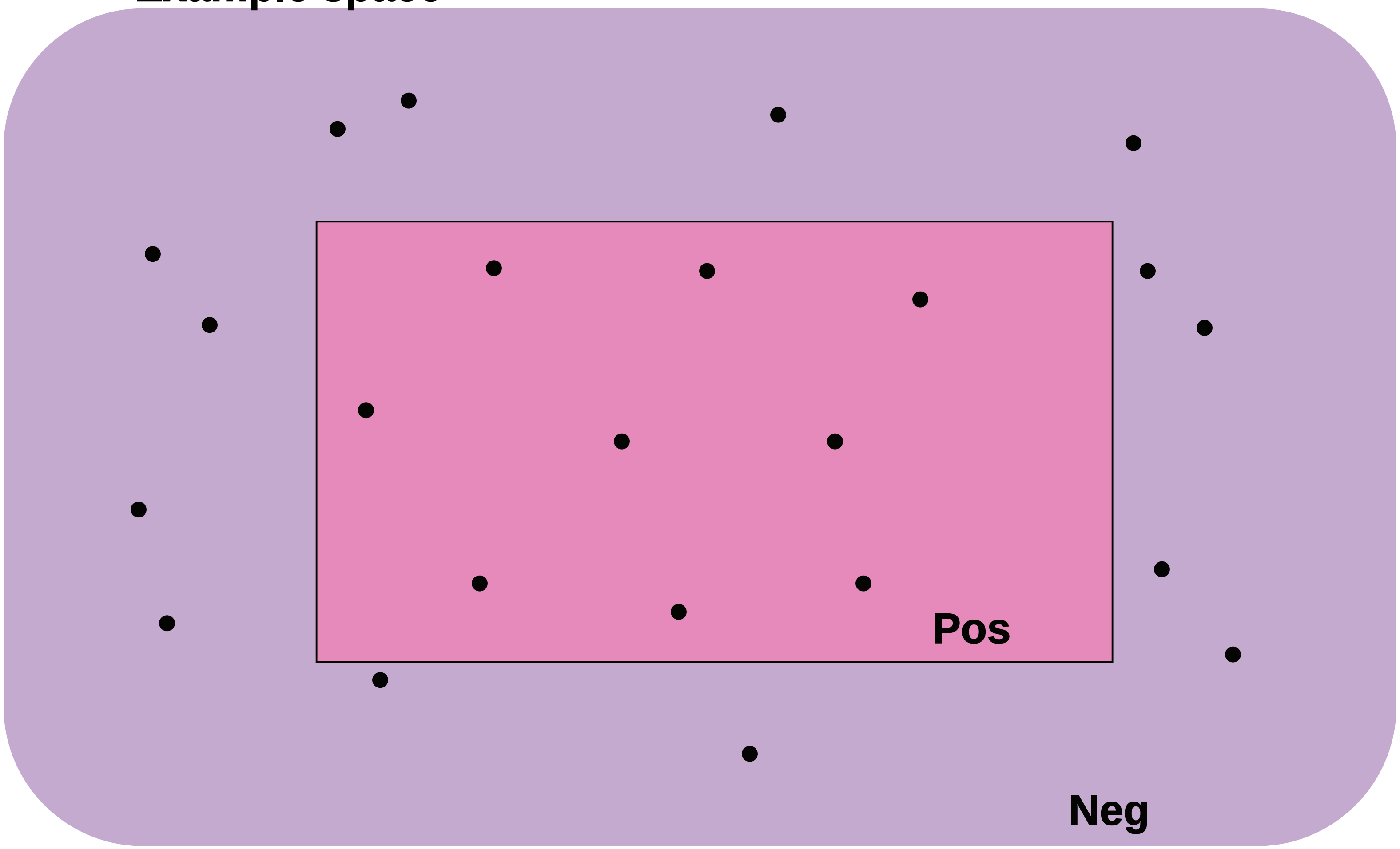
- **Given**
  - a set of positive and negative examples **(Pos, Neg)**
  - a background theory **B**
  - a
  -
  -

Concept-learning in a logical/relational representation

- **Find**
  - A hypothesis **h** over **Lh** that covers all positive **Pos** and no negative **Neg** examples taking **B** into account

Traditional ILP Problem

Example space



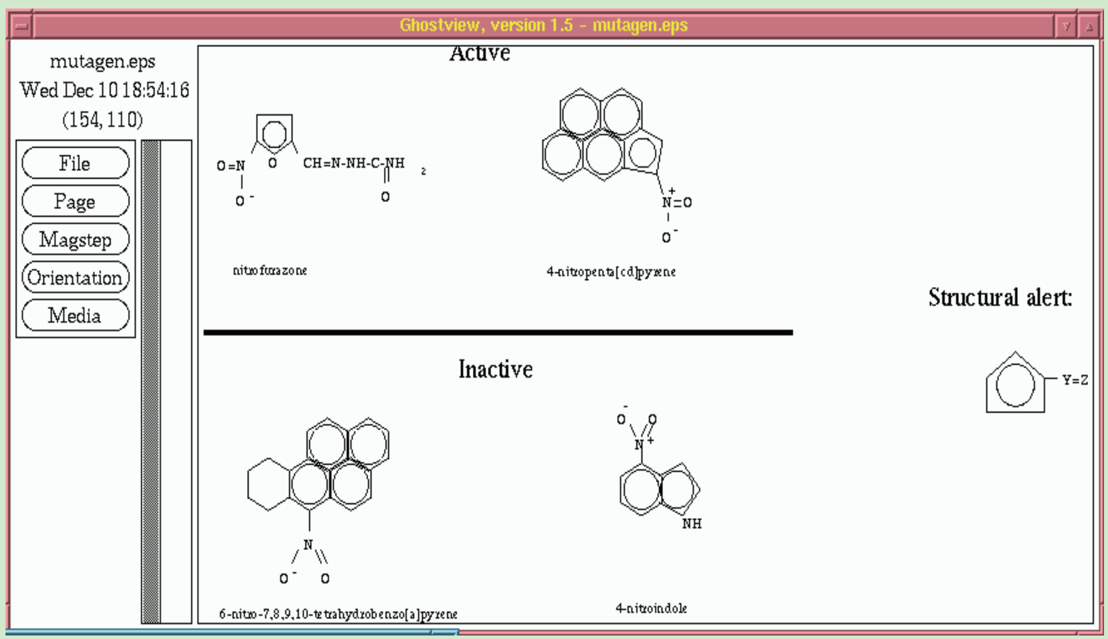
Three possible choices

- **Entailment**
  – Covers(H,e) iff H |= e
- **Interpretations**
  – Covers(H,e) iff e is a model for H
- **Proofs**
  – Covers(H,e) iff e is a proof for H

Learning from entailment

- Examples are facts (or clauses)
- An example e is covered by a hypothesis h if and only if $B \cup h \models e$
- W
  - F vasan), G

Applications



The Mutagenicity dataset

Background theory

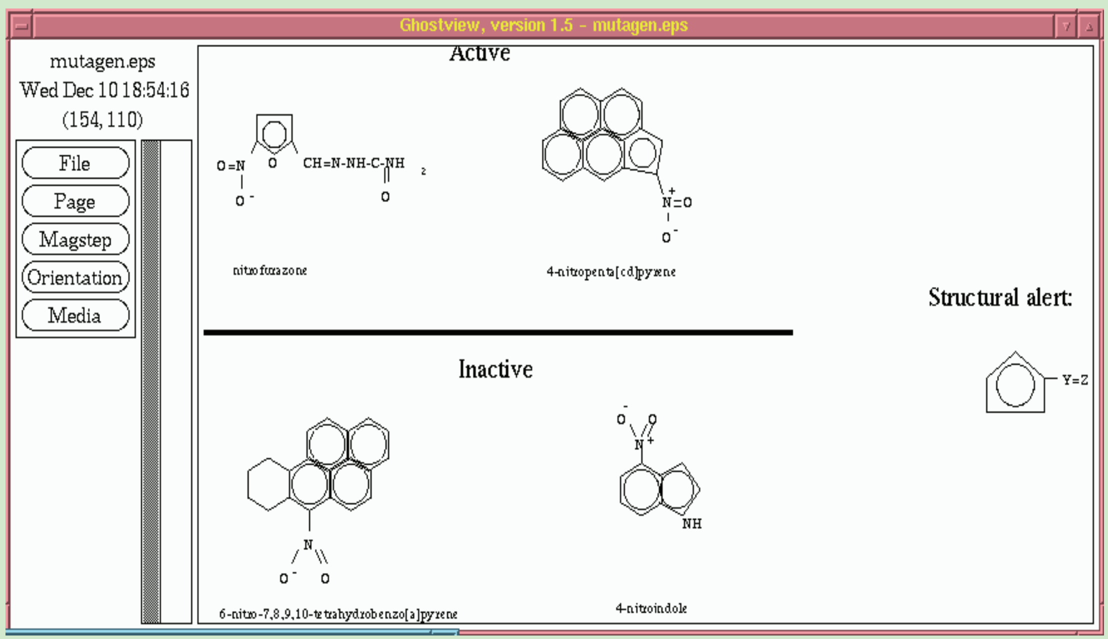
```
molecule(225).                                   bond(225,f1_1,f1_2,7).
logmutag(225,0.64).                              bond(225,f1_2,f1_3,7).
lumo(225,-1.785).                                bond(225,f1_3,f1_4,7).
logp(225,1.01).                                  bond(225,f1_4,f1_5,7).
nitro(225,[f1_4,f1_8,f1_10,f1_9]).               bond(225,f1_5,f1_1,7).
atom(2                                           f1_9,2).
atom                                             _10,2).
atom                                             _11,1).
atom                                             _12,2).
atom                                             _13,1).
...
ring
hete
```

hypo

```
mut
```

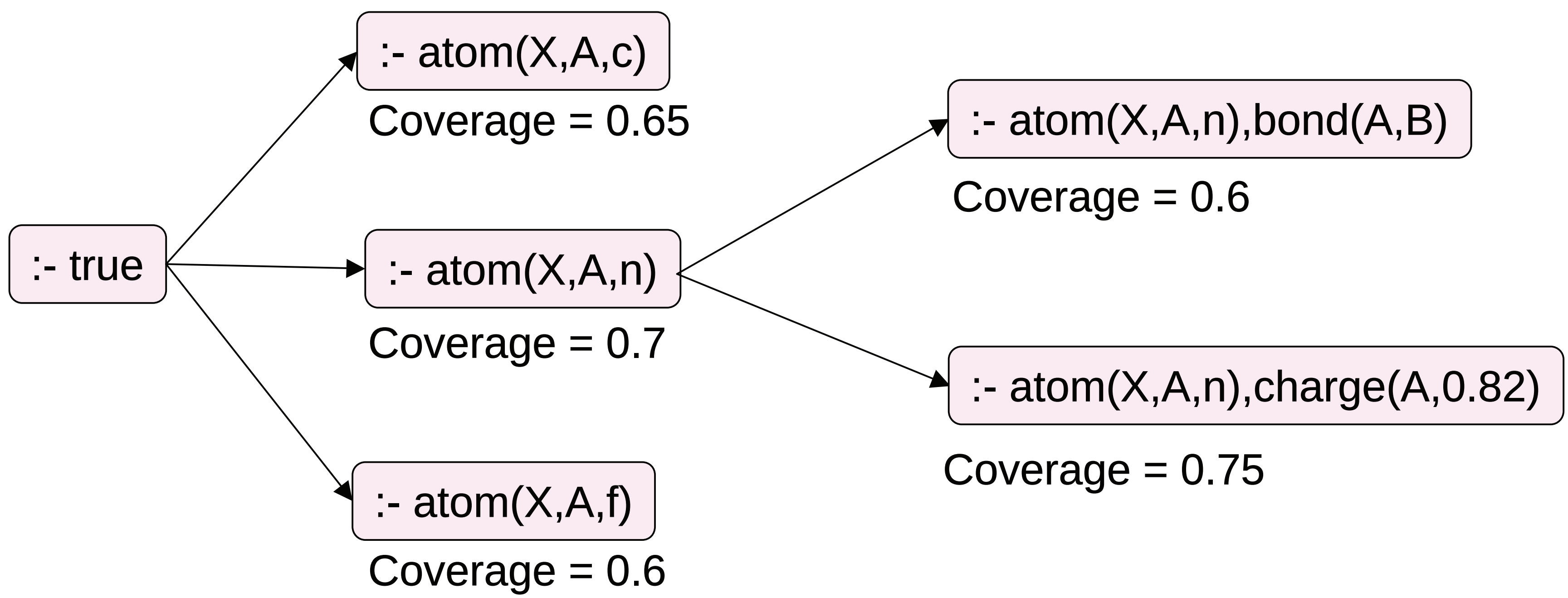
Examp

Applications



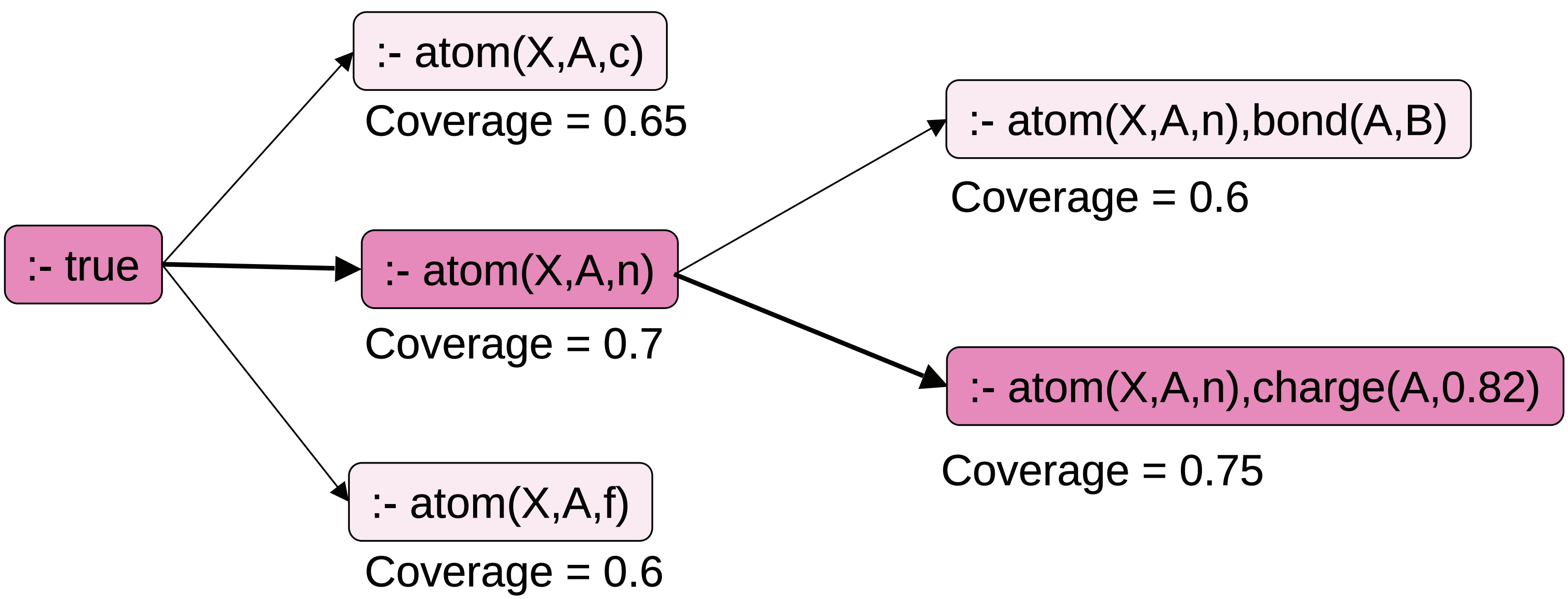
Example ILP Algorithm: FOIL (Quinlan 1990)

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



Example ILP Algorithm: FOIL (Quinlan 1990)

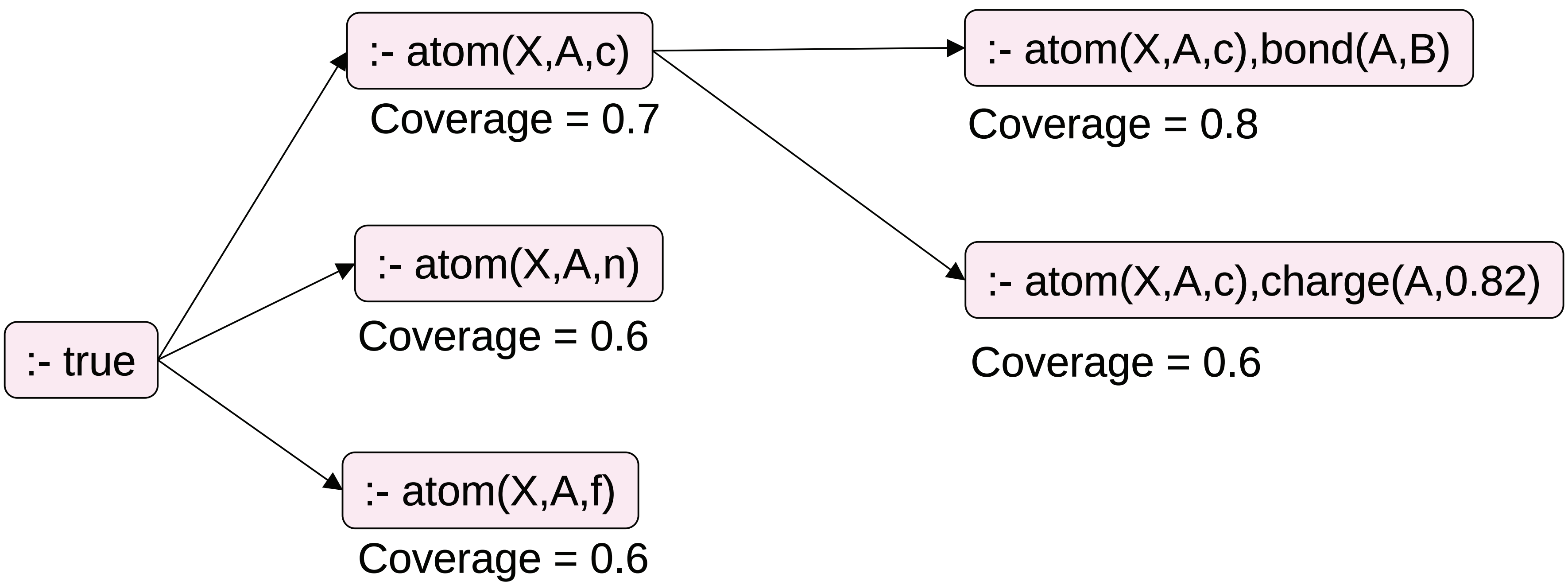
- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



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

Example ILP Algorithm: FOIL (Quinlan 1990)

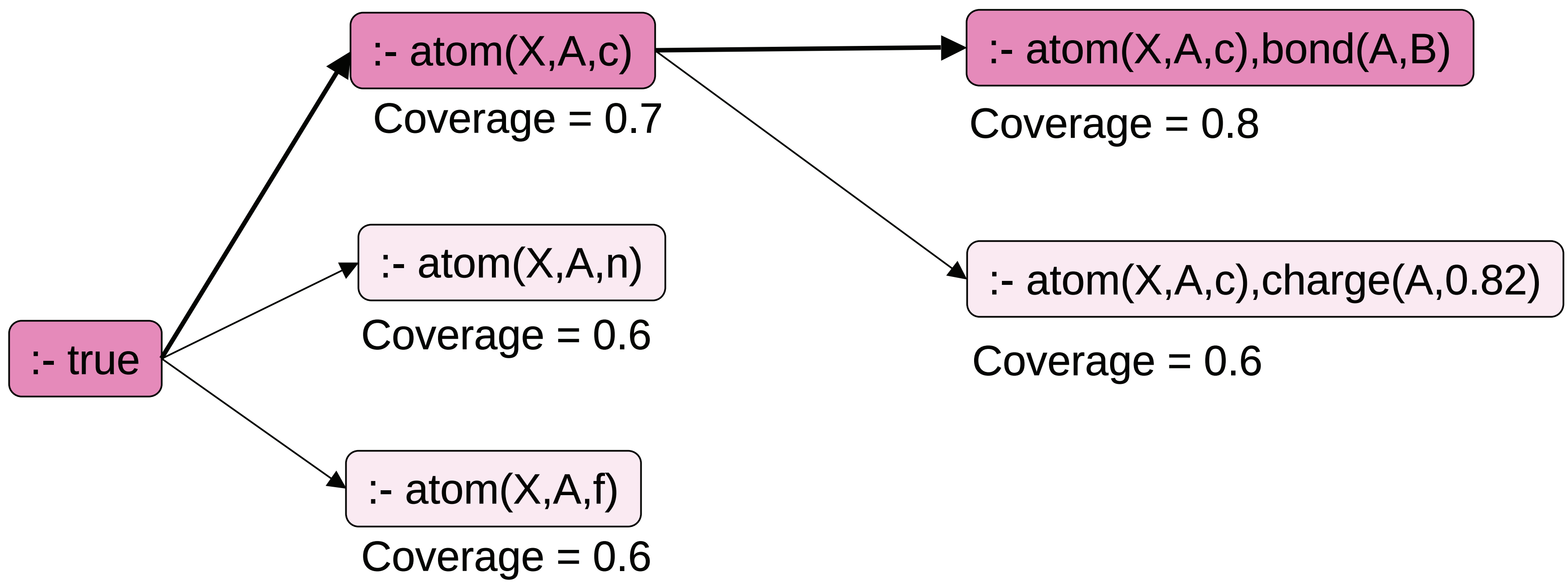
- Greedy separate-and-conquer search for clause set
- Greedy general-to-specific search for single clause



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

Example ILP Algorithm: FOIL (Quinlan 1990)

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



mutagenic(X) :- atom(X,A,c),bond(A,B)

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

Example ILP Algorithm: FOIL (Quinlan 1990)

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

⋮

mutagenic(X) :- atom(X,A,c),charge(A,0.45)

mutagenic(X) :- atom(X,A,c),bond(A,B)

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

ILP Principles

- Searching the space of hypotheses
  - Using ordering such as theta-subsumption
  - Using refinement operators
- Limitations of traditional machine learning
  - Dealing with structured data instead of feature vector, attribute value, boolean, propositional etc. representations
  - Employing background knowledge
  - Interpretability of results
- Application areas
  - Chemo- and bio-informatics, e.g. predictive toxicology
  - Language learning and information retrieval
  - Ecological applications
  - ….

Learning from Interpretations

- Examples are (Herbrand) interpretations, i.e., sets of ground facts
- An example e is covered by a hypothesis h if and only if the example is a model for the hypothesis h
- Well known examples
  - Logan-H (Khardon), Claudien (De Raedt and Dehaspe), Warmr (Dehaspe), …

An example

- Examples
  - Postive: { human(luc), human(lieve), male(luc), female(lieve)}
  - Negative: { bat(dracula), male(dracula), vampire(dracula)}
  - …
- Discriminative Hypothesis
  - Learning from positives and negatives (possibles / impossibles)
  - human(X) :- male(X)
- Interpretation I is a model for clause h :- b1, …, bn iff for all θ such that $\{b1\theta,\ldots,bn\theta\} \subseteq I$, we have that $h\theta$ in I
- Consider {X=dracula} for negative

An example

- Examples
  - Postive: { human(luc), human(lieve), male(luc), female(lieve)}
- Characteristic Hypothesis
  - Learning from positives (possibles) only
  - human(X) :- female(X)
  - human(X) :- male(X)
-

Applications

Finding integrity constraints /

frequent patterns in relational databases

Learning from Traces/Proofs

- Examples are proof trees
- An example e is covered by a hypothesis h if and only if e is a legal proof tree in h
- W
  –

Applications

– Tree bank grammar learning

– Program synthesis

- Shapiro's MIS poses queries in order to reconstruct trace or proof

An example

```
sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
article(singular, A, B) :- terminal(A, a, B).
article(singular, A, B) :- terminal(A, the, B).
article(plural, A, B)   :- terminal(A, the, B).
noun(singular, A, B) :- terminal(A, turtle, B).
noun(plural, A, B)   :- terminal(A, turtles, B).
```

Applications

– Tree bank grammar learning

– Program synthesis

- Shapiro's MIS poses queries in order to reconstruct trace or proof

Use of different Settings

Information

\-

\+

Learning from entailment

– The mo…

Different settings provide different levels of information about target program

(cf. De Raedt, AIJ 97)

Learning from trac…

– Typically used for hard problems, when other settings seem to fail or fail to scale up

– E.g., program synthesis from examples, grammar induction, multiple predicate learning

Outline

1. Motivation / Introduction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. Discriminative ILP
6. Conclusions

Probabilistic ILP: What Changes?

- Clauses annotated with probability labels
  - E.g. in Sato's Prism, Eisele and Muggleton's SLPs, Kersting and De Raedt's BLPs, …
- Prob. covers relation covers(e,H U B) = ***P***(e | H,B)
  - Probability of example given background and hypothesis
  - Probability distribution **P** over the different values e can take; so far only (true,false)
  - impossible examples have probability 0
- Knowledge representation issue
  - Define probability distribution on examples / individuals
  - What are these examples / individuals?

Probabilistic ILP Problem

Given

- a set of examples **E**
- a background theory **B**
- a language **Le** to represent examples
- a language **Lh** to represent hypotheses
- a probabilistic covers **P** relation on **Le x Lh**

Find

- hypothesis h* maximizing some score based on the probabilistic covers relation

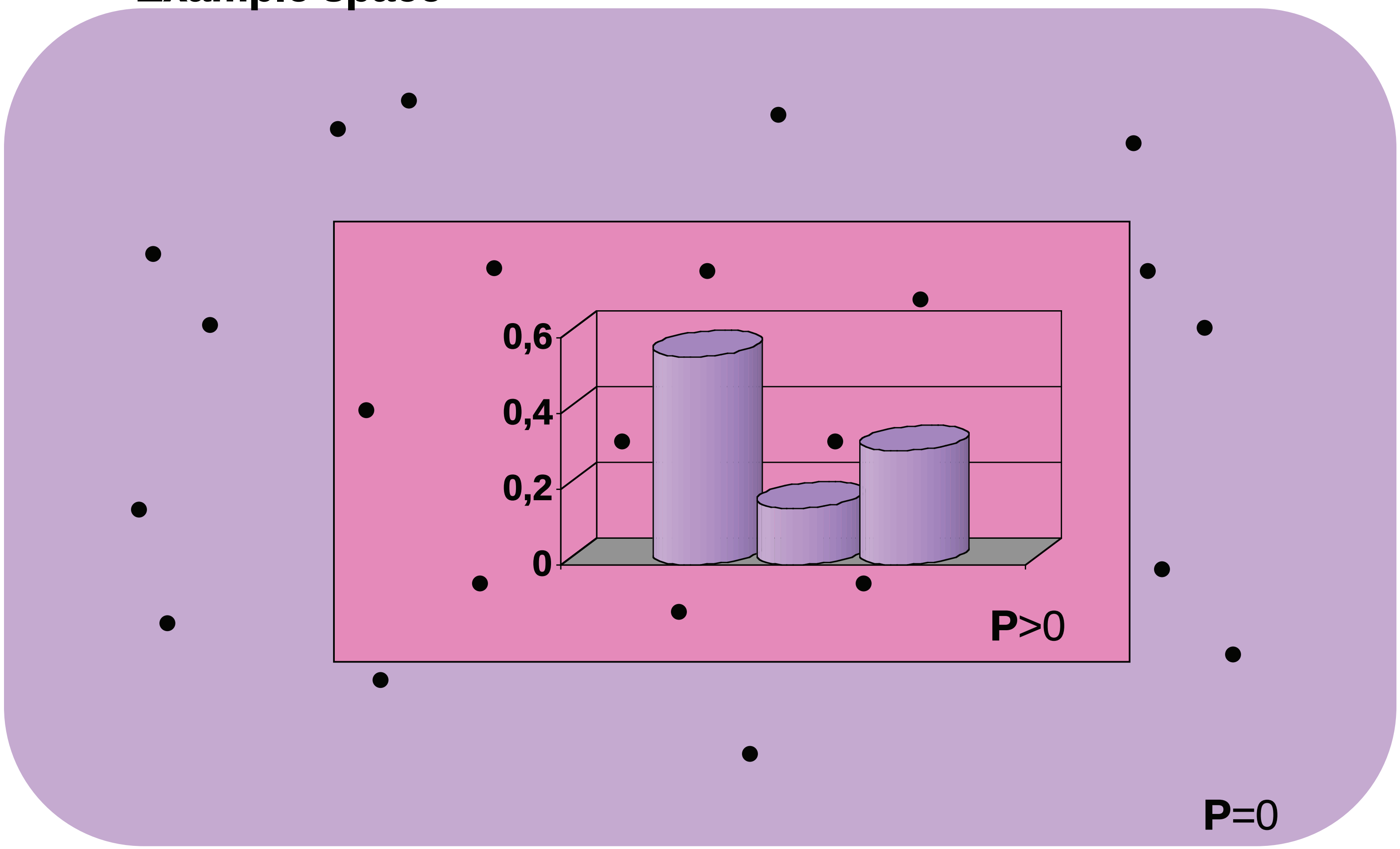
Probabilistic ILP: Three Issues

- Defining Lh and **P**
  - Clauses + Probability Labels

- Learning Methods
  - Parameter Estimation
    - Learning probability labels for fixed clauses
  - Structure learning
    - Learning both components

Probabilistic ILP Problem

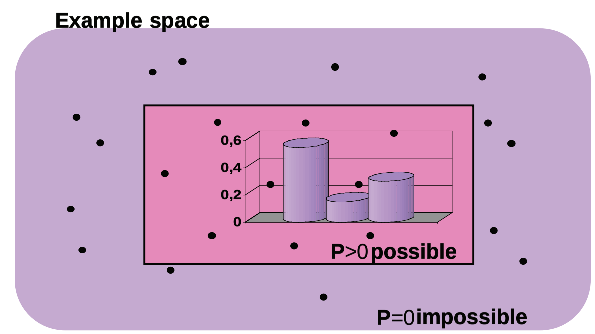
Example space



Probabilistic ILP: Two Objectives

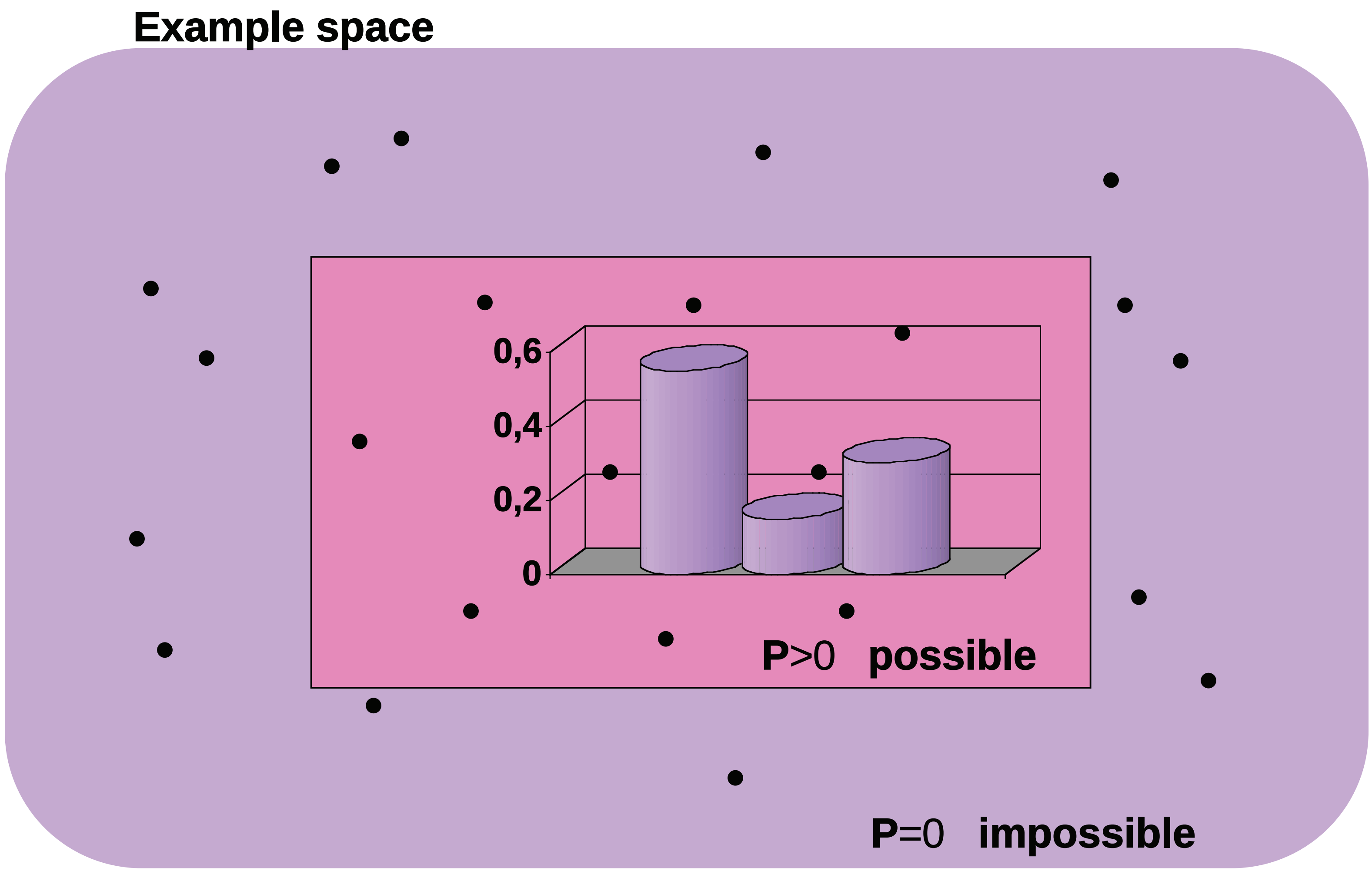
- Generative Learning
  - Estimate joint probability distribution
  - E.g., likelihood + iid

$$h^* = \arg\max_h \mathbf{P}(\mathbf{e}|h,B)$$
$$= \arg\max_h \prod_i \mathbf{P}(e_i|h,B)$$

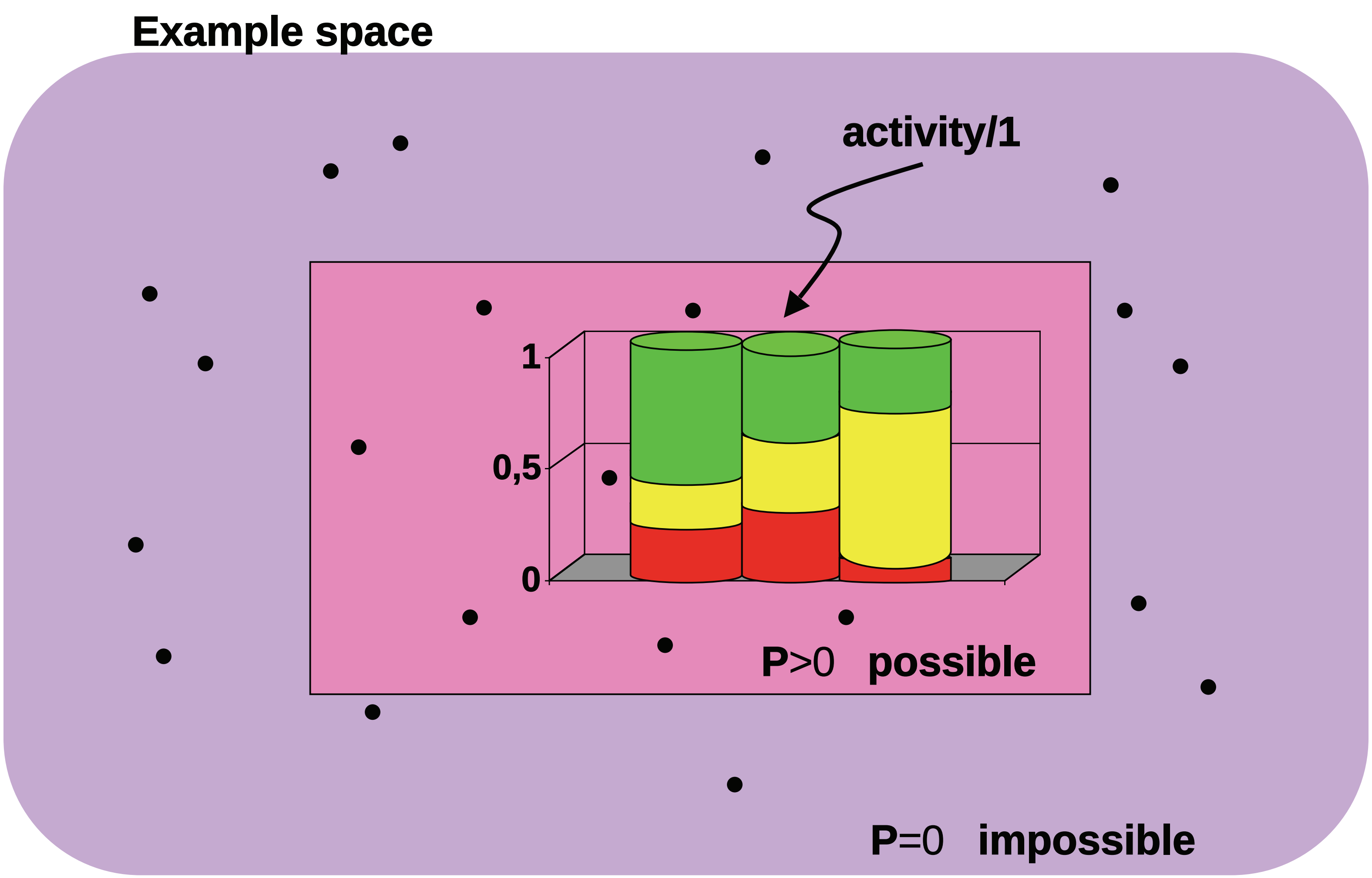


- Discriminative Learning
  - Estimate conditional prob. distribution over some predicates given evidence for the others

Probabilistic ILP Problem



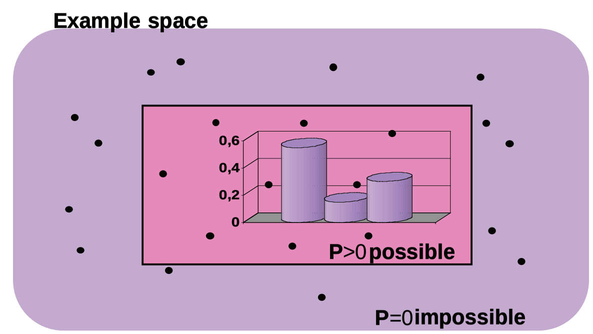
Probabilistic ILP Problem



Probabilistic ILP: Two Objectives

- Generative Learning
  - Estimate joint probability distribution
  - E.g., likelihood

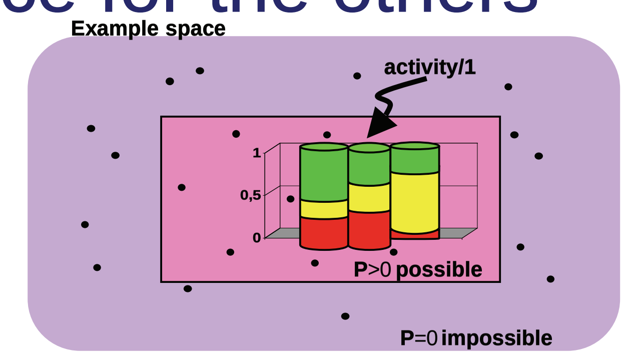
    h* = arg max_h **P**(**e**|h,B)



- Discriminative Learning
  - Estimate conditional prob. distribution over some predicates given evidence for the others
    E.g., conditional likelihood

    c* = arg max_c **P**(**e**|c,h\c,B)

    = arg max$_c$ $\prod_i$ **P**(e_i|c,h\c,B)



Probabilistic ILP: Three Settings

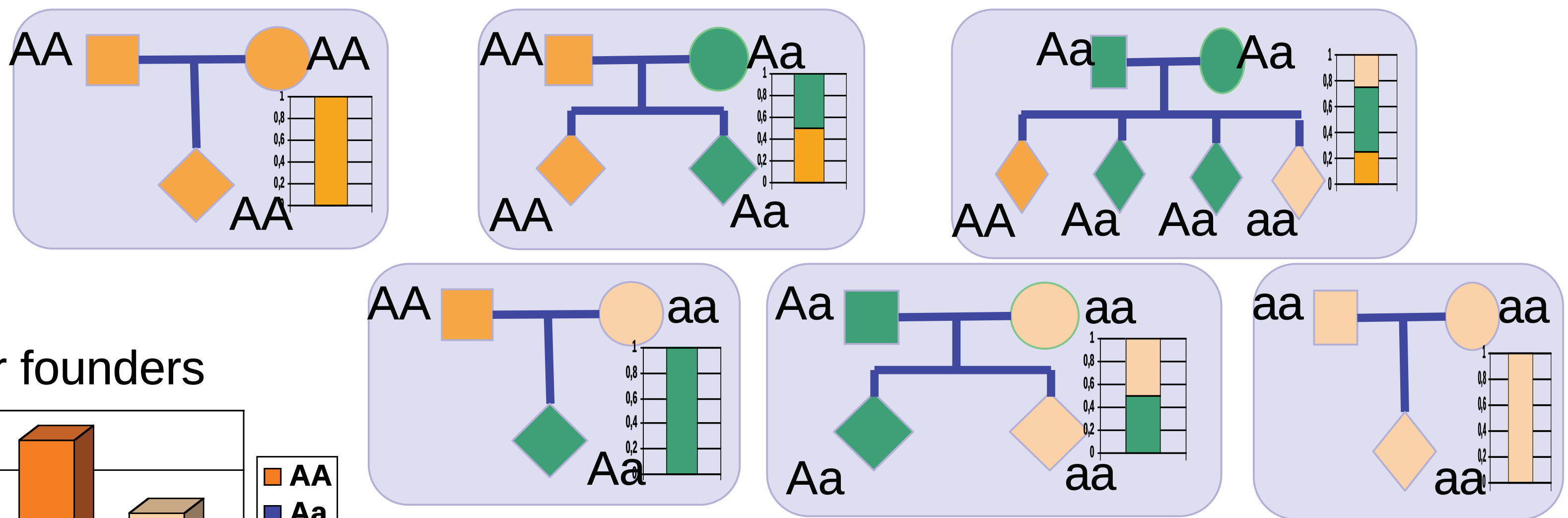
- Probabilistic learning from entailment
  - Eichele and Muggleton's Stochastic Logic Programs, Sato's Prism, Poole's ICL
- Probabilistic learning from proofs
  - Learning the structure of SLPs; a tree-bank grammar based approach, Logical HMMs, Anderson's RMMs
- Probabilistic learning from interpretations
  - Bayesian logic programs, Koller's PRMs, Domingos' MLNs, Vennekens' LPADs, Taskar's RMNs

Outline

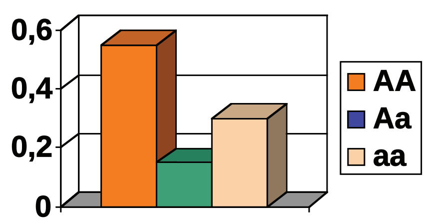
1. Motivation / Introduction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. Discriminative ILP
6. Conclusions

Blood Type / Genetics/ Breeding

- 2 Alleles: A and a
- Probability of Genotypes AA, Aa, aa ?



Prior for founders



Bayesian Networks [Pearl]

[illustration inspired by Kevin Murphy]

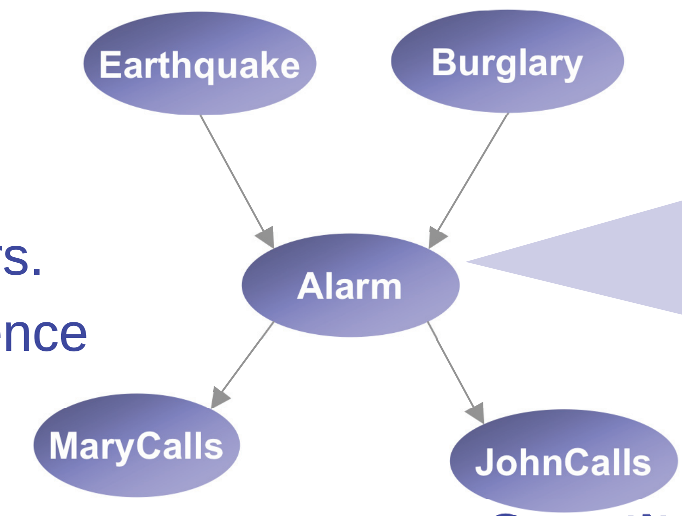
Compact representation of joint probability distributions

P(E,B,A,M,J)

Qualitative part:

Directed acyclic graph

- Nodes - random vars.
- Edges - direct influence



E	B	P(A | B,E)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

Quantitative part:

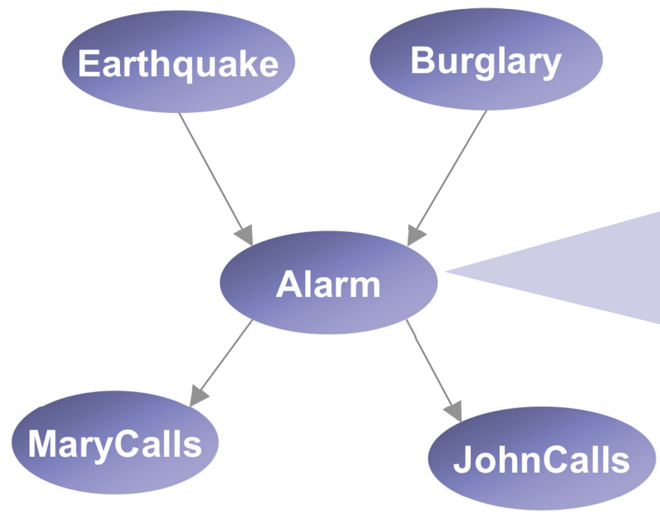
Set of conditional probability distributions

Together:

Define a unique distribution in a compact, factored form

P(E,B,A,M,J)=P(E) * P(B) * P(A|E,B) * P(M|A) * P(J|A)

Bayesian Networks [Pearl]



E	B	$P(A \mid B,E)$	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

$$P(j) = P(j|a) * P(m|a) * P(a|e,b) * P(e) * P(b)$$

$$+ P(j|a) * P(m|a) * P(a|e,\bar{b}) * P(e) * P(\bar{b})$$

...

$$+ P(j|\bar{a}) * P(\bar{m}|\bar{a}) * P(\bar{a}|\bar{e},\bar{b}) * P(\bar{e}) * P(\bar{b})$$

Bayesian nets: Parameter Estimation

complete data set
simply counting

A1	A2	A3	A4	A5	A6	
true	true	false	true	false	false	X1
false	true	true	true	false	false	X2
...	...	...	...	...	...	⋮
true	false	false	false	true	true	XM

Bayesian nets: Parameter Estimation

incomplete data set

Real-world data: states of some random variables are missing

- E.g. medical diagnose: not all patient are subjects to all test
- Parameter reduction, e.g. clustering, ...

hidden/ latent

A1	A2	A3	A4	A5	A6
true	true	?	true	false	false
?	true	?	?	false	false
...	...	...	...	...	...
true	false	?	false	true	?

missing value

Expectation-Maximization (EM): Idea

- In the case of complete data, ML parameter estimation is easy:
  - simply counting (1 iteration)

Incomplete data ?

1. Complete data (Imputation)
   - most probable?, average?, ... value
2. Count
3. Iterate

EM Idea: Complete the data

$A \rightarrow B$

incomplete data

A	B
true	true
true	?
false	true
true	false
false	?

$P(B = true \mid A = true) = 0.6$

$P(B = true \mid A = false) = 0.2$

complete

complete data

expected counts

A	B	N
true	true	1.6
true	false	1.4
false	true	1.2
false	false	0.8

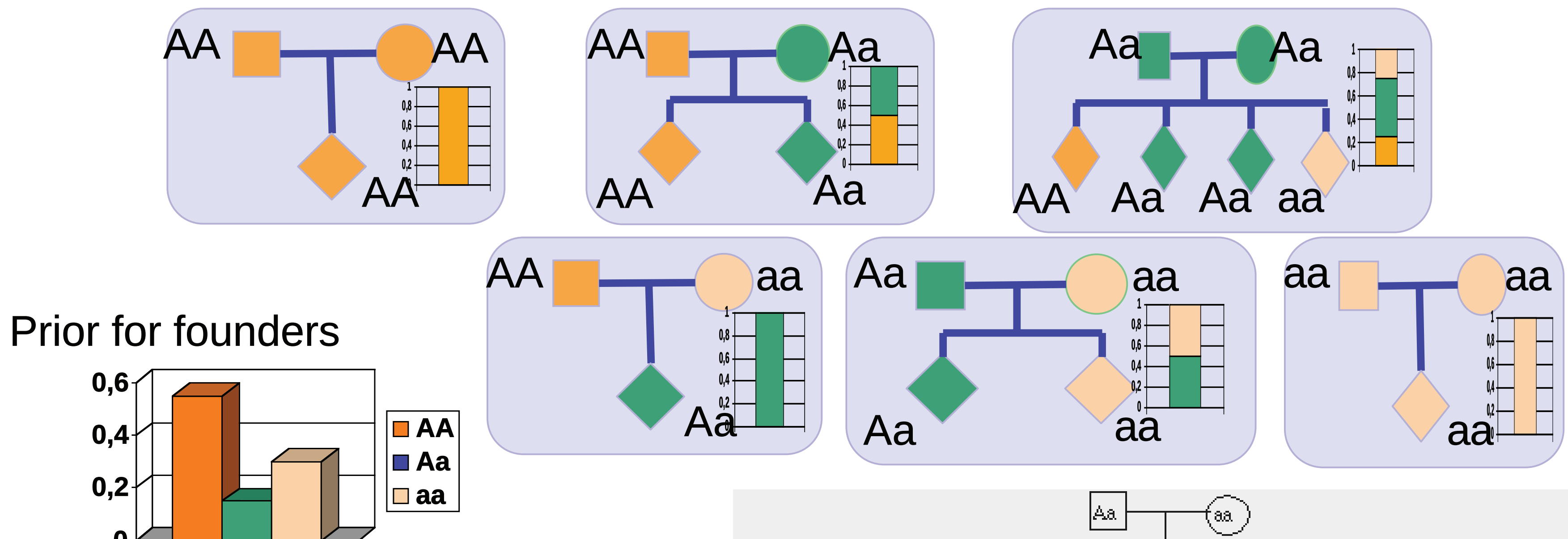
count + iterate

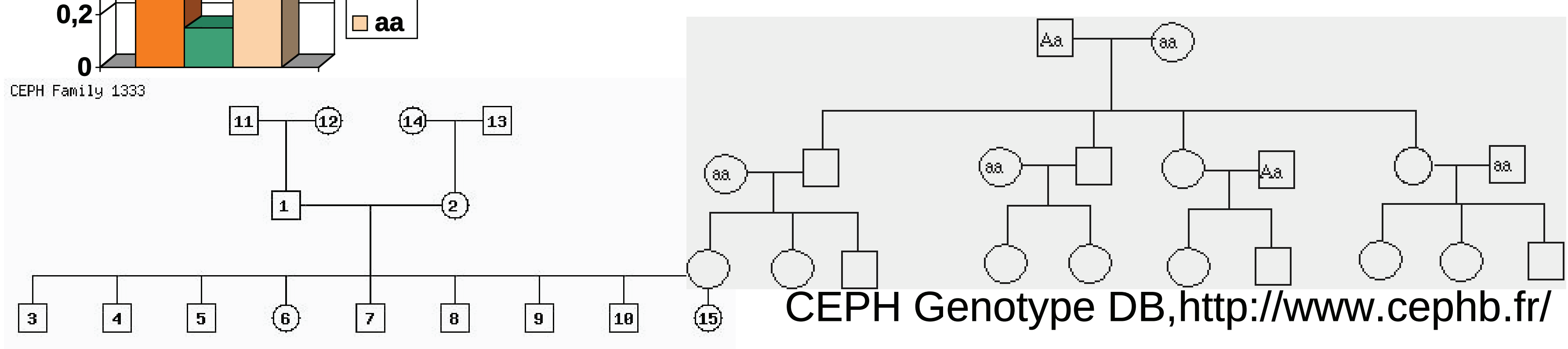
$$\theta_{B=true \mid A=true} = \frac{1.6}{1.6+1.4} = 0.54$$

$$\theta_{B=true \mid A=false} = \frac{1.2}{1.2+0.8} = 0.6$$

Blood Type / Genetics/ Breeding

- 2 Alleles: A and a
- Probability of Genotypes AA, Aa, aa ?



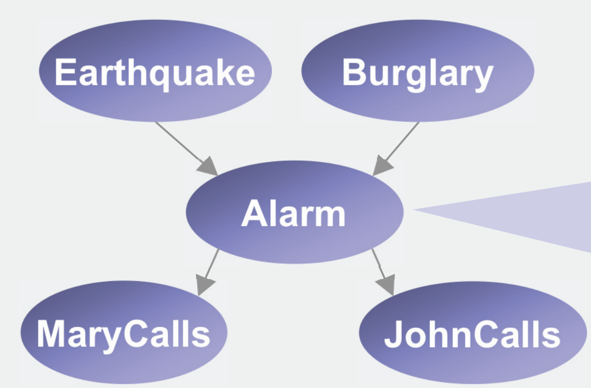


CEPH Genotype DB,http://www.cephb.fr/

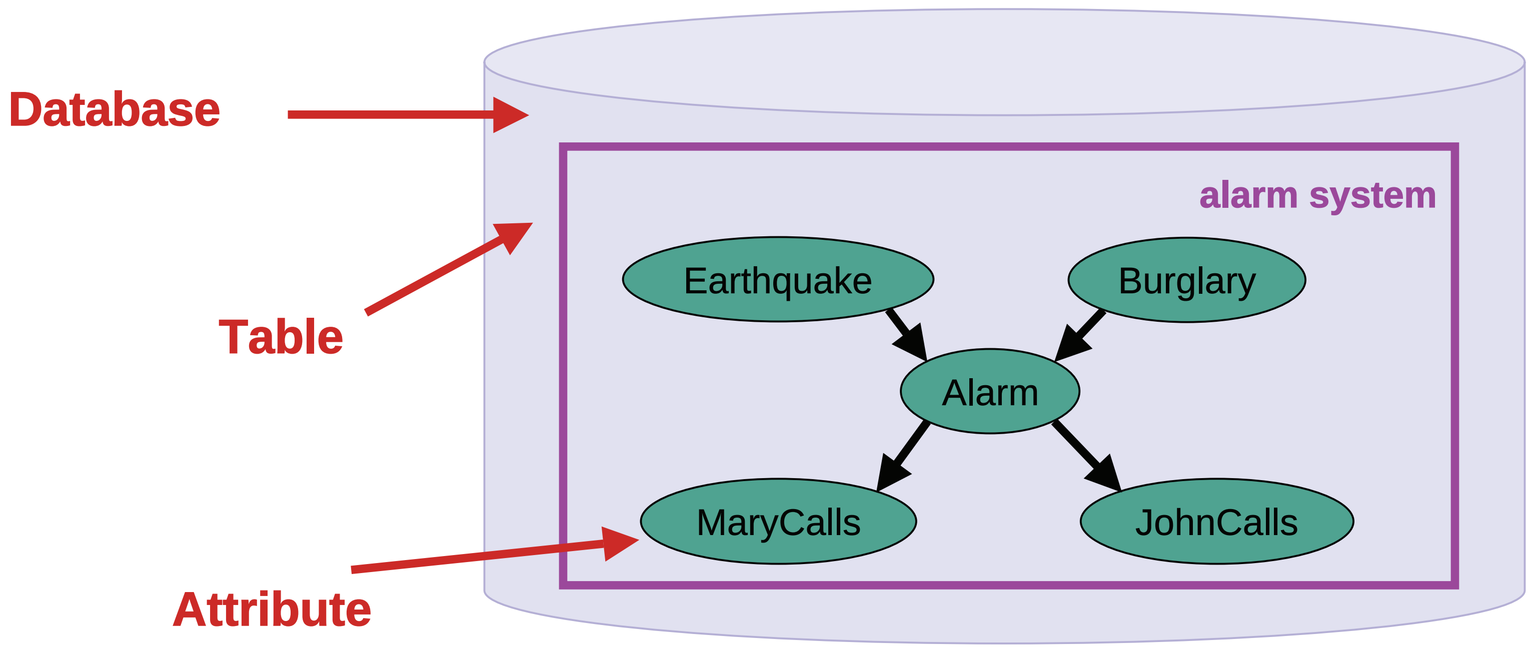
Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]

- Database theory
- Entity-Relationship Models
  - Attributes = RVs

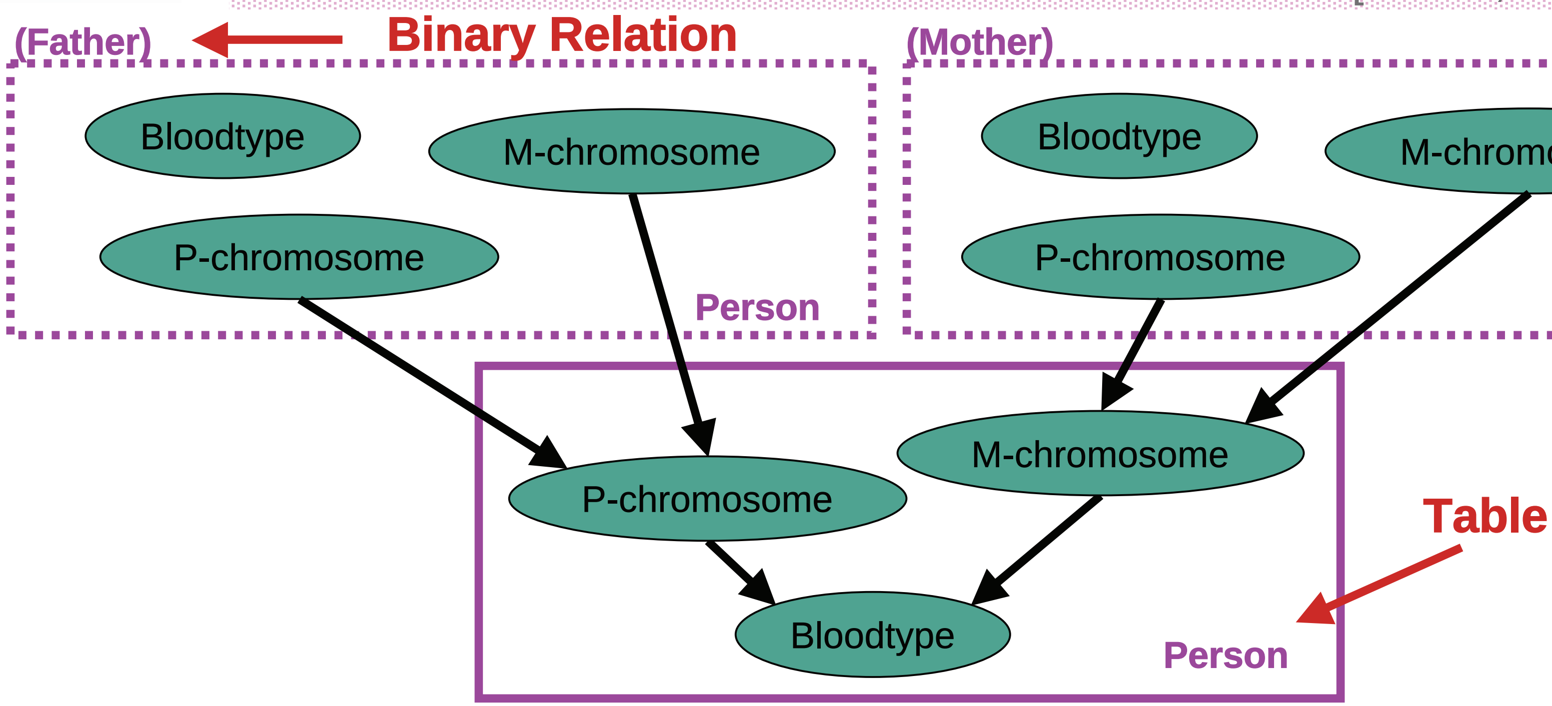


E	B	$P(A \mid B,E)$	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99



Probabilistic Relational Models (PRMs)

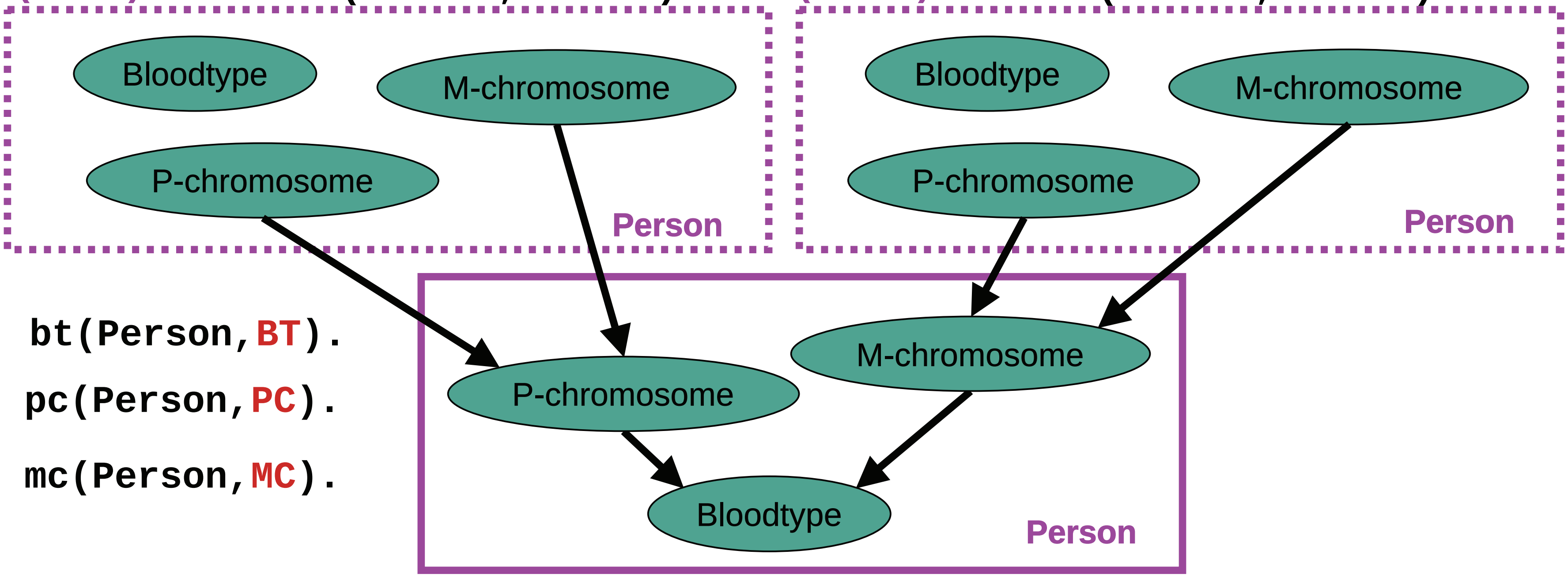
[Getoor,Koller, Pfeffer]



Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]

(Father) `father(Father,Person).` (Mother) `mother(Mother,Person).`




```
bt(Person,BT).
pc(Person,PC).
mc(Person,MC).
```

Dependencies (CPDs associated with):

```
bt(Person,BT) :- pc(Person,PC), mc(Person,MC).
pc(Person,PC) :- pc_father(Father,PCf), mc_father(Father,MCf).
```

View :

```
pc_father(Person,PCf) |  father(Father,Person),pc(Father,PC).
...
```

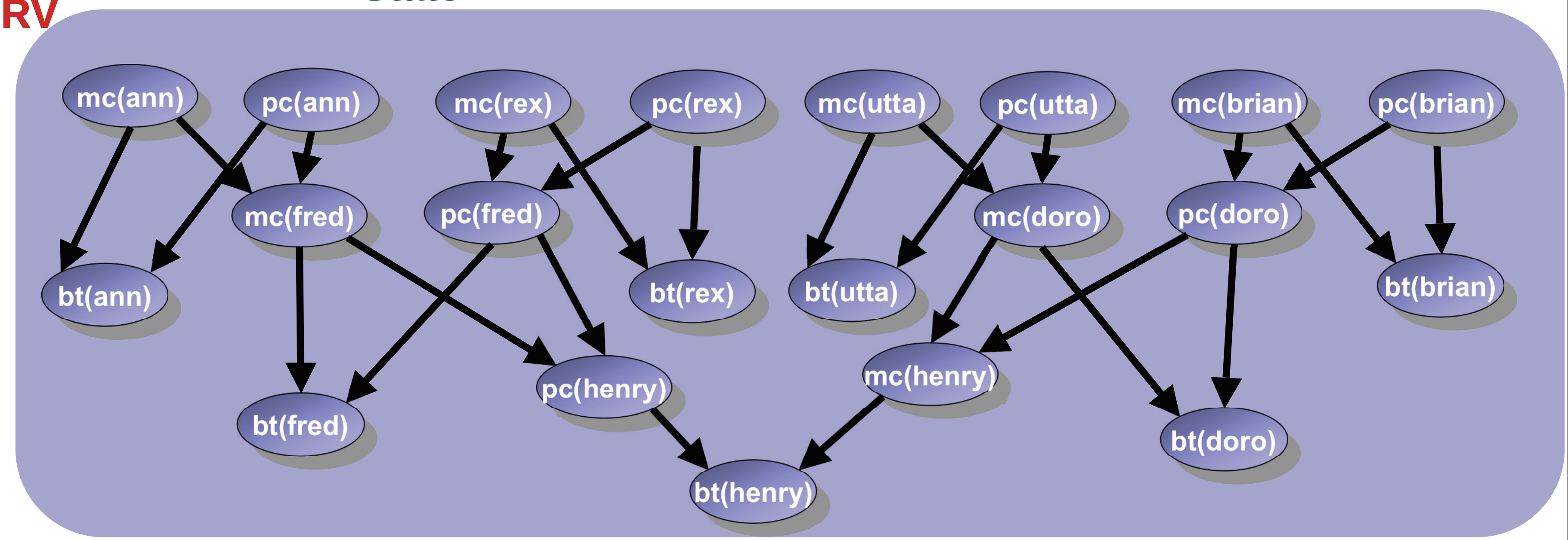
Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]

```
father(rex,fred).      mother(ann,fred).
father(brian,doro).    mother(utta, doro).
father(fred,henry).    mother(doro,henry).

pc_father(Person,PCf) |  father(Father,Person),pc(Father,PC).
...
mc(Person,MC) | pc_mother(Person,PCm), pc_mother(Person,MCm).
pc(Person,PC) | pc_father(Person,PCf), mc_father(Person,MCf).
bt(Person,BT) | pc(Person,PC), mc(Person,MC).
```

RV

State



Gene Regulation

[Segal et al.]

- System Biology
- Gene expression: two-phase process
  1. Gene is transcribed into mRNA — Measured by gene expression microarrays
  2. mRNA is translated Protein
- Genes that are similar expressed are often coregulated and involved in the same cellular processes
- Clustering: identification of clusters of genes and/or experiments that share similar expression patterns

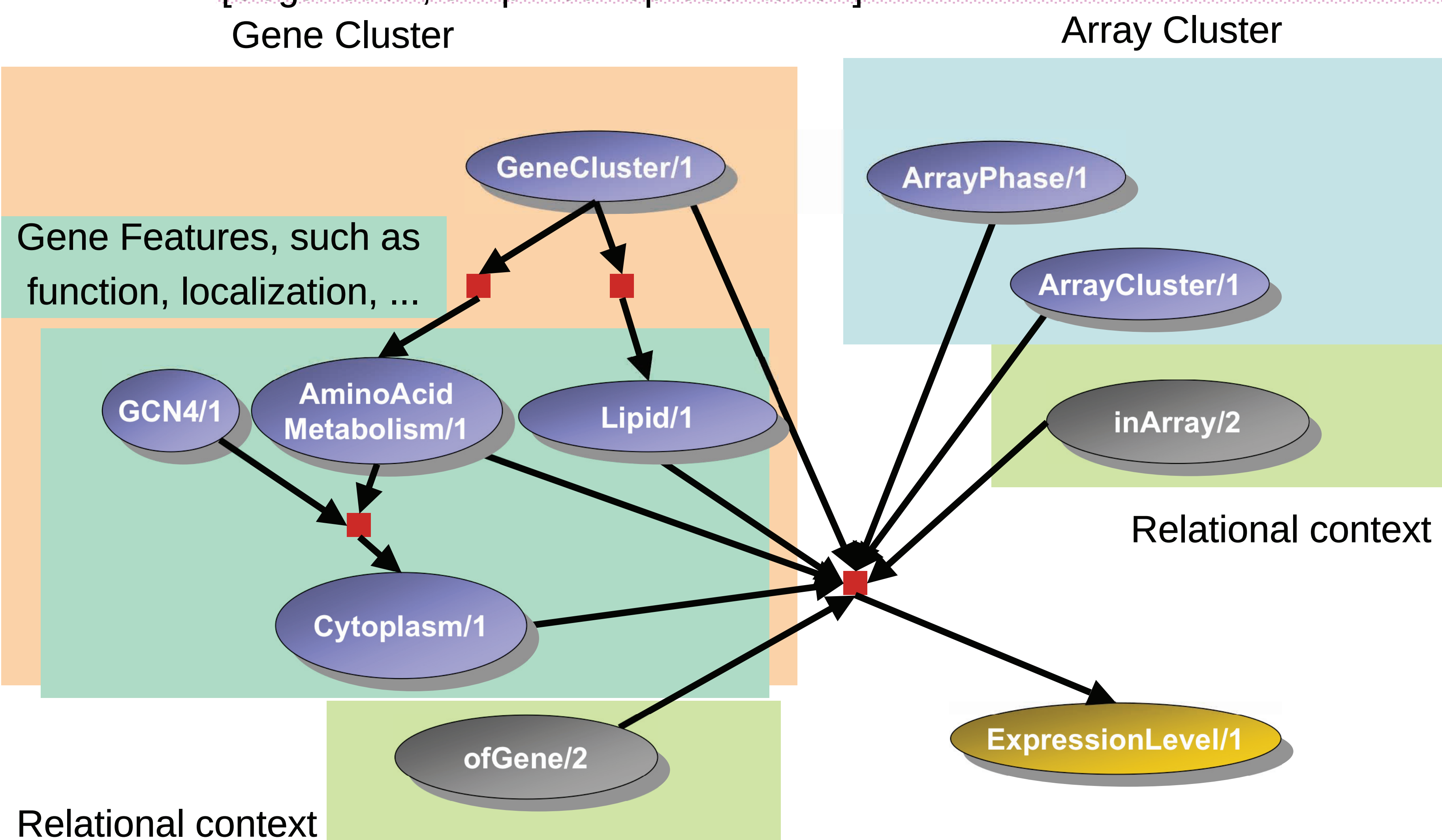
PRM Application: Gene Regulation

[Segal et al.]

- System Biology: heterogenous data
- Limitations of Clustering:
  - Similarities over all measurements
  - Difficult to incorporate readily background knowledge such as clinical data or experimental details

PRM Application: Gene Regulation

[Segal et al., simplified representation]



PRM Application: Gene Regulation

[Segal et al.]

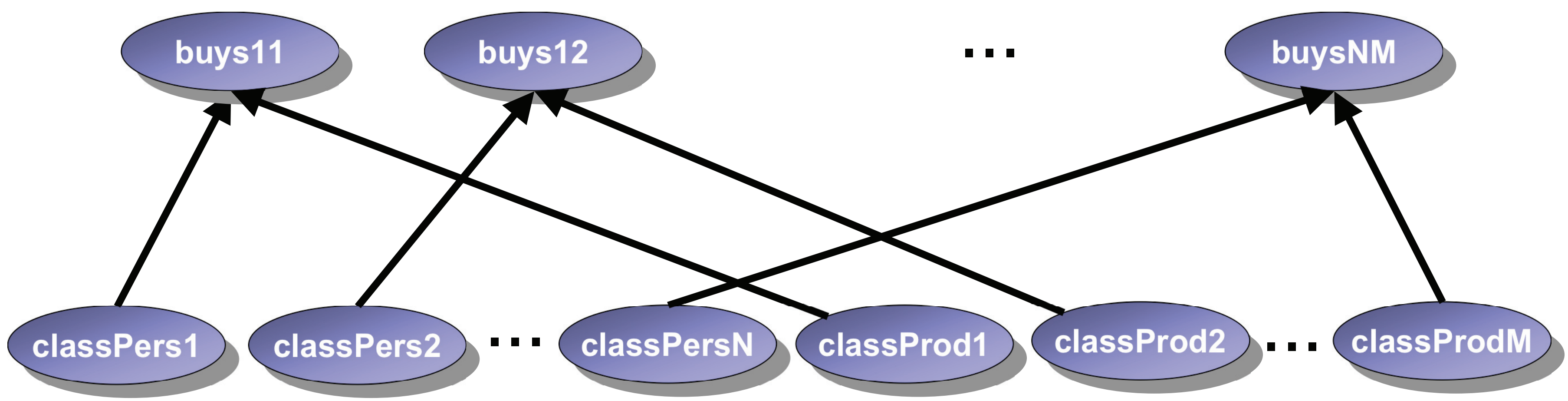
- Synthatic data: 1000 genes, 90 arrays (= 90.000 measurements), each gene 15 functions and 30 transcription factors.

	Cluster recovery	
	Naive Bayes	PRMs
Simulated data	90.8±0.42	98.4±1.07
Noisy simluated data	76.7±1.42	88.1±1.52

PRM Application: Collaborative Filtering

[Getoor, Sahami]

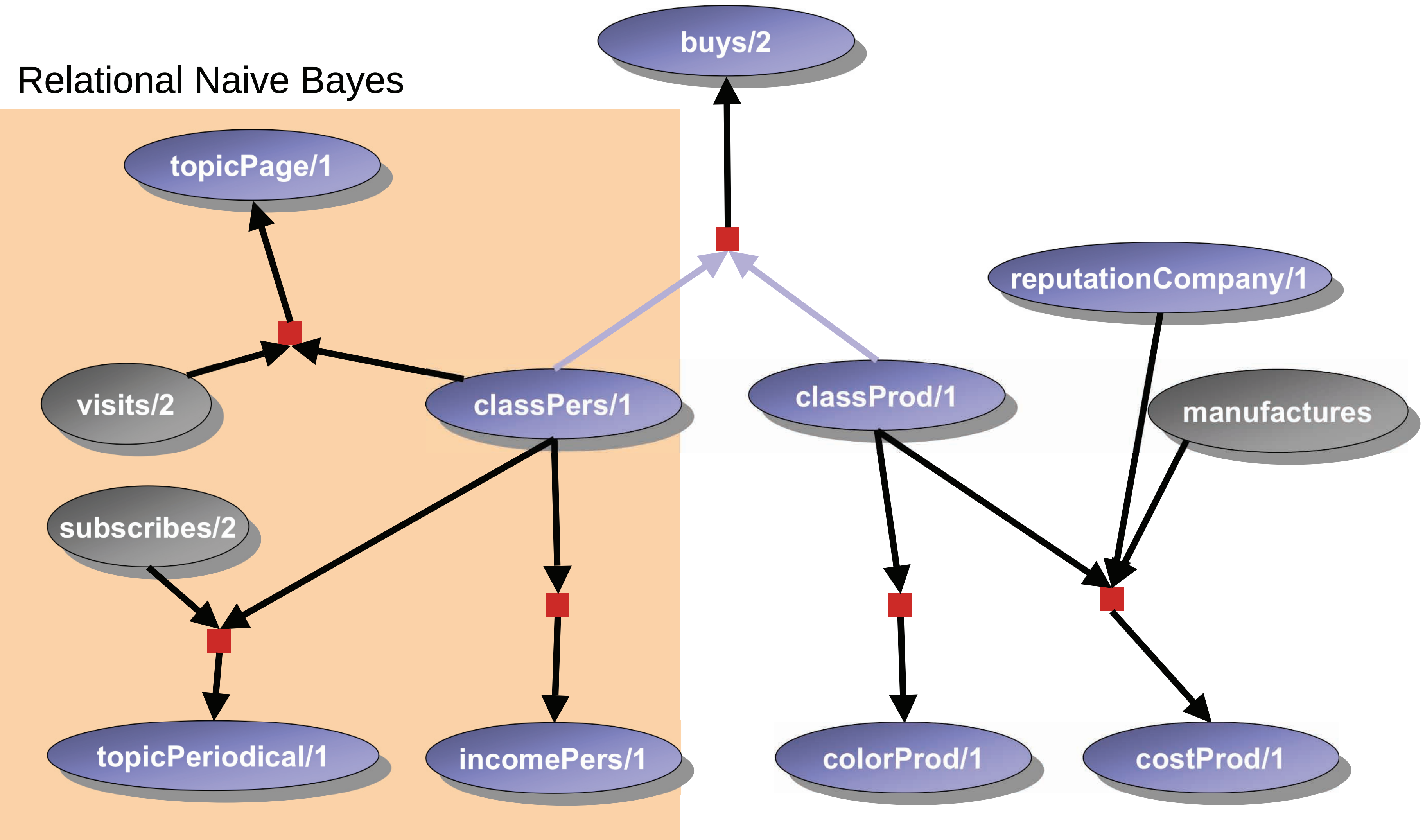
- User preference relationships for products / information.
- Traditionally: single dyactic relationship between the objects.



PRM Application: Collaborative Filtering

[Getoor, Sahami; simplified representation]

Relational Naive Bayes

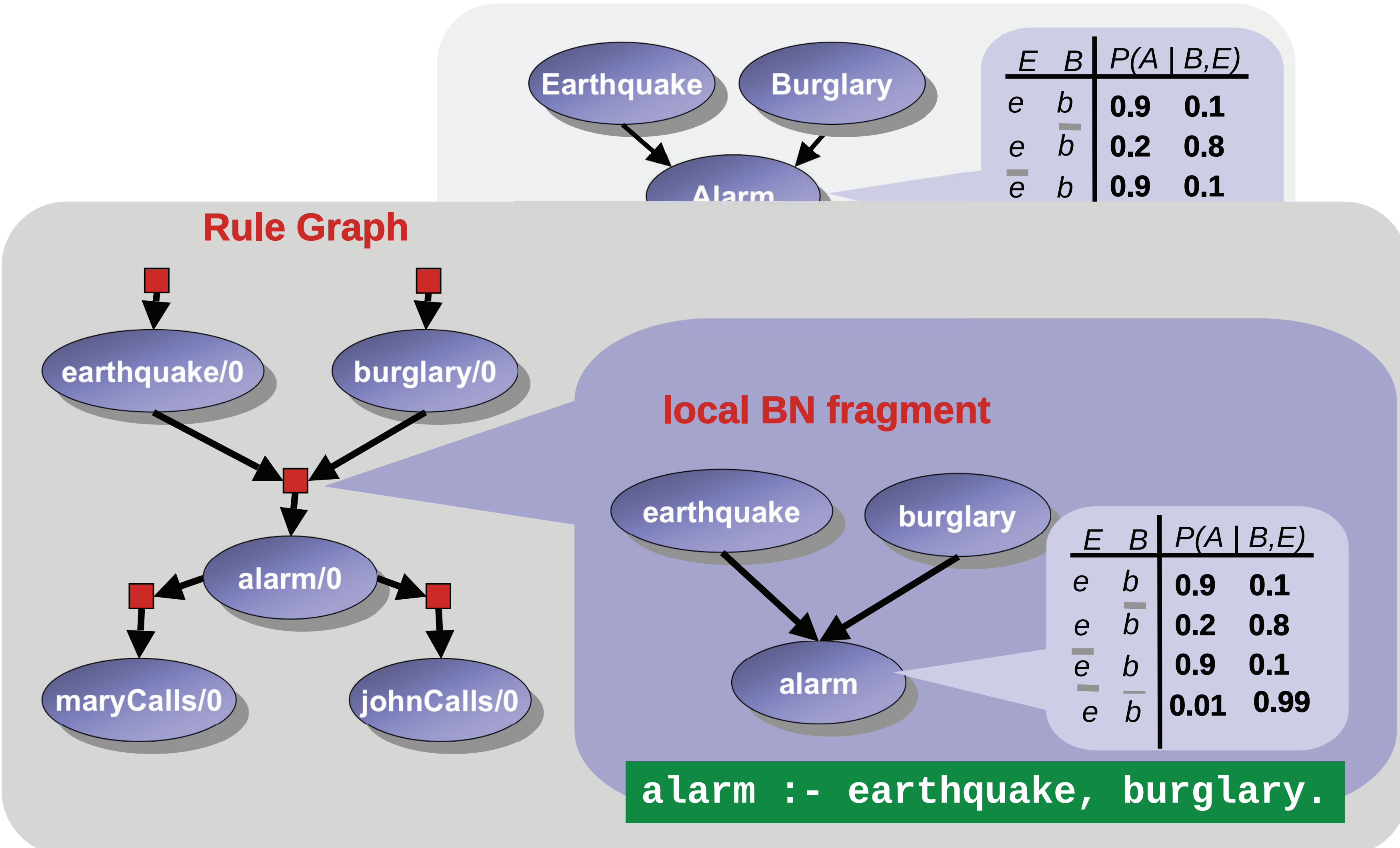


Probabilistic Relational Models (PRMs)

[Koller,Pfeffer,Getoor]

- Database View
- Unique Probability Distribution over finite Herbrand interpretations
  - No self-dependency
- Discrete and continuous RV
- BN used to do inference
- Graphical Representation
- BNs + extenstions: hierarchical PRMS, dynamic PRMs, relational uncertainty types
- Learning

Bayesian Logic Programs (BLPs)

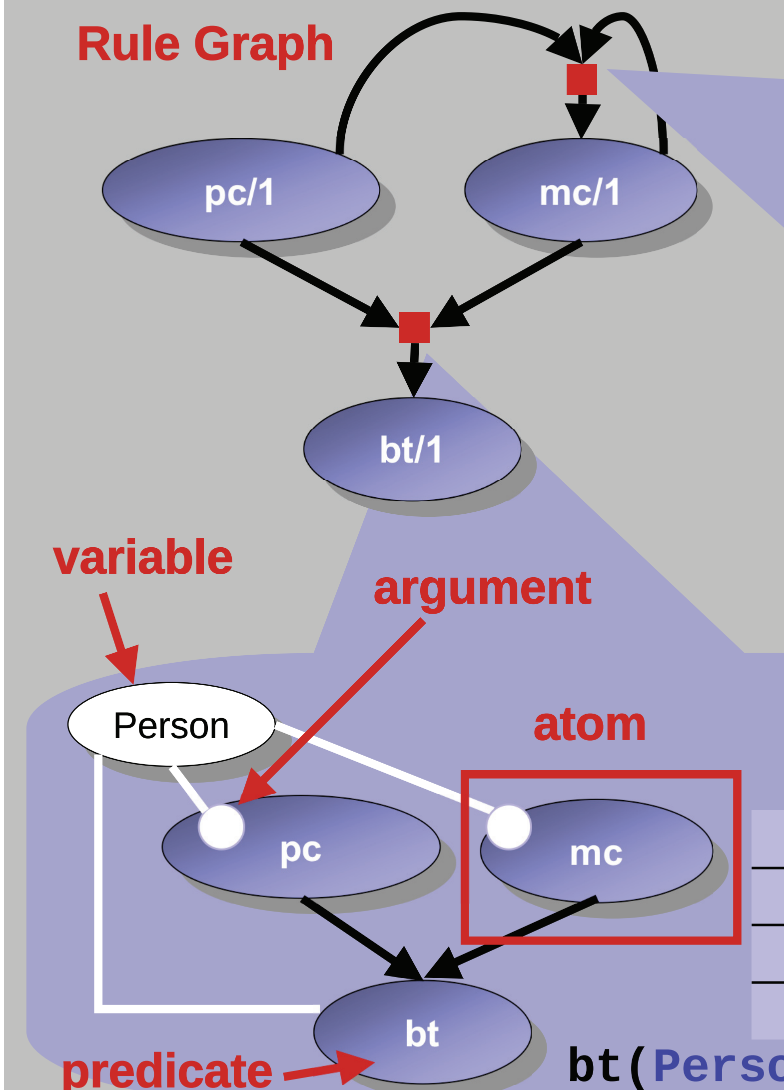


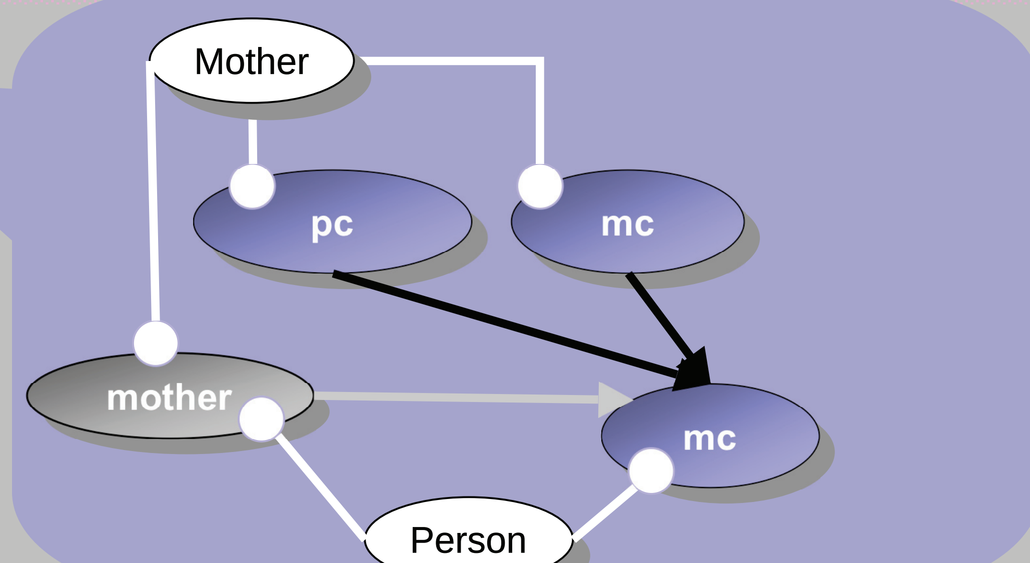
E	B	P(A | B,E)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1

E	B	P(A | B,E)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

```
alarm :- earthquake, burglary.
```

Bayesian Logic Programs (BLPs)



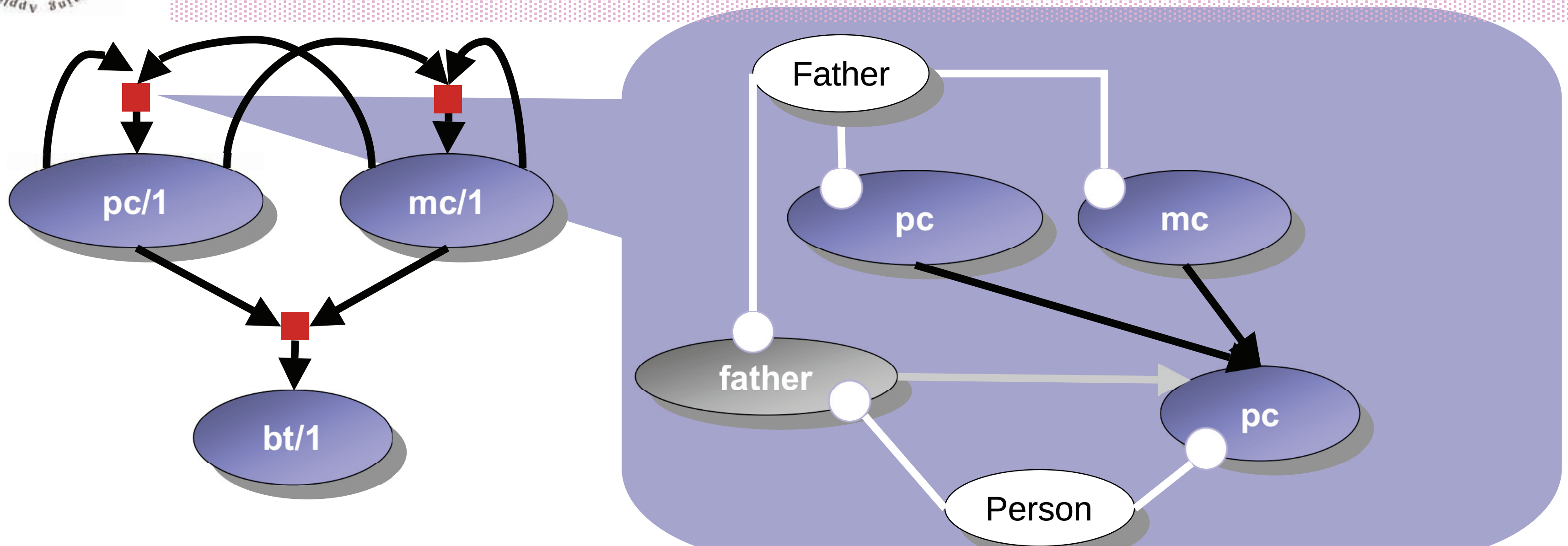


mc(Person)	pc(Mother)	mc(Mother)
(.9,.05,.05)	a	a
(.495,.495,.01)	a	b
...	...	...

bt(Person)	pc(Person)	mc(Person)
(.9,.03,.03,.03)	a	a
(.03,.03,.9,.03)	a	b
...	...	...

bt(Person) :- pc(Person),mc(Person).

Bayesian Logic Programs (BLPs)



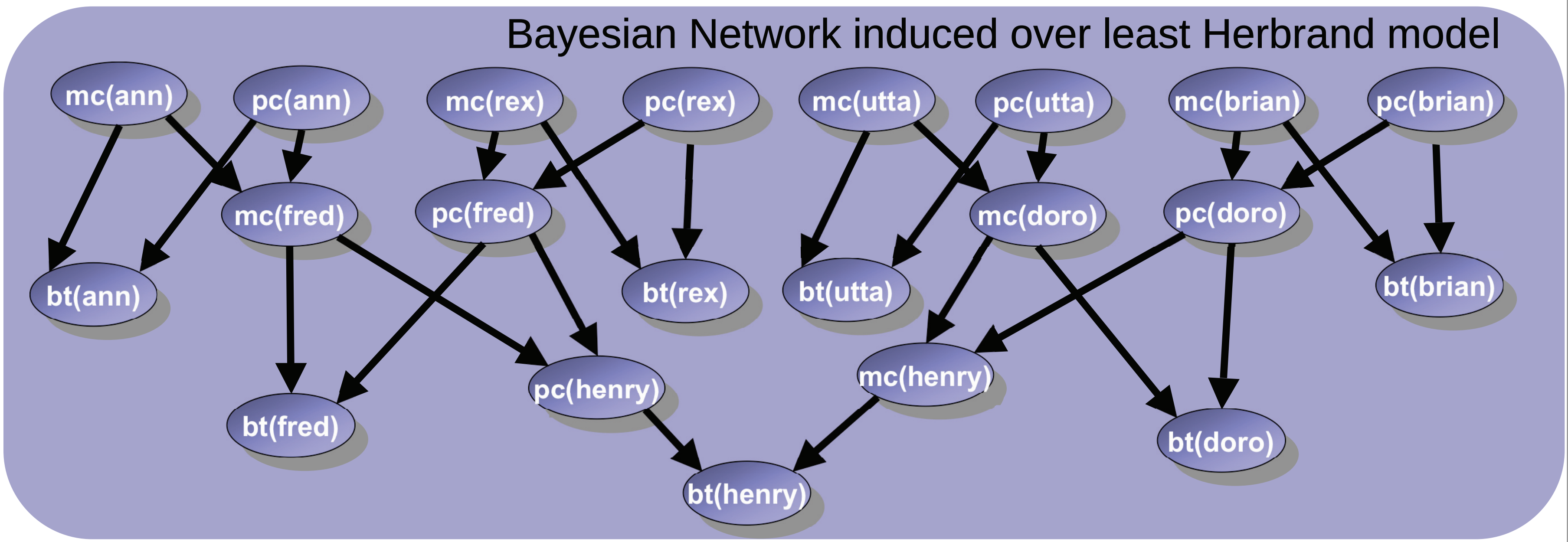
pc(Person)	pc(Father)	mc(Father)
(.9,.05,.05)	a	a
(.495,.495,.01)	a	b
...	...	...

```
mc(Person) | mother(Mother,Person), pc(Mother),mc(Mother).
pc(Person) | father(Father,Person), pc(Father),mc(Father).
bt(Person) | pc(Person),mc(Person).
```

Bayesian Logic Programs (BLPs)

```
father(rex,fred).      mother(ann,fred).
father(brian,doro).    mother(utta, doro).
father(fred,henry).    mother(doro,henry).
```

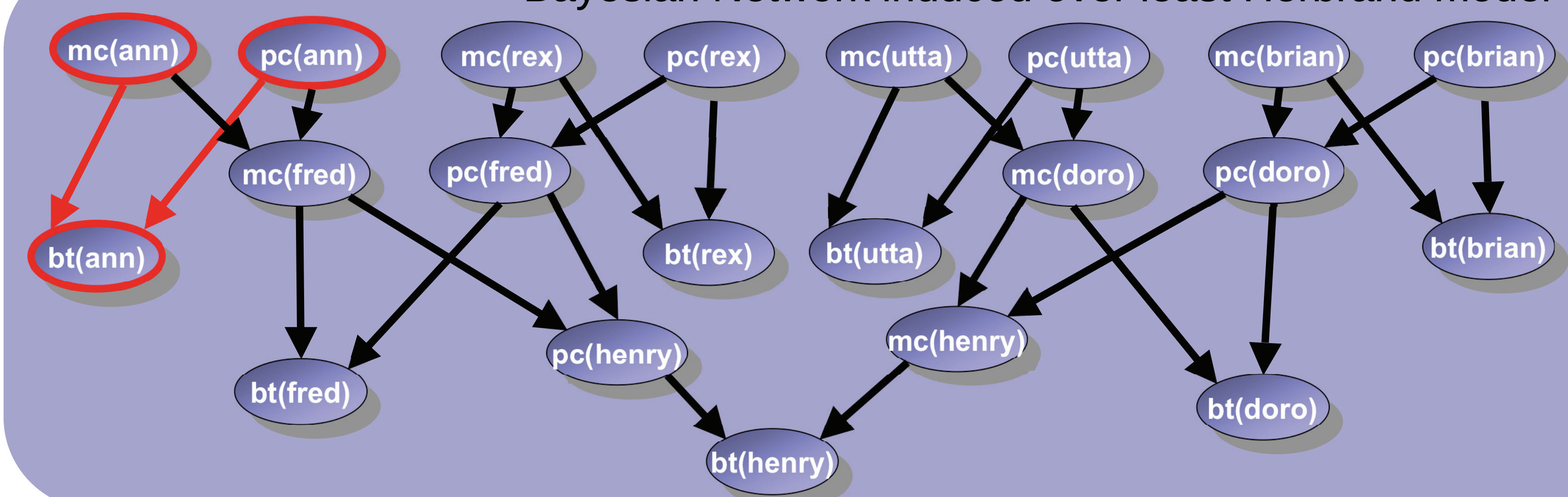
```
mc(Person) | mother(Mother,Person), pc(Mother),mc(Mother).
pc(Person) | father(Father,Person), pc(Father),mc(Father).
bt(Person) | pc(Person),mc(Person).
```



Answering Queries

P(`bt(ann)`**) ?**

Bayesian Network induced over least Herbrand model

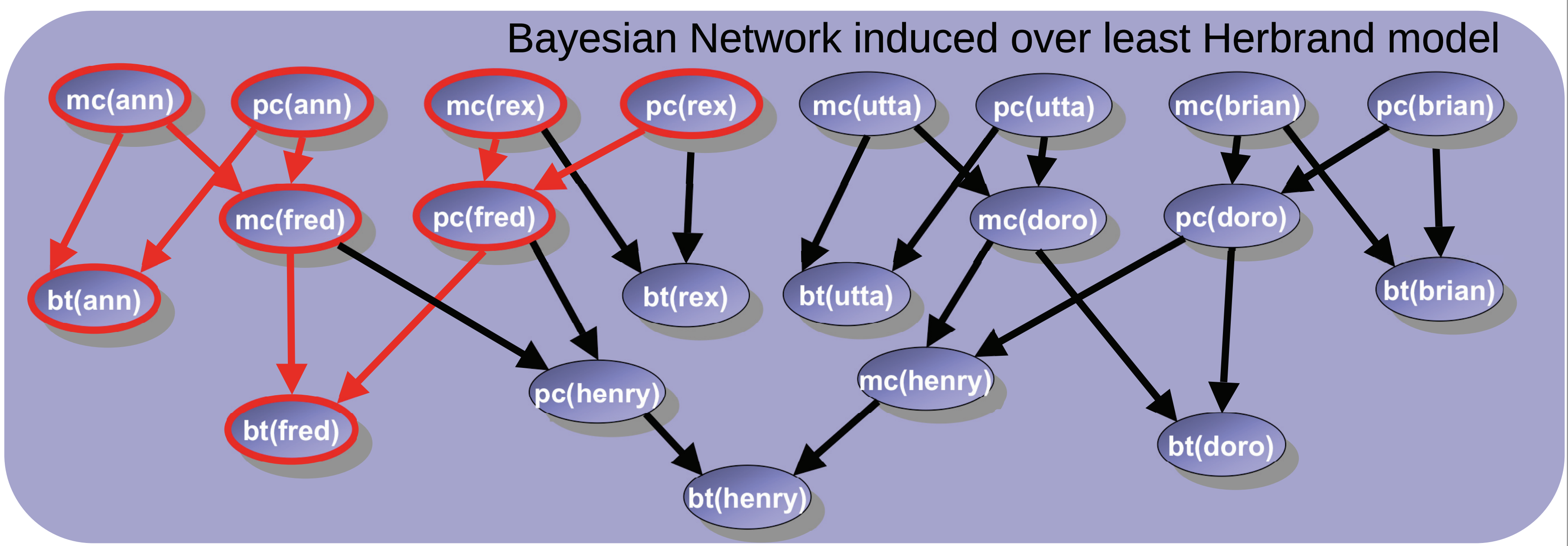


Answering Queries

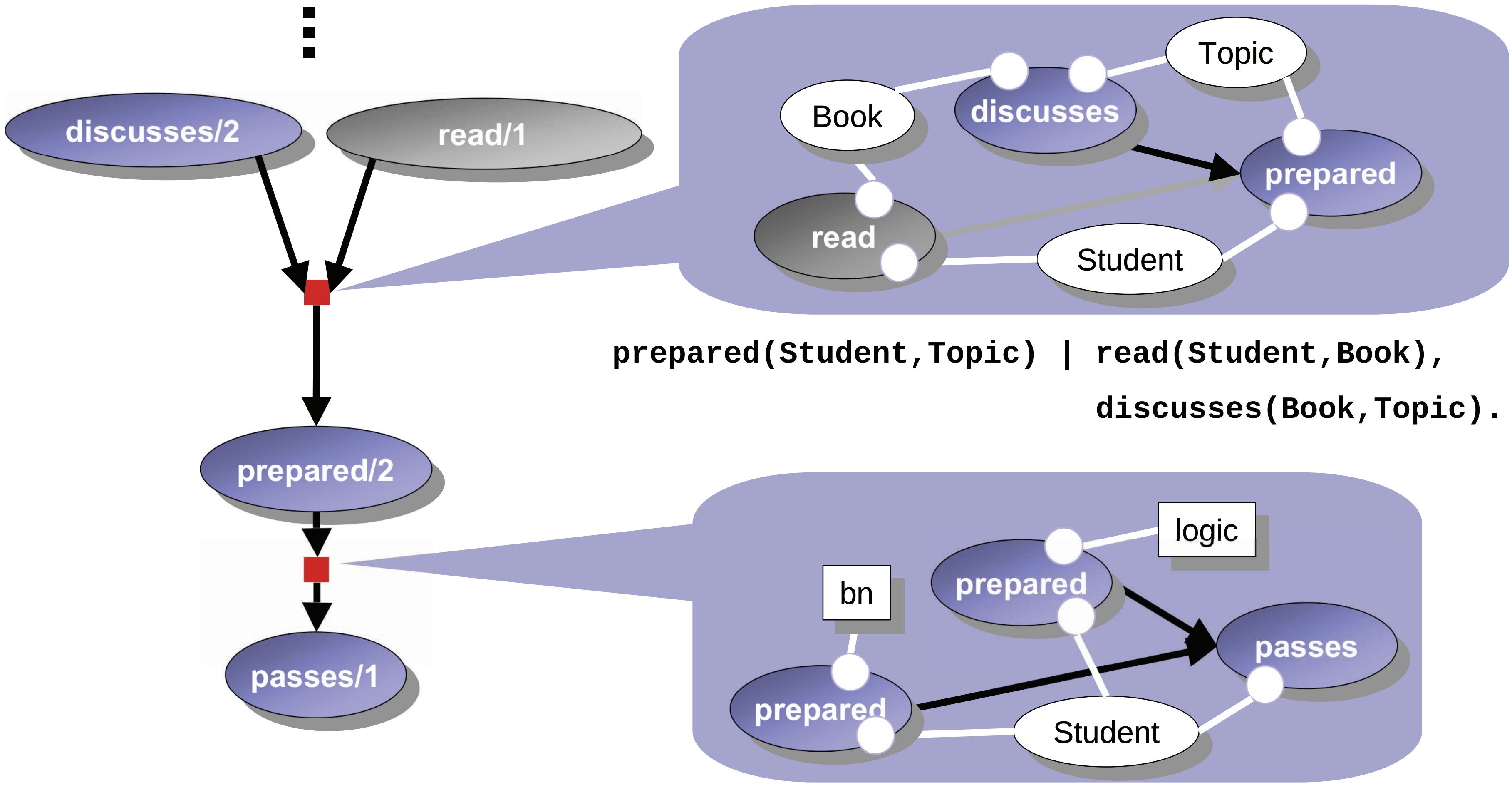
Bayes' rule

$$\mathbf{P}(\texttt{bt(ann)} \mid \texttt{bt(fred)}) = \frac{\mathbf{P}(\texttt{bt(ann)},\texttt{bt(fred)})}{\mathbf{P}(\texttt{bt(fred)})}$$

P(`bt(ann), bt(fred)`**) ?**



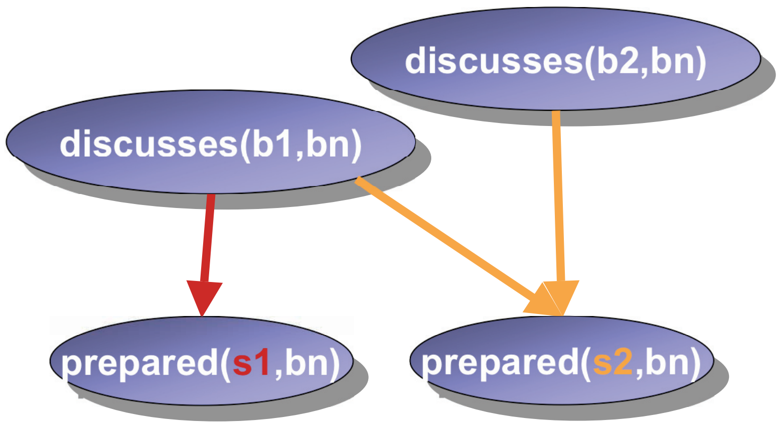
Combining Partial Knowledge

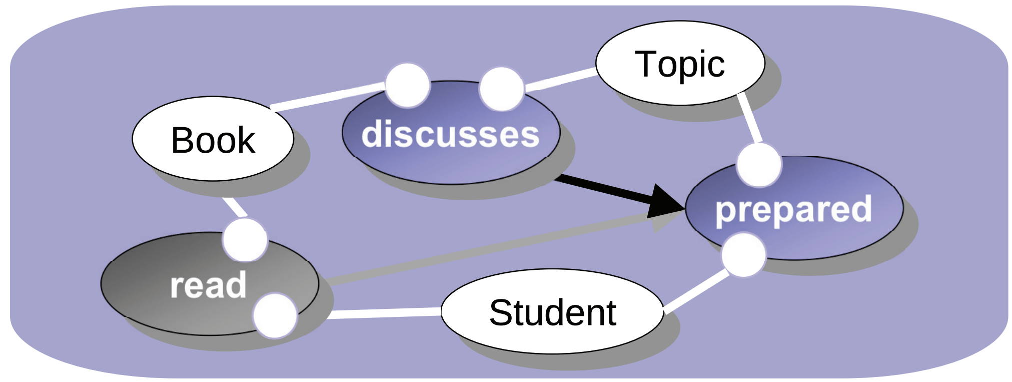



```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

```
passes(Student) | prepared(Student,bn),
                  prepared(Student,logic).
```

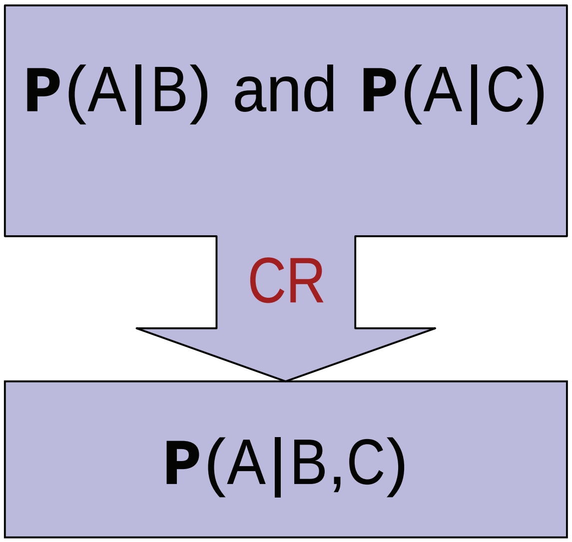
Combining Partial Knowledge

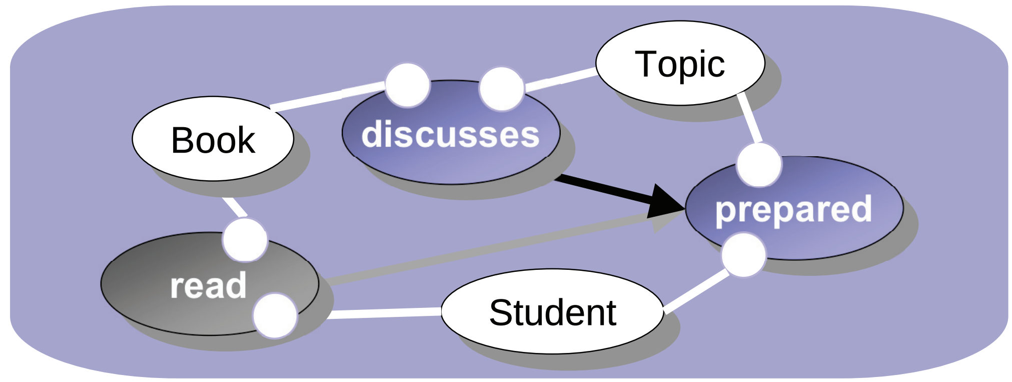




- variable # of parents for `prepared/2` due to `read/2`
  - whether a student prepared a topic depends on the books she read
- CPD only for one book-topic pair

Combining Rules

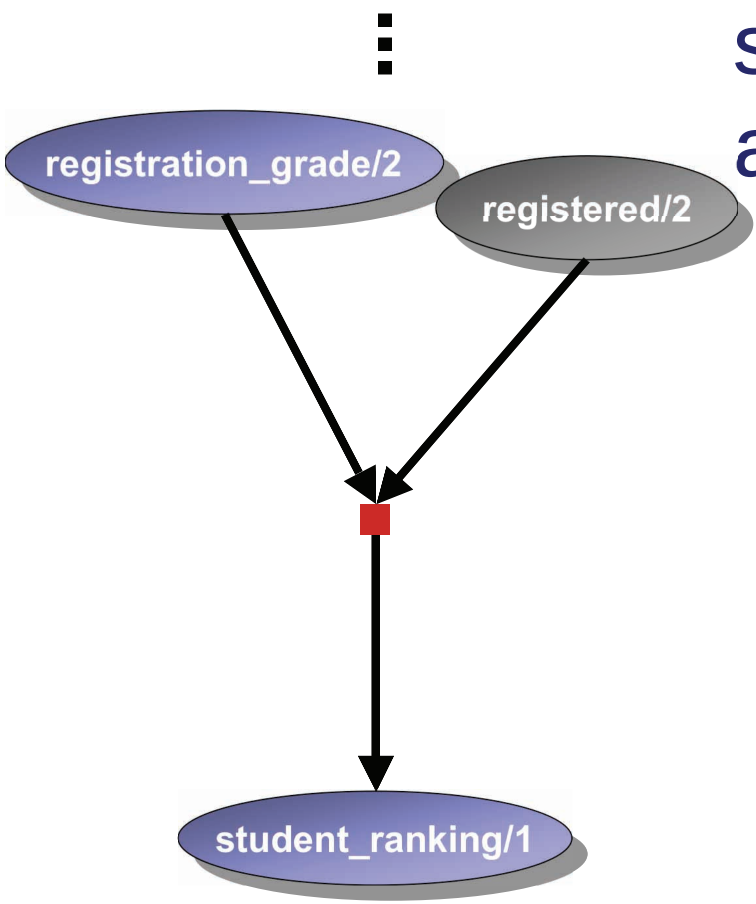




- Any algorithm which
  - has an empty output if and only if the input is empty
  - combines a set of CPDs into a single (combined) CPD
- E.g. noisy-or, regression, ...

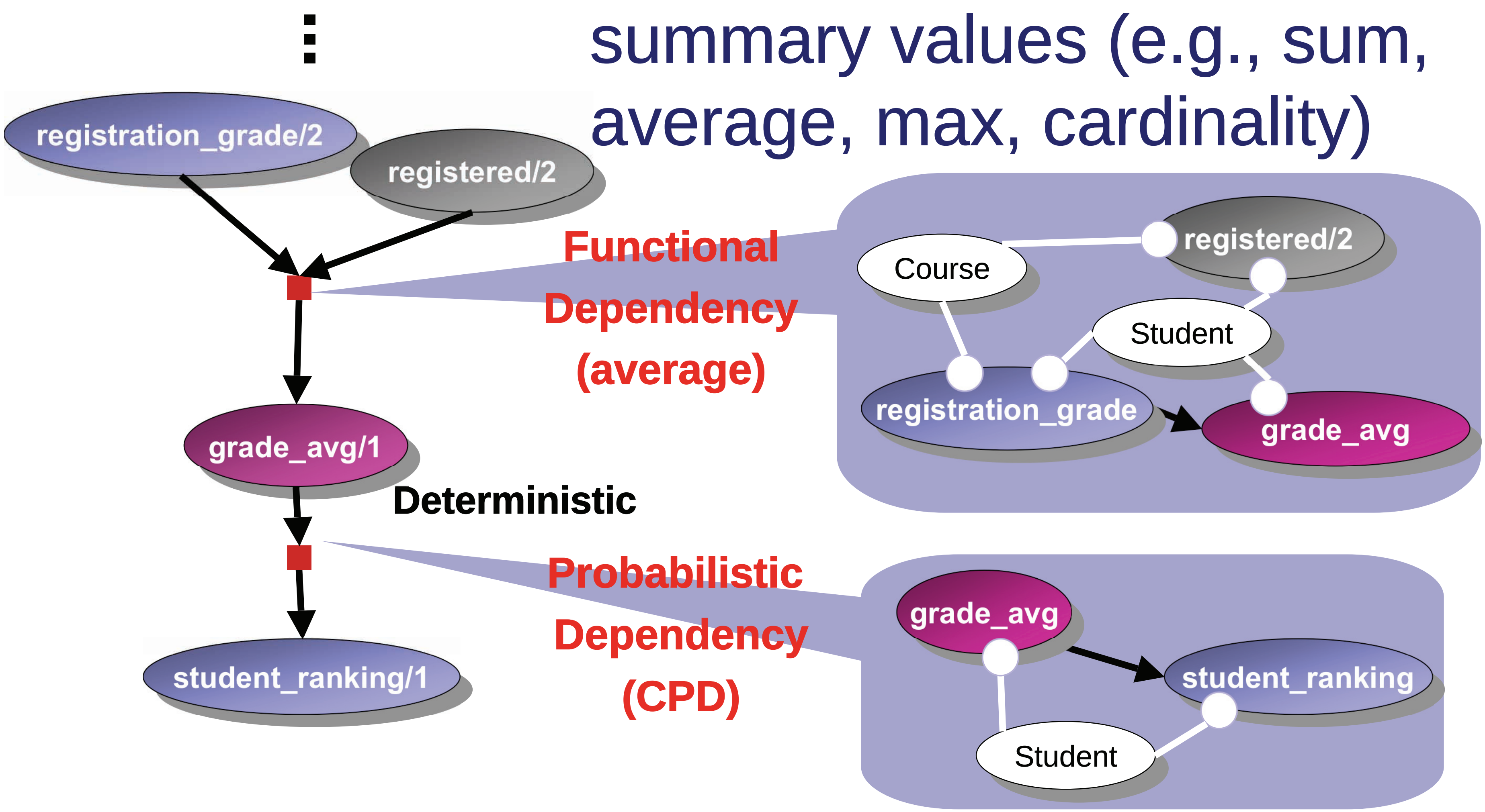
Aggregates

Map multisets of values to summary values (e.g., sum, average, max, cardinality)



Aggregates

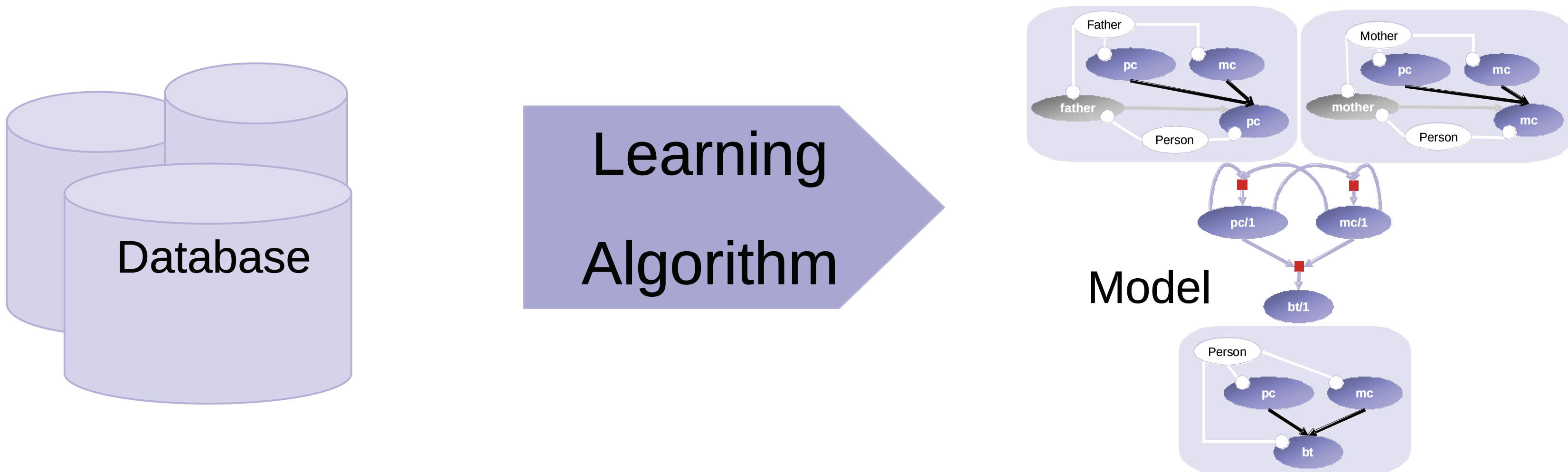
Map multisets of values to summary values (e.g., sum, average, max, cardinality)



Bayesian Logic Programs (BLPs)

- Unique probability distribution over Herbrand interpretations
  - Finite branching factor, finite proofs, no self-dependency
- Highlight
  - Separation of qualitative and quantitative parts
  - Functors
- Graphical Representation
- Discrete and continuous RV
- BNs, DBNs, HMMs, SCFGs, Prolog ...
- Subsume PRMs
- Learning

Learning Tasks



- Parameter Estimation
  - Numerical Optimization Problem
- Model Selection
  - Combinatorical Search

What is the data about?

RVs + States = (partial) Herbrand interpretation

Probabilistic learning from interpretations

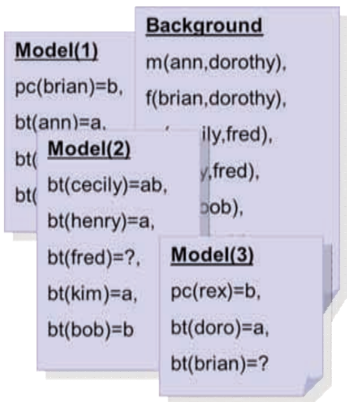
<u>Background</u>
m(ann,dorothy),
f(brian,dorothy),
m(cecily,fred),
f(henry,fred),
f(fred,bob),
m(kim,bob),
...

<u>Family(1)</u>
pc(brian)=b,
bt(ann)=a,
bt(brian)=?,
bt(dorothy)=a

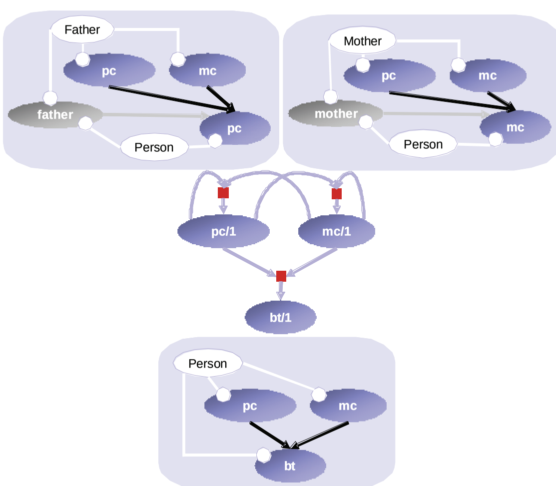
<u>Family(2)</u>
bt(cecily)=ab,
pc(henry)=a,
mc(fred)=?,
bt(kim)=a,
pc(bob)=b

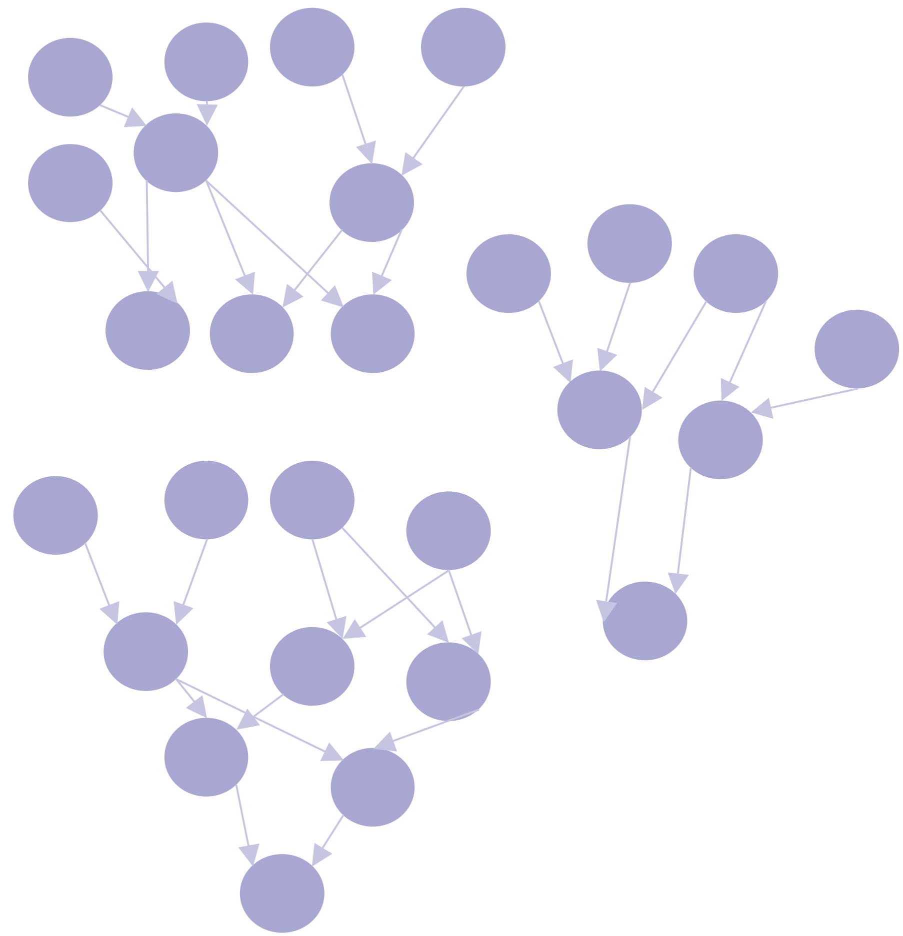
<u>Family(3)</u>
pc(rex)=b,
bt(doro)=a,
bt(brian)=?

Parameter Estimation

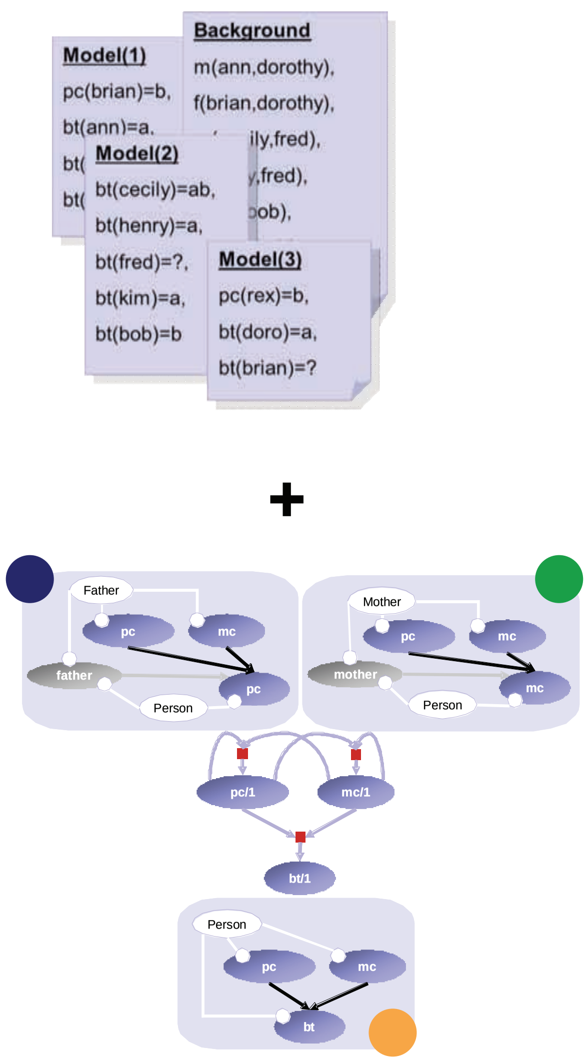


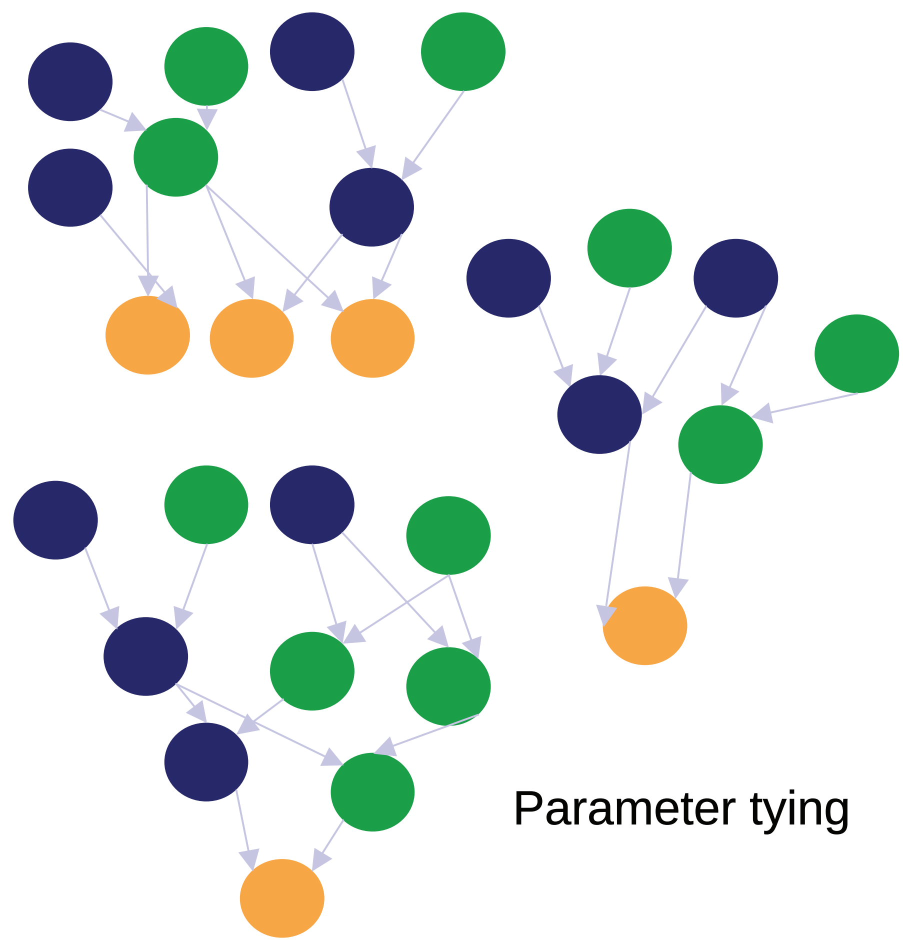
+





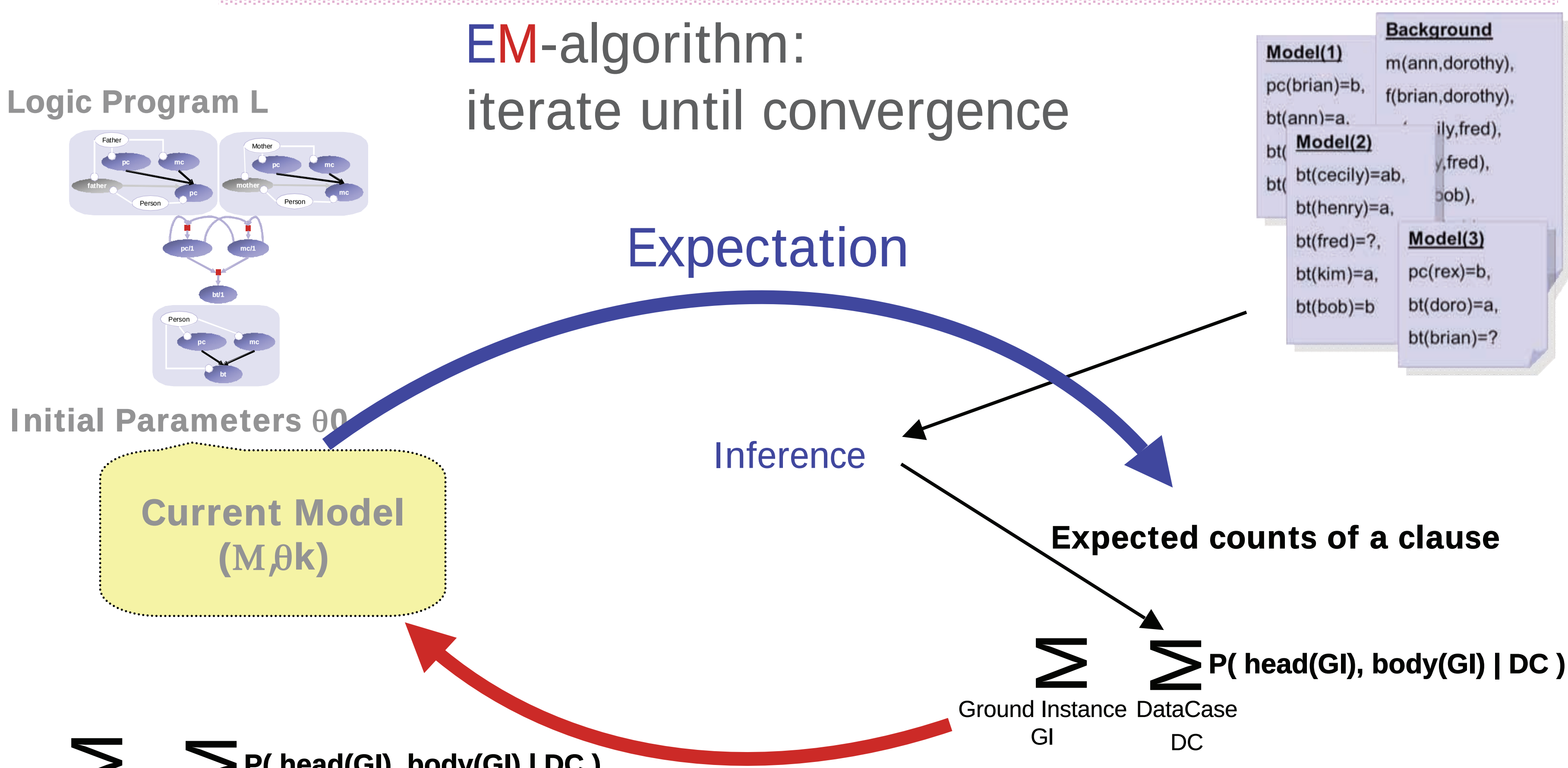
Parameter Estimation





Expectation Maximization

EM-algorithm:
iterate until convergence



$$\sum_{\text{Ground Instance GI}} \sum_{\text{DataCase DC}} P(\,\text{head(GI), body(GI)} \mid \text{DC}\,)$$

$$\frac{\displaystyle\sum_{\text{Ground Instance GI}} \sum_{\text{DataCase DC}} P(\,\text{head(GI), body(GI)} \mid \text{DC}\,)}{\displaystyle\sum_{\text{Ground Instance GI}} \sum_{\text{DataCase DC}} P(\,\text{body(GI)} \mid \text{DC}\,)}$$

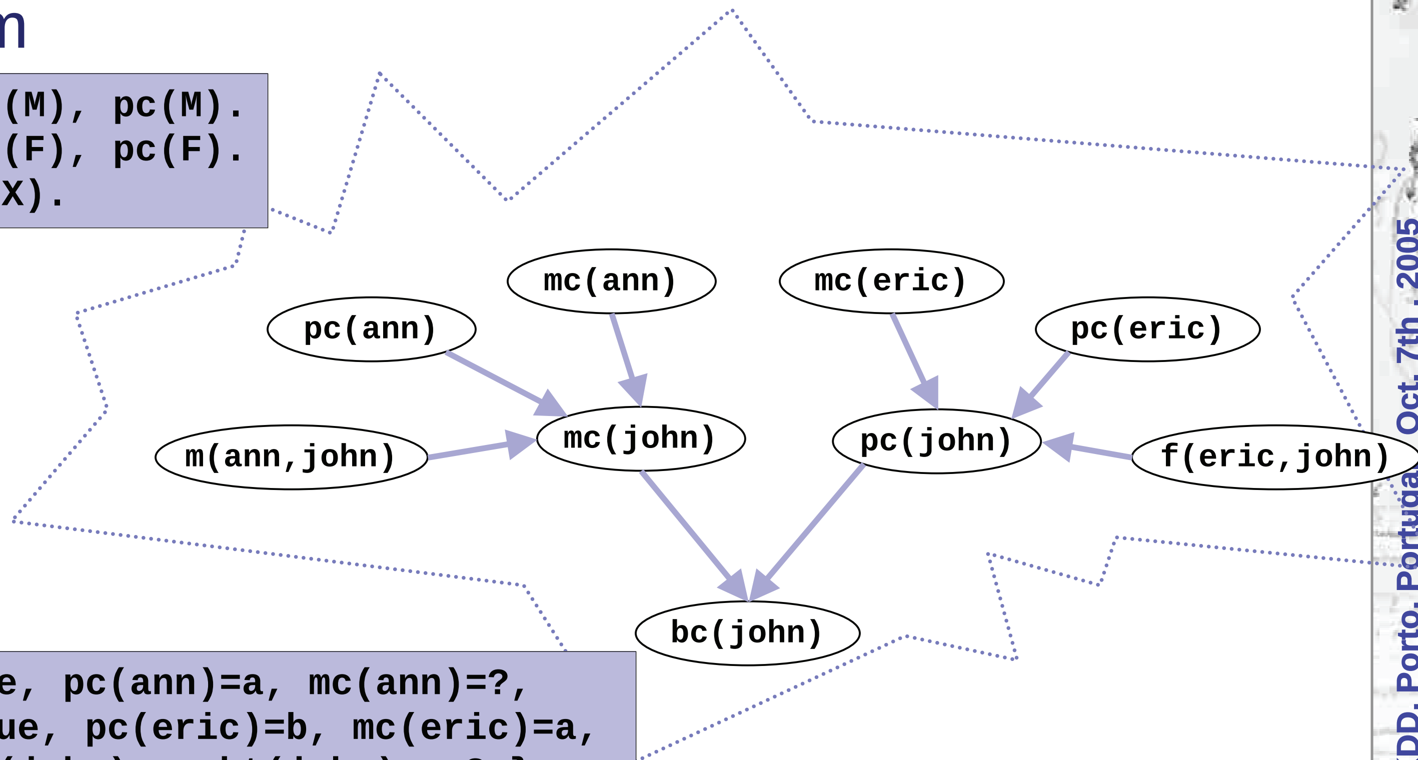
Model Selection

- Combination of ILP and BN learning
- Combinatorical search for hypo M* s.t.
  - M* logically covers the data D
  - M* is optimal w.r.t. some scoring function score, i.e., $M^* = \text{argmax}_M \text{score}(M,D)$.
- Again, the prob. ILP approach *highlights*
  - *Refinement operators*
  - *Background knowledge*
  - *Language bias*
  - *Search bias*

Example

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```



Data cases

```
{m(ann,john)=true, pc(ann)=a, mc(ann)=?,
 f(eric,john)=true, pc(eric)=b, mc(eric)=a,
 mc(john)=ab, pc(john)=a, bt(john) = ? }

...
```

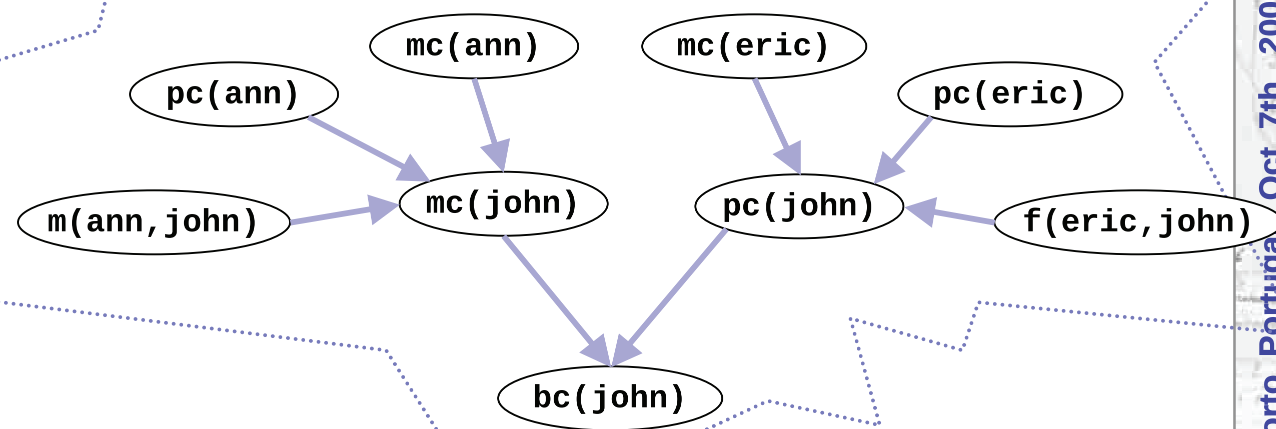
Example

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```



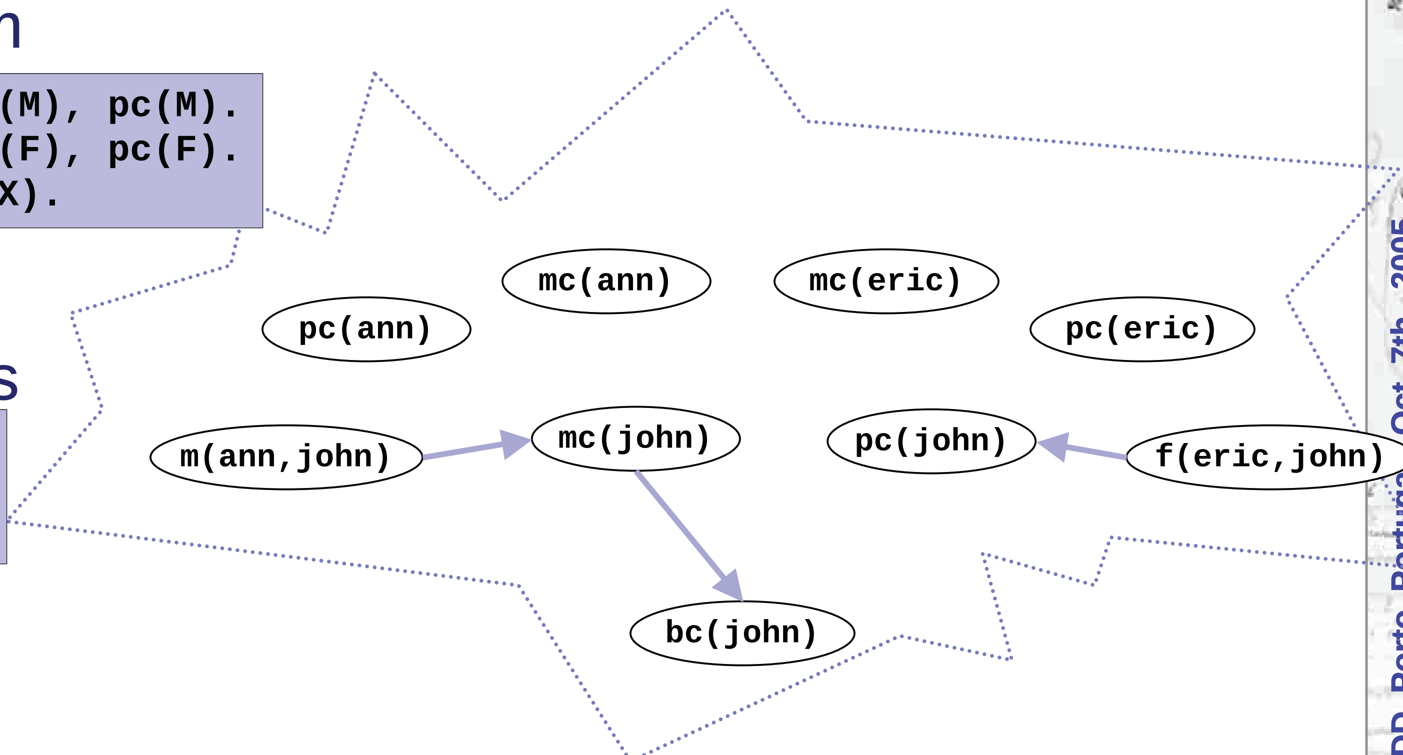
Example

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```



Example

Original program

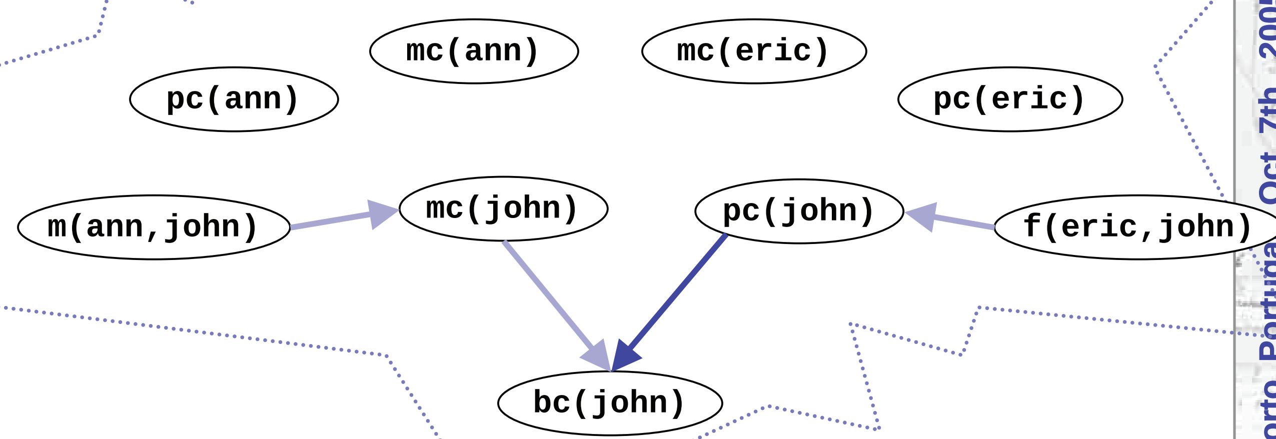
```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```

Refinement

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```



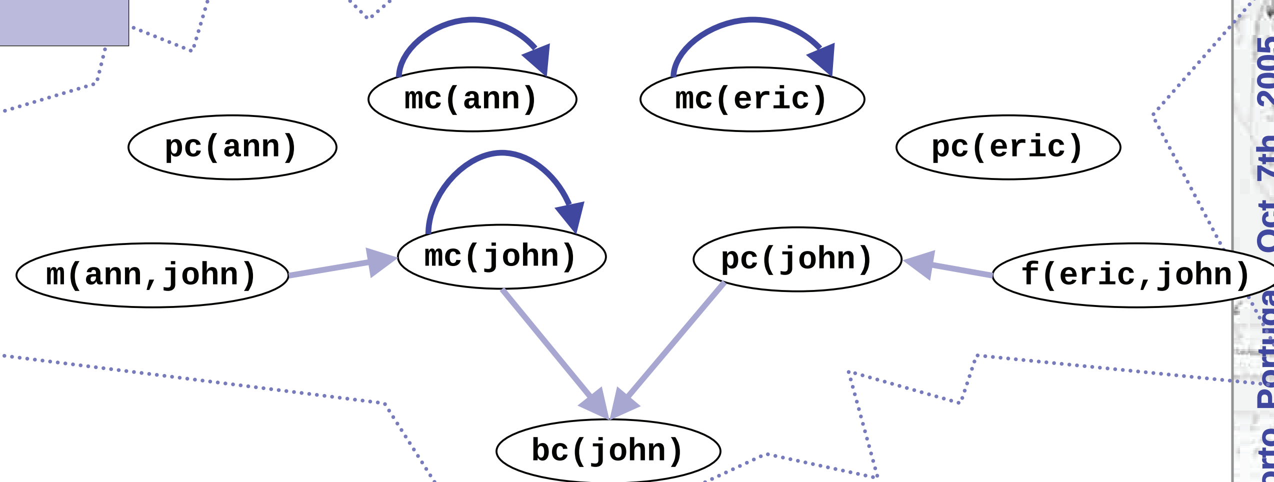
Example

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```



Refinement

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

Refinement

```
mc(X) | m(M,X),mc(X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

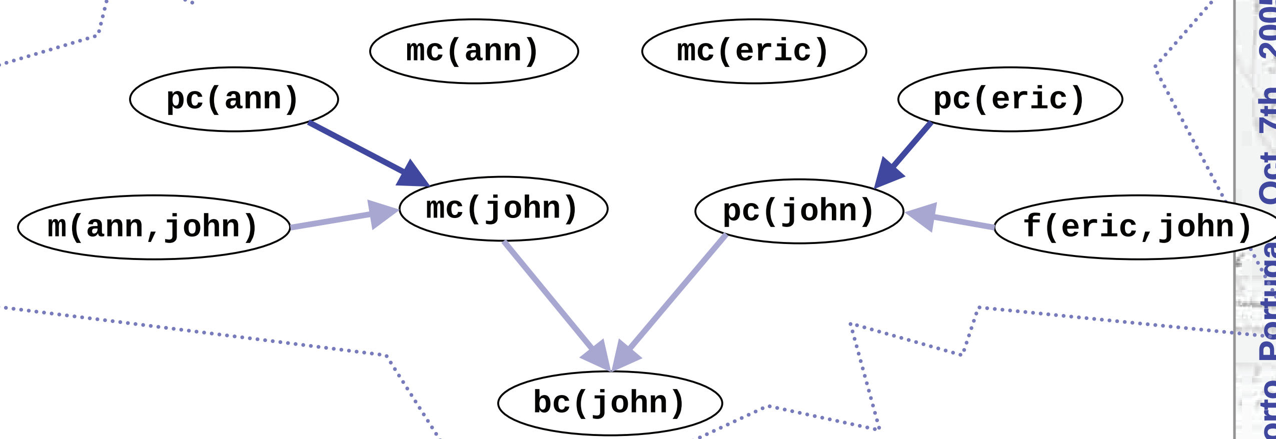
Example

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```



Refinement

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

Refinement

```
mc(X) | m(M,X),pc(X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

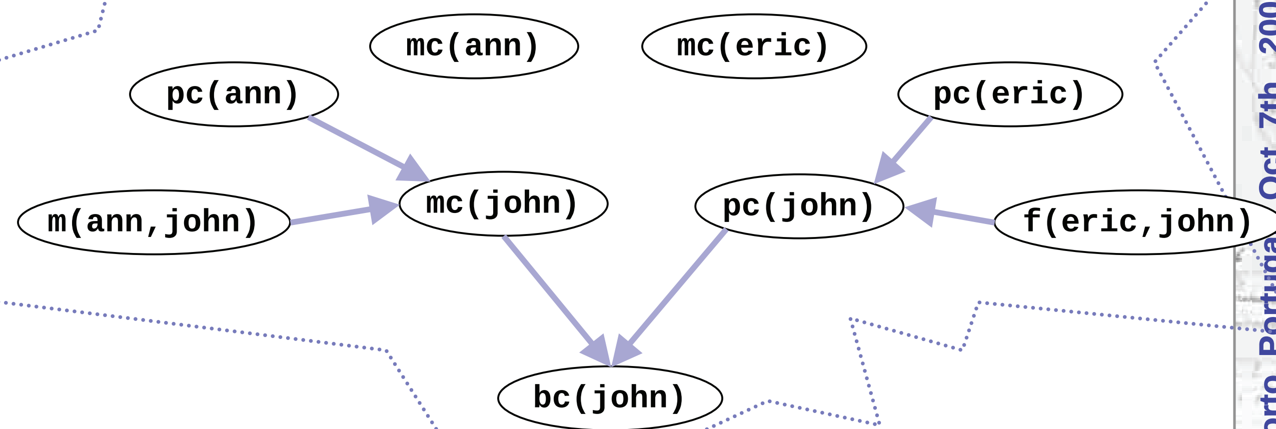
Example

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```



Refinement

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

Refinement

```
mc(X) | m(M,X),pc(X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

...

Undirected Probabilistic Relational Models

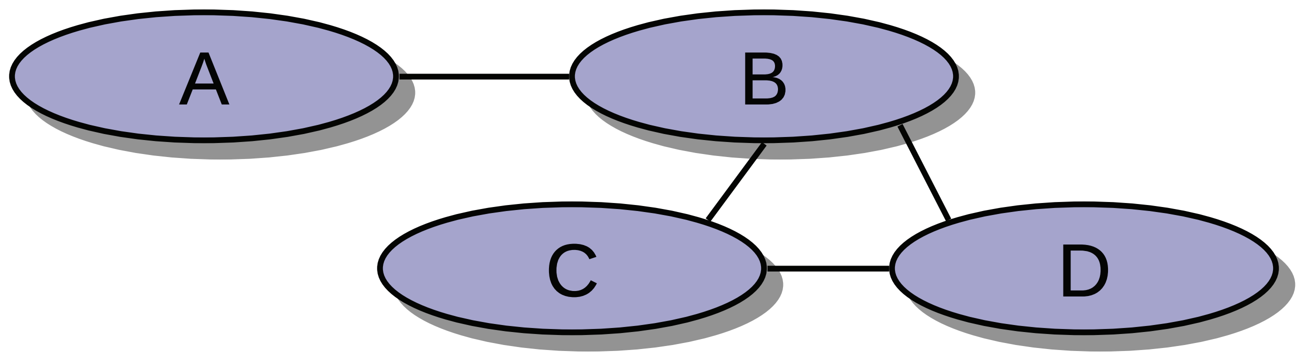
- So far, directed graphical models only
- Impose acyclicity constraint

- Undirected graphical models do not impose the acyclicity constraint

Undirected Probabilistic Relational Models

- Two approaches
  - Relational Markov Networks (RMNs)
    - (Taskar et al.)
  - Markov Logic Networks (MLNs)
    - (Anderson et al.)
- Idea
  - Semantics defined in terms of Markov Networks (undirected graphical models)
  - More natural for certain applications
- RMNs ~ undirected PRM
- MLNs ~ undirected BLP

Undirected Graphical Models/ Markov Networks



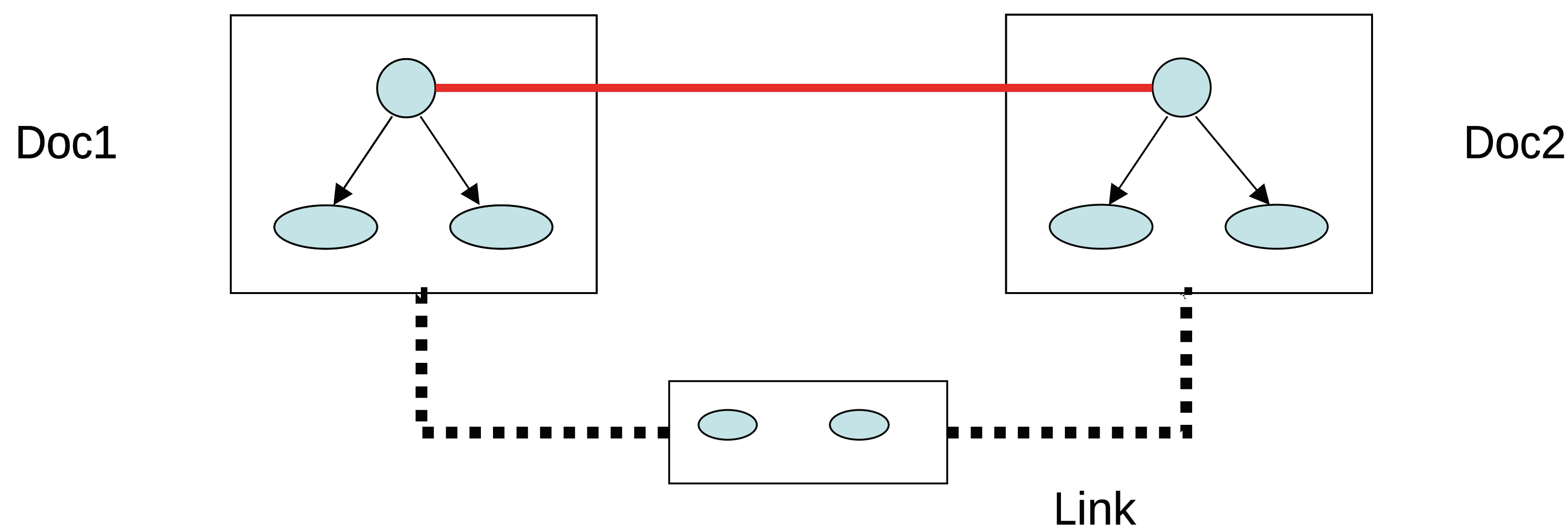
- To each clique c, a potential ϕ_c is associated
- Given the values $\mathbf{v}$ of all nodes in the Markov Network

$$P(\mathbf{v}) = \frac{1}{Z}\prod_{c \in C(G)} \phi_c(\mathbf{v}_c) \qquad Z = \sum_{\mathbf{v}} \prod_{c \in C(G)} \phi_c(\mathbf{v}_c')$$

$$\log P(\mathbf{v}) = \sum_c \mathbf{w}_c \cdot \mathbf{f}_c(v_c) - \log Z = \mathbf{w} \cdot \mathbf{f}(v) - \log Z$$

Relational Markov Networks

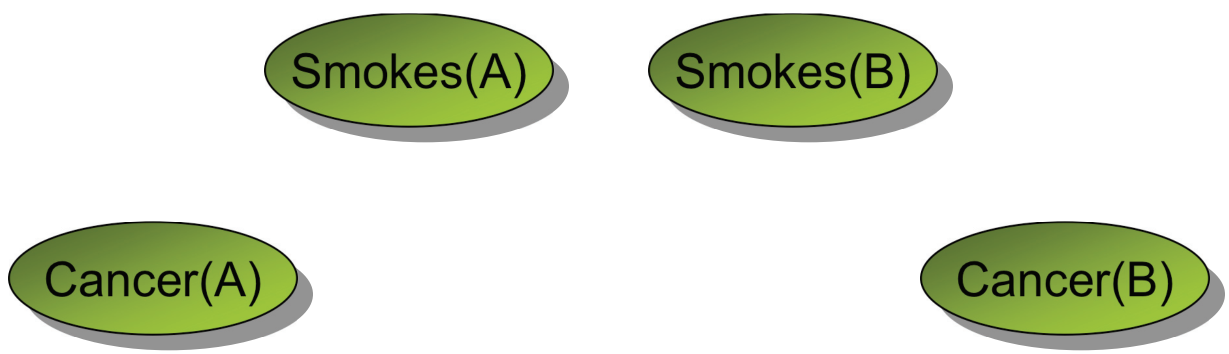
SELECT doc1.Category,doc2.Category
FROM doc1,doc2,Link link
WHERE link.From=doc1.key and link.To=doc2.key



Markov Logic Networks

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

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

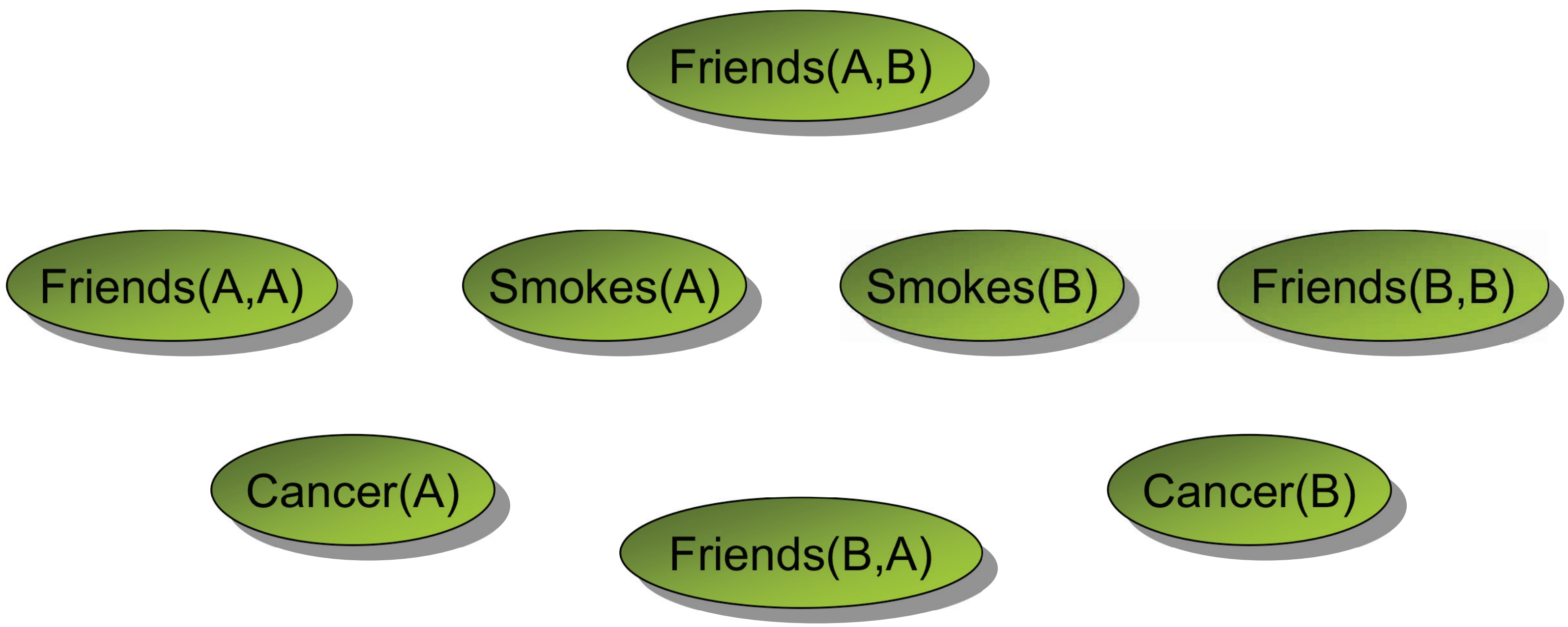


slides by Pedro Domingos

Markov Logic Networks

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)

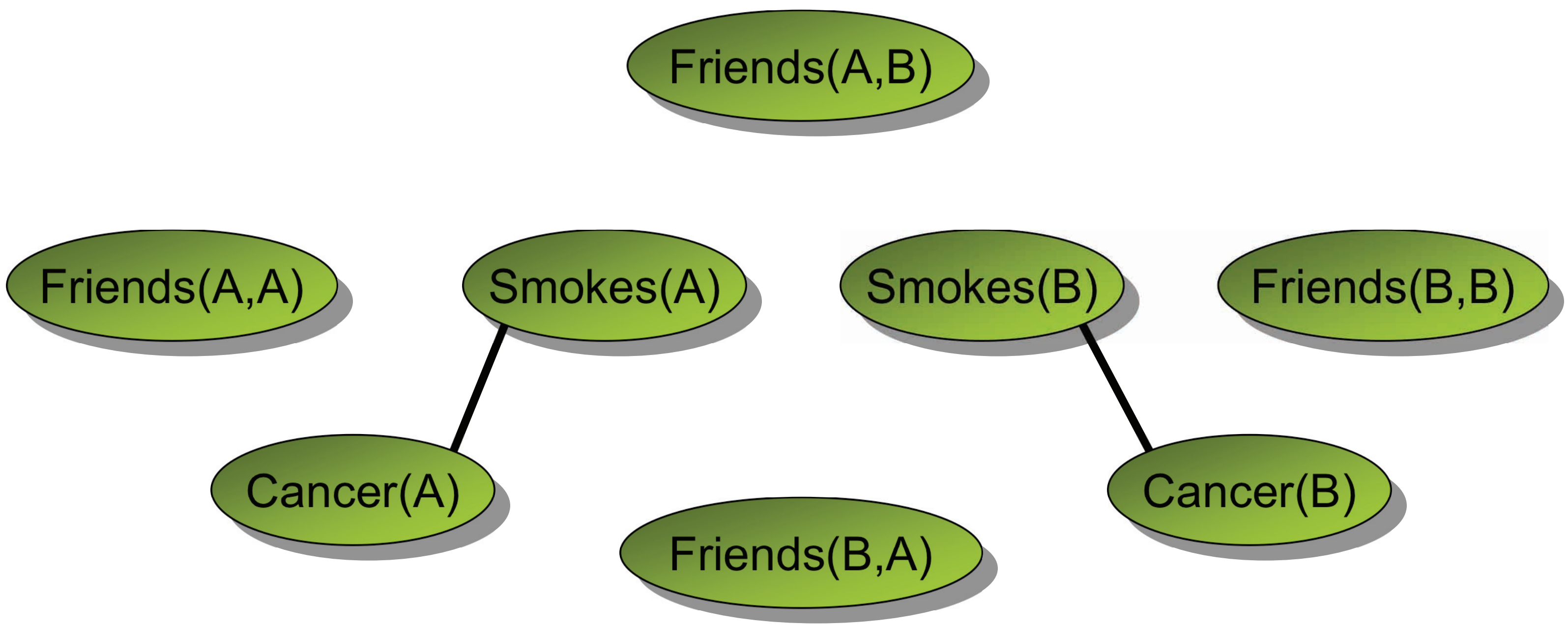


slides by Pedro Domingos

Markov Logic Networks

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

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

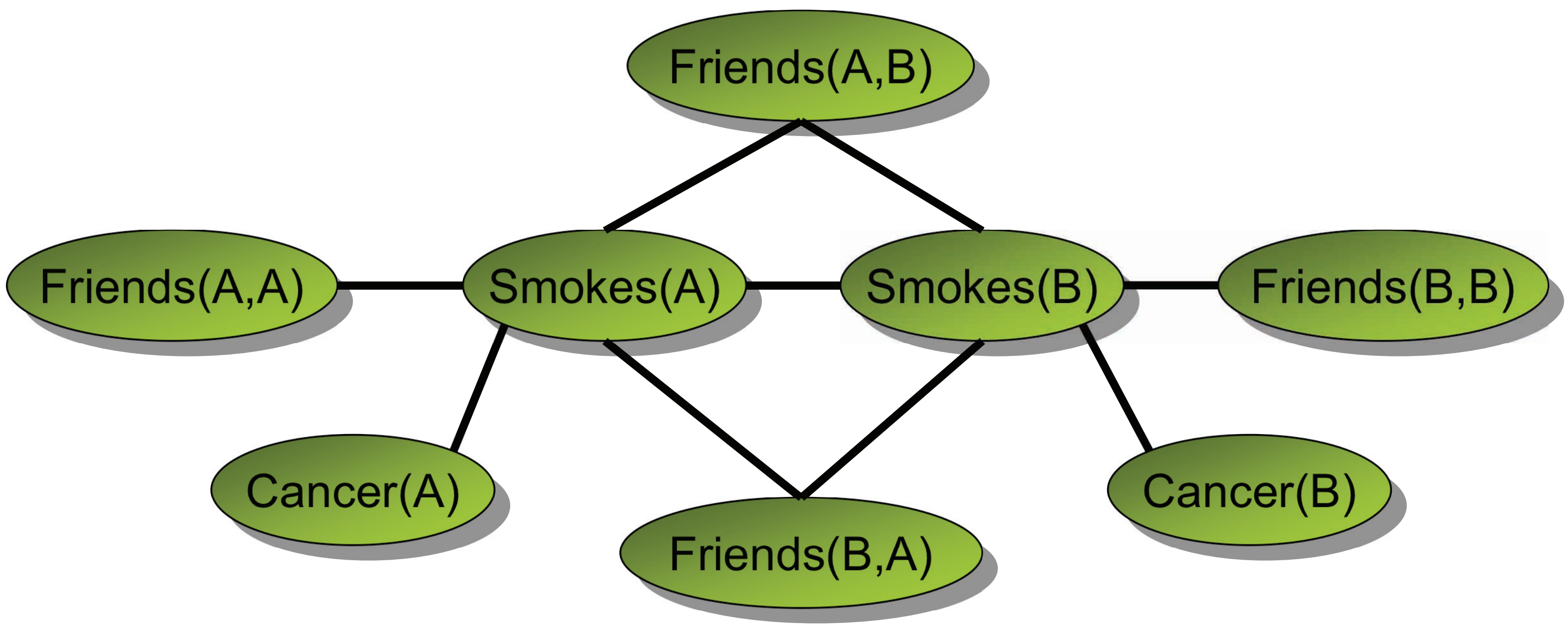


slides by Pedro Domingos

Markov Logic Networks

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

Learning Undirected Probabilistic Relational Models

- Parameter estimation
  - –discriminative (gradient, max-margin)
  - –generative setting using pseudo-likelihood
- Structure learning
  - –Similar to (Probabilistic) Inductive Logic Programming

Applications

- Computer Vision (Taskar et al.)
  - Collective classification of 3D scan data
- Citation Analysis (Taskar et al., Singla&Domingos)
- Activity Recognition (Liao et al.)

Conclusions Learning from Interpretations

- Data cases are herbrand interpretations

- Most prob. ILP approaches incorporates objects and relations among the objects into Bayesian and Markov networks
- Learning includes principles from
  - Inductive logic programming / multi-relational data mining
    - Refinement operators, Background knowledge, Bias
  - Statistical learning
    - Likelihood, Independencies, Priors

Outline

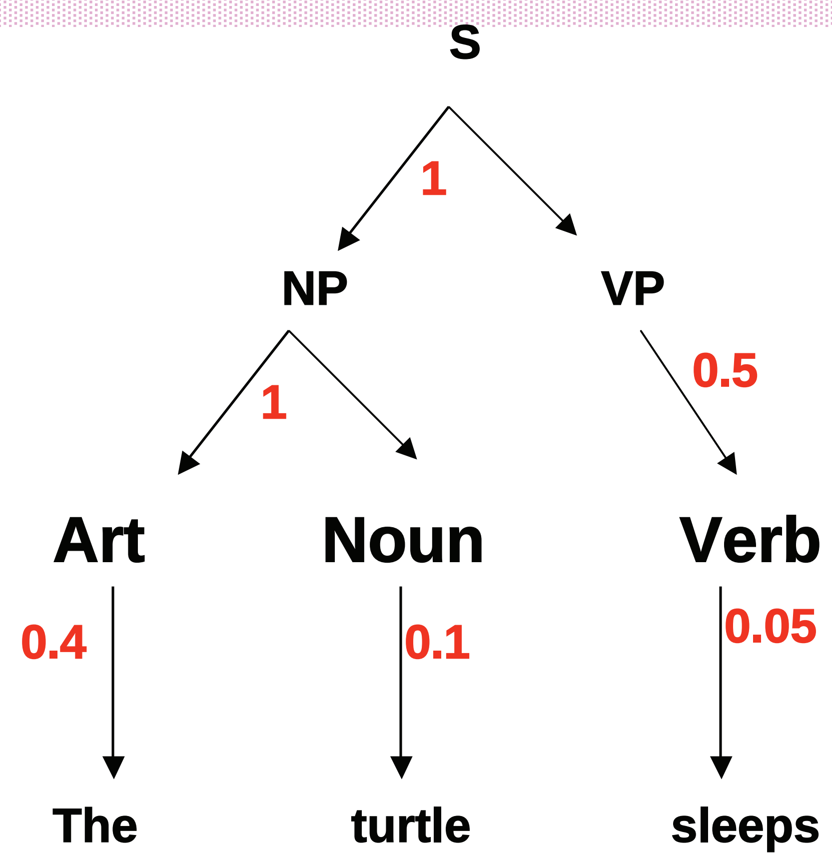
1. Motivation / Introduction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. Discriminative ILP
6. Conclusions

Learning from entailment and from proofs

- Stochastic Logic Programs
  - Derived from Probabilistic Context Free Grammars by Eisele and Muggleton
  - Closely related to Sato’s PRISM and Poole’s ICL
- Learning from entailment
  - Parameter estimation (Cussens’ FAM)
- Learning from proofs
  - Structure learning

Probabilistic Context Free Grammars

1.0 : S -> NP, VP
1.0 : NP -> Art, Noun
0.6 : Art -> a
0.4 : Art -> the
0.1 : Noun -> turtle
0.1 : Noun -> turtles
…
0.5 : VP -> Verb
0.5 : VP -> Verb, NP
0.05 : Verb -> sleep
0.05 : Verb -> sleeps
….



P(parse tree) = 1x1x.5x.1x.4x.05

PCFGs

$$P(\text{ parse tree }) = \prod_i p_i^{c_i}$$

where p_i is the label of rule i

and c_i the number of times it was applied

$$P(\text{ sentence }) = \sum_{i \text{ is a parse tree for sentence}} P(tree_i)$$

Observe: all derivation/rewriting steps succeed

i.e. S->T,Q

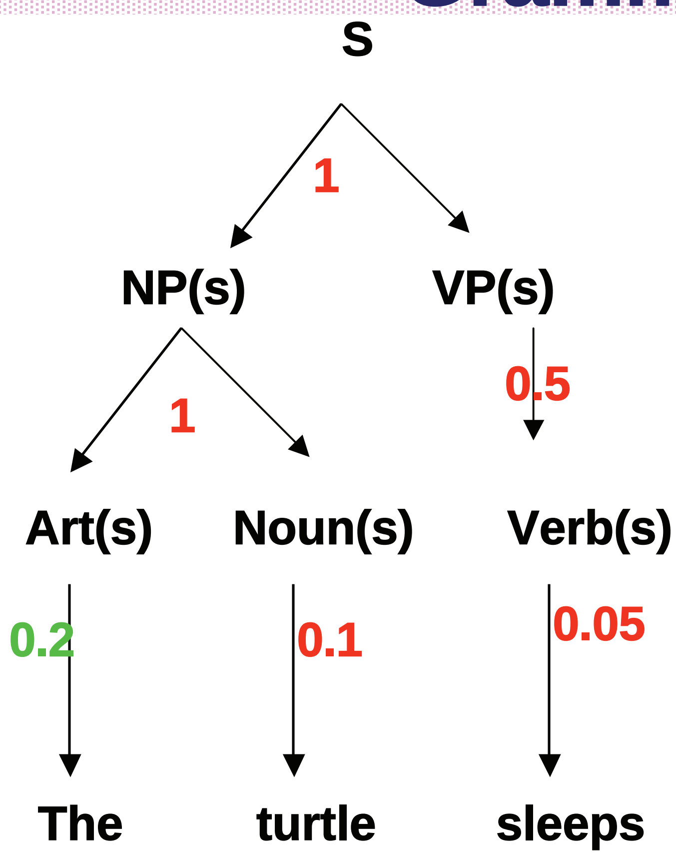
T->R,U

always gives

S-> R,U,Q

Probabilistic Definite Clause Grammar

1.0 : S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
…
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleep
0.05 Verb(plur) -> sleeps
….

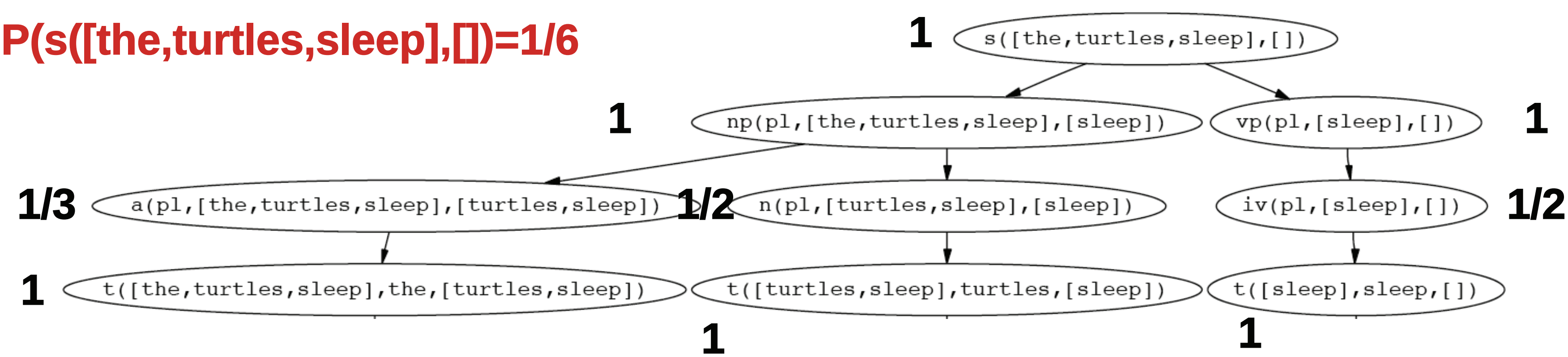


P(derivation tree) = 1x1x.5x.1x .2 x.05

In SLP notation

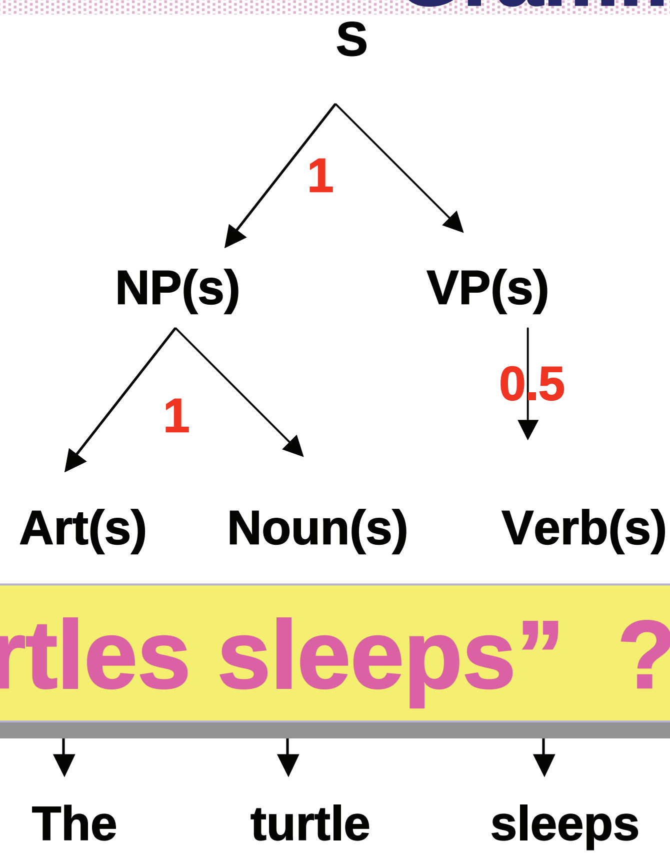
```
    sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
1   noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
    verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
    article(singular, A, B) :- terminal(A, a, B).
1/3 article(singular, A, B) :- terminal(A, the, B).
    article(plural, A, B)   :- terminal(A, the, B).
1/2 noun(singular, A, B) :- terminal(A, turtle, B).
    noun(plural, A, B)   :- terminal(A, turtles, B).
    intransitive_verb(singular, A, B) :- terminal(A, sleeps, B).
    intransitive_verb(plural, A, B) :- terminal(A, sleep, B).
1   terminal([A|B],A,B).
```

P(s([the,turtles,sleep],[])=1/6



Probabilistic Definite Clause Grammar

1.0 : S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(sing) -> the
0.1 Noun(sing) -> turtle
0.1 No…
…
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleep
0.05 Verb(plur) -> sleeps
….



P(derivation tree) = 1x1x.5x.1x .2 x.05

What about "A turtles sleeps" ?

SLPs

$P_D(\text{ derivation for goal } g(X_1,\ldots,X_n)) = \prod_i p_i^{c_i}$

Observe: some derivations/resolution steps fail

e.g. NP(*Num*)–>Art(*Num*), Noun(*Num*)

and Art(sing)-> a and Noun(plur)->turtles

np(Num,S1,S2):- art(Num,S1,S3),noun(Num,S3,S2)

and art(sing,[a|S],S) and noun(plur,[turtles|S],S)

Interest in successful derivations/proofs/refutations

-> normalization necessary

$$P_S(\text{ proof }) = \frac{P_D(\text{ proof })}{\sum_i P_D(\text{ proof}_i\text{ })}$$

$$P_A(\text{ ground atom } g(X_1,\ldots,X_n)\theta\text{ }) = \sum_{i \text{ is a proof tree for } g(X_1,\ldots,X_n)\theta} P_s(tree_i)$$

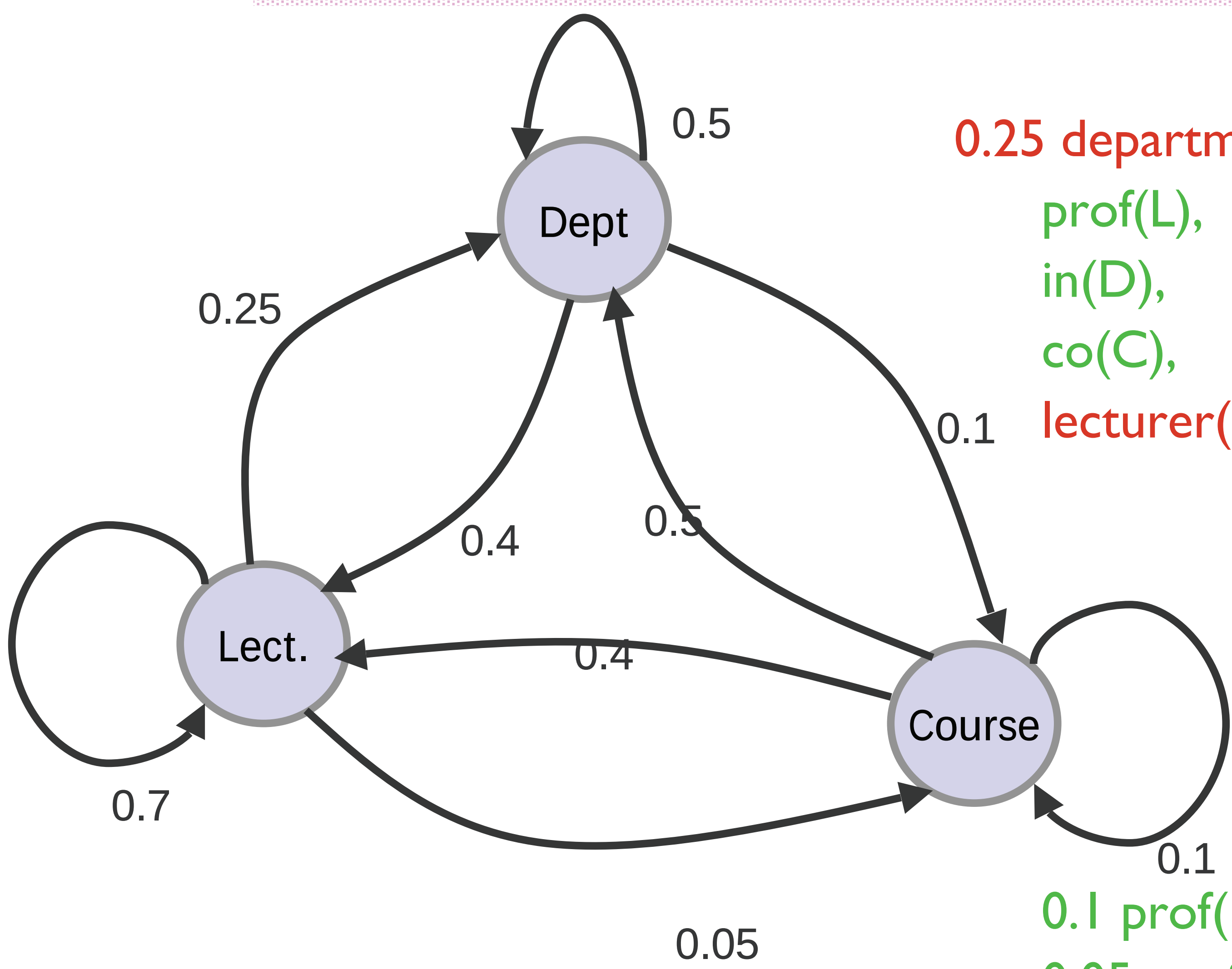
Example Application

- Consider traversing a university website
- Pages are characterized by predicates
  department(cs,nebel)   denotes the page of cs following the link to nebel
- Rules applied would be of the form

```
department(cs,nebel) :-
        prof(nebel), in(cs), co(ai), lecturer(nebel,ai).
pagetype1(t1,t2) :-
        type1(t1), type2(t2), type3(t3), pagetype2(t2,t3)
```

- SLP models probabilities over traces / proofs / web logs
  department(cs,nebel), lecturer(nebel,ai007), course(ai007,burgard), …
- This is actually a Logical Markov Model
  – Logical Hidden Markov Model (cf. Kersting et al. JAIR)
  – Includes also structured observations and abstraction

Logical Markov Model



0.25 department(D,L) :-
prof(L),
in(D),
co(C),
lecturer(L,C).

0.1 prof(nebel).
0.05 prof(burgard).

PRISM (Sato) / ICL (Poole)

- A logic program in which probability labels are attached to facts;
- Clauses carry no probability label (or equiv. P =1)

$$disjoint(h_1 : p_1; ..., h_n : p_n)$$
$$\text{statements, facts } h_i; \sum_i p_i = 1$$

  - Disjoint(head(C) : 0.5; tail(C): 0.5)
- Probability distributions can be defined in a related/similar fashion on proofs
  - on explanations
  - on atoms
  - though some differences with SLP

PRISM (Sato) / ICL (Poole)

Logical Abduction

mortal(X) :- human(X)   and mortal(X) :- animal(X)

From mortal(socrates) infer human(socrates)

OR

infer animal(socrates)

Probabilistic Abduction (ICL Poole)

From moral(socrates) infer

probabilities for human(socrates)

and animal(socrates)

Backword reasoning

ICL / PRISM example

```
btype('A') : - (gtype(a,a); gtype(a,o); gtype(o,a)).
btype('B') : - (gtype(b,b); gtype(b,o); gtype(o,b)).
btype('O') : - gtype(o,o).
btype('AB') : - (gtype(a,b); gtype(b, a)).
gtype(X,Y) :- gene(father,X),   gene(mother,Y).
gene(P,G) :- msw(gene,P,G).          probabilistic switch

disjoint(p1: msw(gene,P,o),
         p2: msw(gene,P,a),
         p3: msw(gene,P,b))          outcomes +  prob switch
```

Infer e.g. P(btype('A')) from probabilities of proofs - facts

Infer e.g. explanations and prob.

P(gene(father,a), gene(mother,a)) = p2 * p2

…

P(gene(father,a), gene(mother,a) | btype('A')) : normalize

Parameter Estimation for CFGs

- Given
  - A set of sentences for the startsymbol
    - E.g. the, turtles, sleep
  - The structure of the CFG
    - (i.e. the rules - not the probability labels)
- Find
  - The maximum likelihood parameters of the CFG
- Approach
  - Inside - Outside Algorithm
    - Parse trees are unobserved
    - Instance of EM
    - dynamic programming (using CYK)

Parameter Estimation for SLPs

- Given
  - A set of ground facts for a predicate *g*
    - E.g. s([the, turtles, sleep],[])
  - The structure of the stochastic logic program
    - (i.e. the logic part - not the probability labels)
- Find
  - The maximum likelihood parameters of the SLP
- Approach
  - Failure Adjusted Maximisation Algorithm (Cussens)
    - Parse trees and failures are unobserved
    - Instance of EM
    - Ako dynamic programming using tabling (Sato)
    - Very efficient

Structure Learning

- From entailment : Muggleton ILP 02
  - Learns a single clause at a time from facts only
  - Hard problem, requires one to solve the full inductive logic programming problem
- From proof trees : De Raedt et al AAAI 05
  - Learn from proof-trees instead of from ground facts
  - Proof-trees carry much more information
  - Upgrade idea of tree bank grammars

- Given
  - A set of proof trees
- Find
  - An SLP that maximizes the likelihood

Initial Rule Set DCG

S -> NP(s), VP(s)

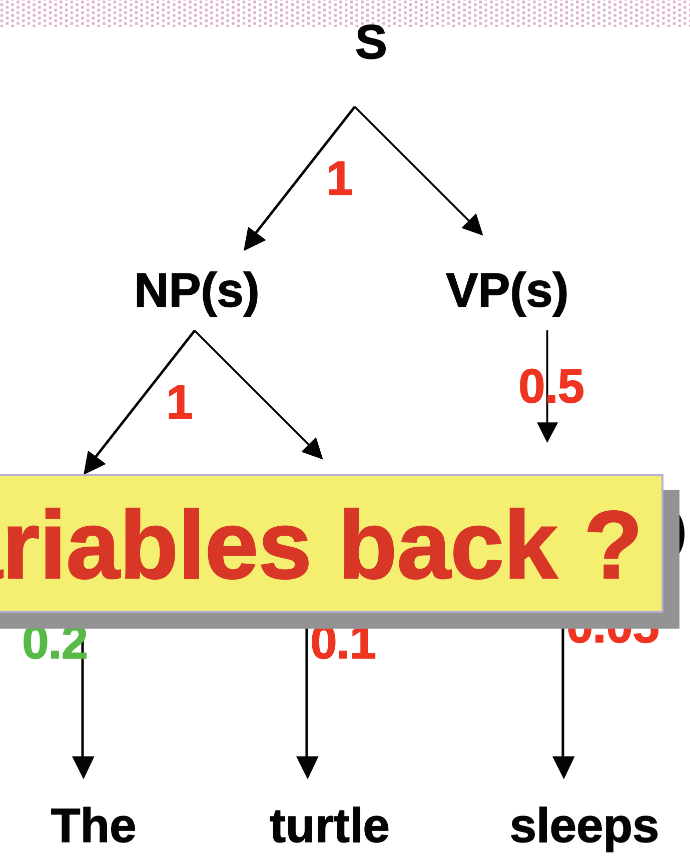
NP(s) -> Art(s), Noun(s)

VP(

Art(s) -> the

Noun(s) -> turtle

Verb(s) -> sleeps



P(derivation tree) = 1x1x.5x.1x.4x.05

How to get the variables back ?

Learning SLPs from Proof Trees

- Based on Tree-Bank Grammar idea, e.g. Penn Tree Bank

- Key algorithm
  - Let S be the set of all (instantiated) rules that occur in an example proof tree
  - Initialize parameters
  - repeat as long as the score of S improves
    - Generalize S
    - Estimate the parameters of S using Cussens' FAM
      - (which can be simplified - proofs are now observed)
  - Output S

Generalizing Rules in SLPs

- Generalization in ILP
  - Take two clauses for same predicate and replace them by the lgg under θ - subsumption (Plotkin)
  - Example

    department(cs,nebel) :-
    prof(nebel), in(cs), course(ai), lect(nebel,ai).
    department(cs,burgard) :-
    prof(burgard), in(cs),course(ai), lect(burgard,ai)
  - Induce

    department(cs,P) :-
    prof(P), in(cs),course(ai), lect(P,ai)

Strong logical constraints

- Replacing the rules r1 and r2 by the lgg should preserve the proofs !
- So, two rules r1 and r2 should only be generalized when
  – There is a one to one mapping (with corresponding substitutions) between literals in r1, r2 and lgg(r1,r2)
- Exclude

  father(j,a) :- m(j),f(a),parent(j,a)

  father(j,t) :- m(j),m(t), parent(j,t)
- Gives

  father(j,P) :- m(j),m(X),parent(j,P)

Web Log Data

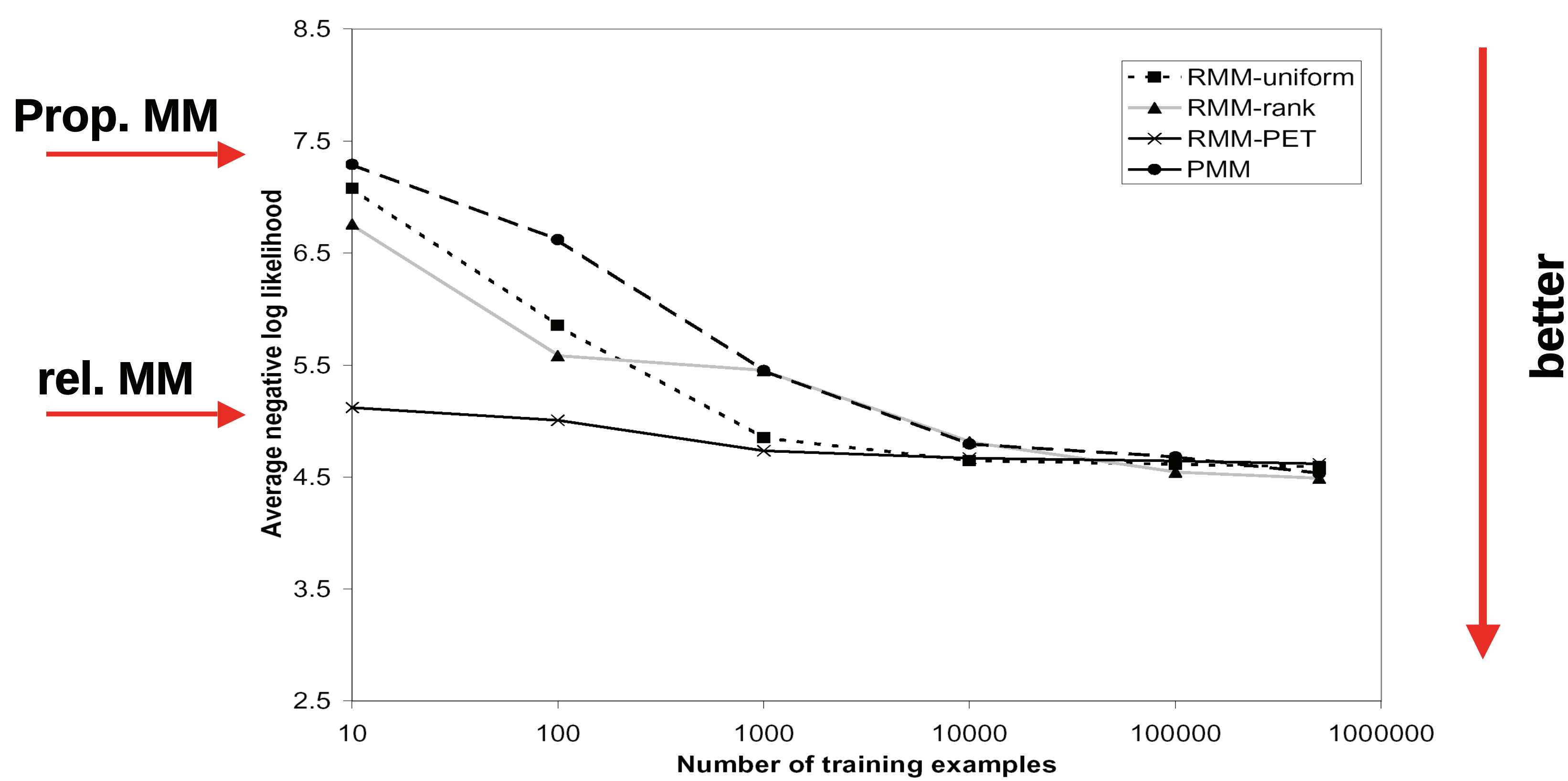
[Anderson et al.]

- Log data of web sides
- KDDCup 200 (www.gazelle.com)
- RMM over
  - Home()
  - Boutique()
  - Departments()
  - Legcare_vendor()
  - Lifestyles()
  - Vendor()
  - AssortmentDefault()
  - Assortment(Assortment)
  - ProductDetailLegcareDefault()
  - ProductDetailLegcare(Product)
  - ProductDetailLegwearDefault()
  - ProductDetailLegwearProduct(Product)
  - ProductDetailLegwearAssortment(Assortment)
  - ProductDetailLegwearProdCollect(Product, Collection)
  - ProductDetailLegwearProdAssort(Product, Assortment)
  - ProductDetailLegwear(Product, Collection, Assortment)

User Log Data

[Anderson et al.]

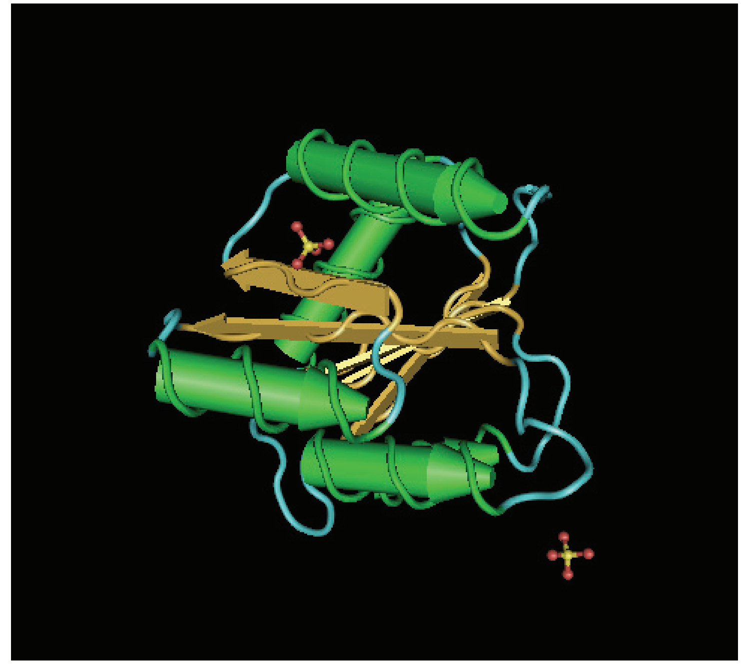
Relational representation pays off



Protein Fold Recognition

[Kersting et al.; Kersting, Gaertner]

- Comparison of protein structure is fundamental to biology, e.g. function prediction
- Two proteins show sufficient sequence similarity = essentially adopt the same structure.



- If one of the two similar proteins has a known structure, can build a rough model of the protein of unknown structure.

Protein Secondary Structure

[Kersting et al.; Kersting, Gaertner]

helix

type of helix

quantized number of acids

```
[helix(h(right,3to10),5),
 helix(h(right,alpha),13),
 strand(null,7),
 strand(minus,7),
 strand(minus,5),
 helix(h(right,3to10),5),…]
```

length

orientation

strand

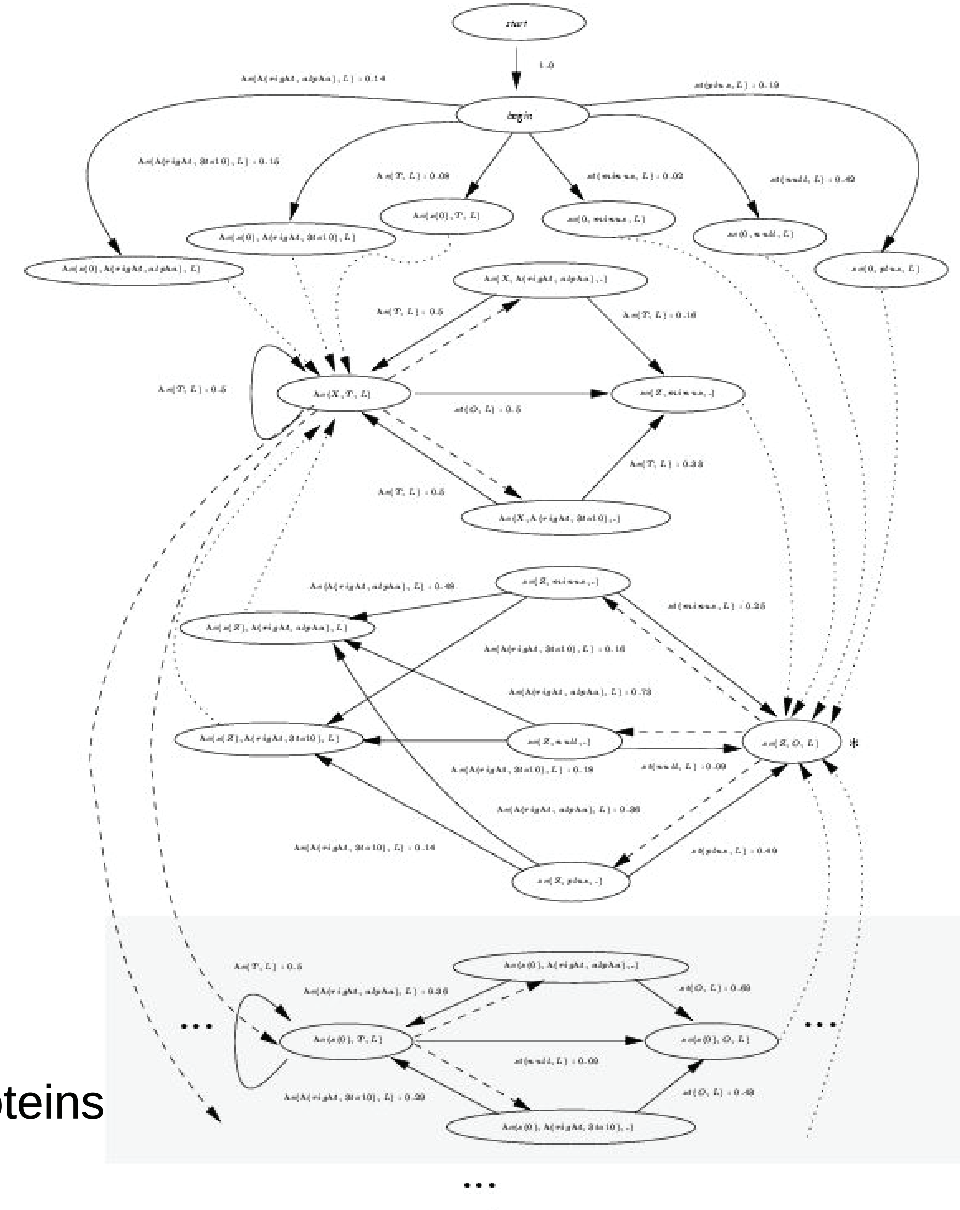
Model

[Kersting et al.]

~120 parameters

vs.

over 62000 parameters



Secondary structure of domains of proteins
(from PDB and SCOP)

fold1: TIM beta/alpha barrel fold, fold2: NAD(P)-binding Rossman-fold fold23: Ribosomal protein L4, fold37: glucosamine 6-phosphate deaminase/isomerase old fold55: leucine aminopeptidas fold. 3187 logical sequences (> 30000 ground atoms)

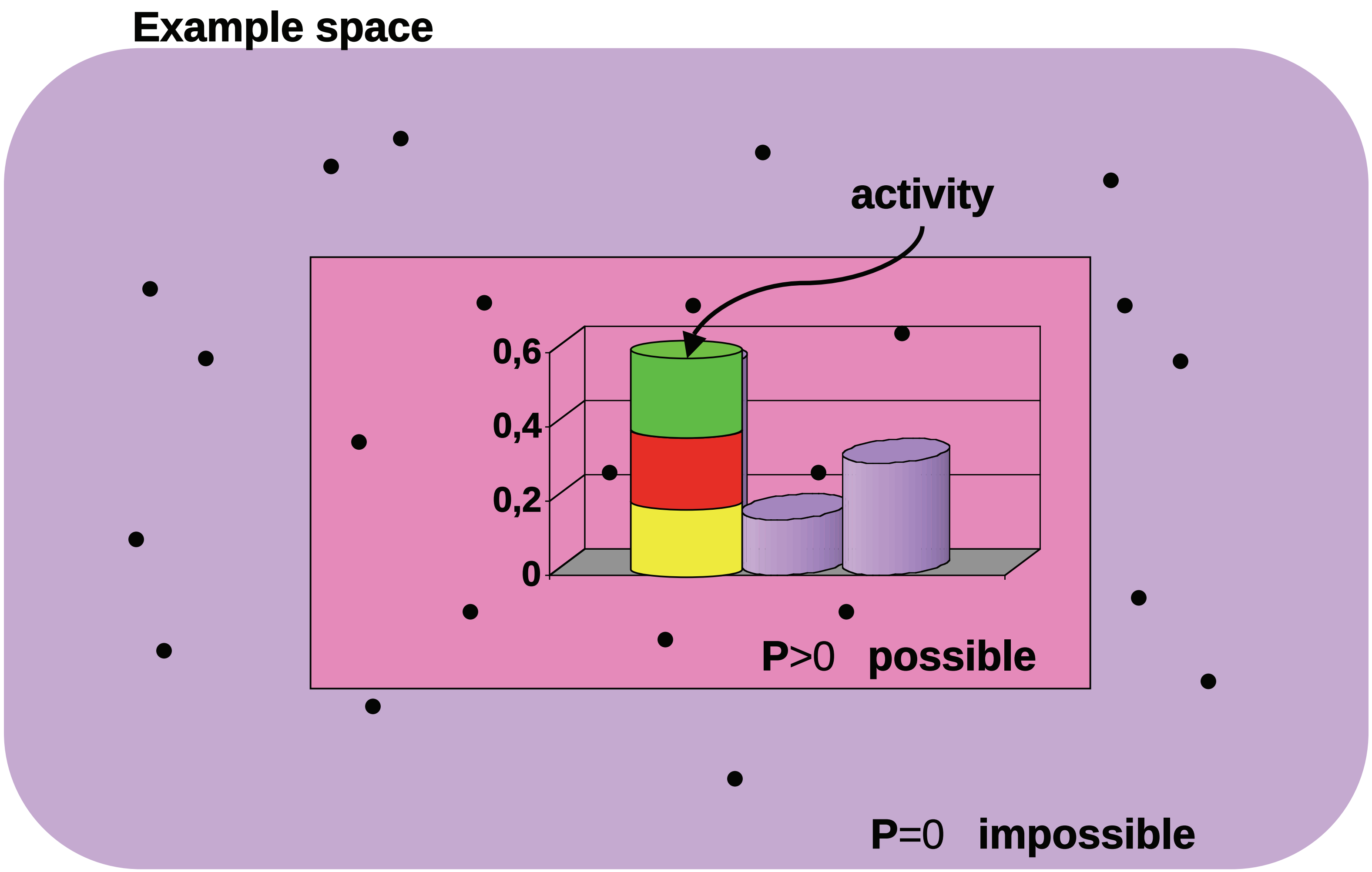
Conclusions Learning from Entailment and Proofs

- SLPs extend PCFGs as a representation
- Proof-trees for SLPs correspond to parse-trees in PCFGs
- Upgrading the learning from tree-banks setting for use in SLPs
- Learning from proof trees is a new setting for inductive logic programming/statistical relational learning
  - Generalizes learning from traces
- Strong logical constraints at structure level
- Allows one also to elegantly model and study RMMs and LOHMMs
  - Sequential relational / logical traces.
- A lot of further research questions
  - Most of all : experiments on real data

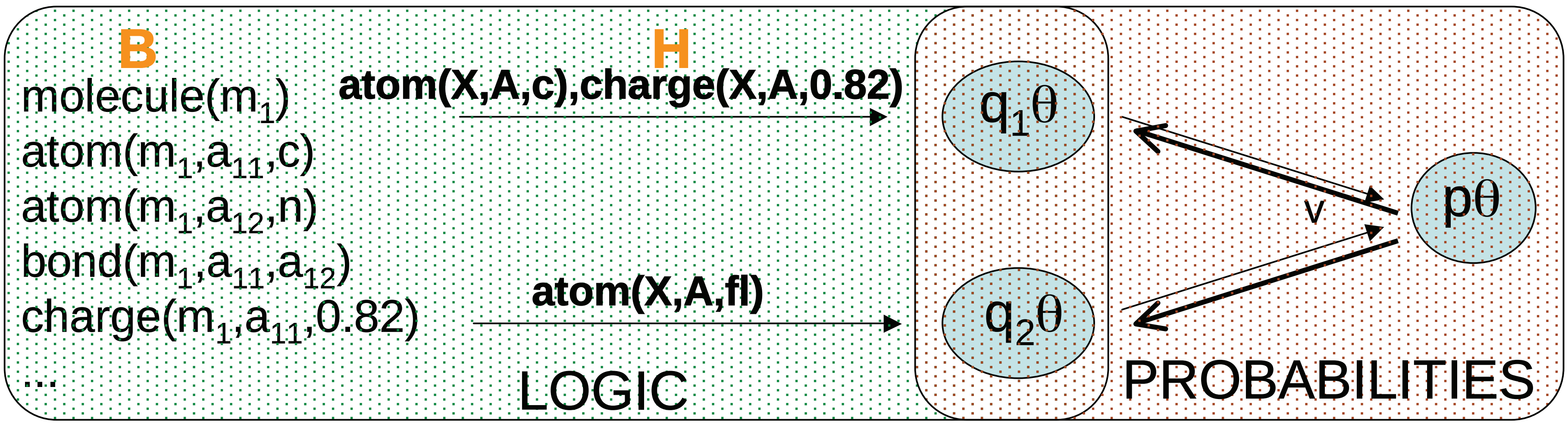
Outline

1. Motivation / Introduction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. **Discriminative ILP**
6. Conclusions

Probabilistic ILP Problem



nFOIL = naive Bayes + FOIL



- Clause set + simple probabilistic model
- Idea: Clauses are independent
- Success/failure of a query is random variable in a Naive Bayes model

The nFOIL model

- Naive Bayes assumption translates into

$$P(p\theta \mid H,B) = P(p\theta \mid q_1\theta,\ldots,q_k\theta)$$

$$= \frac{P(q_1\theta,\ldots,q_k\theta \mid p\theta) * P(p\theta)}{P(q_1\theta,\ldots,q_k\theta)}$$

$$= \frac{\prod_i P(q_i\theta \mid p\theta) * P(p\theta)}{P(q_1\theta,\ldots,q_k\theta)}$$

- Model consists of clauses $q_1,\ldots,q_k$ and parameters $P(q_i\theta \mid p\theta)$, $P(p\theta)$
- Classify positive if $P(p\theta \mid H,B) > 0.5$

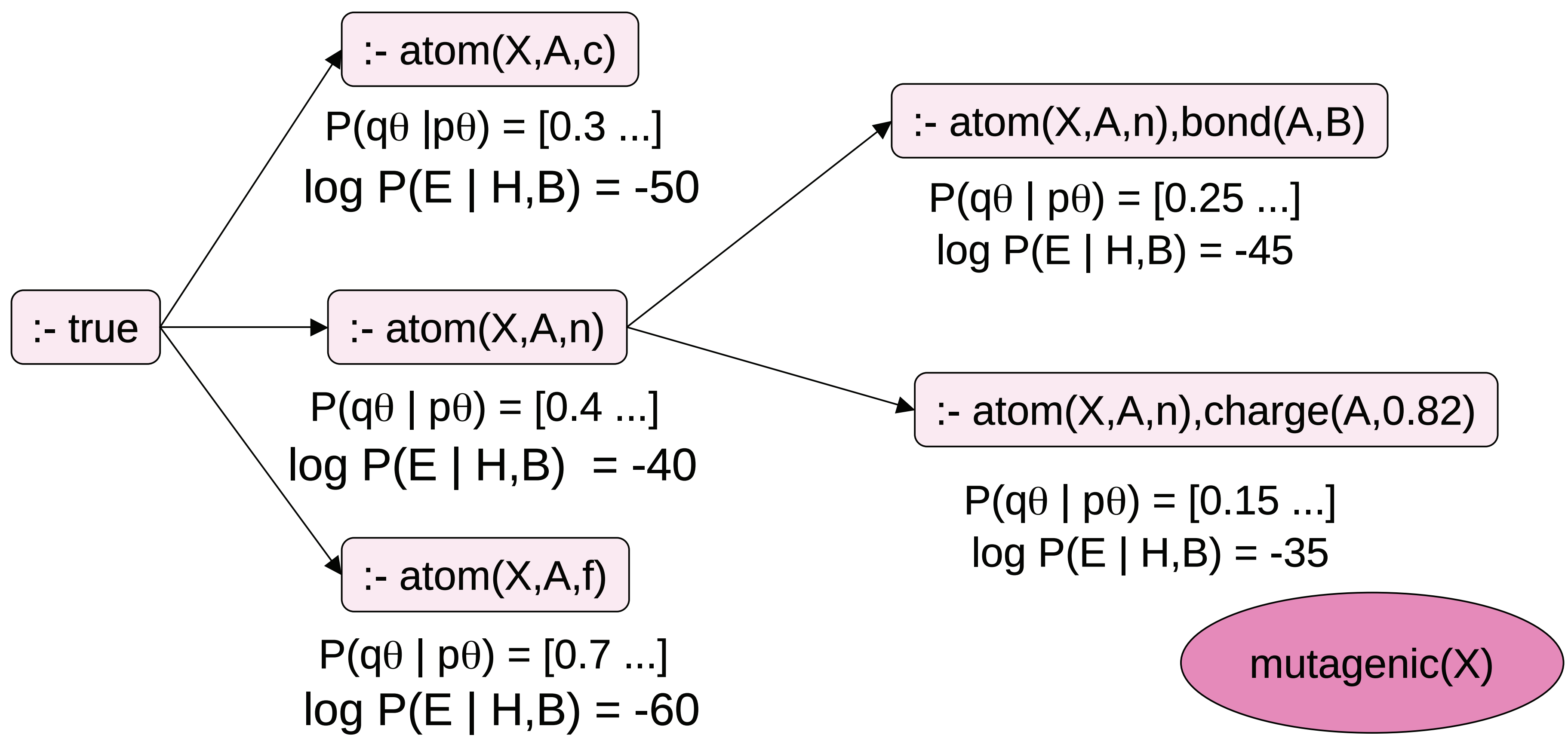
Learning: nFOIL

- Modified FOIL: search guided by cond. likelihood
- FOIL
  - a clause is scored by its coverage
  - Covered positive examples are removed
- nFOIL
  - score **a set of clauses** $\{q_1,\ldots,q_k\}$ by conditional likelihood:

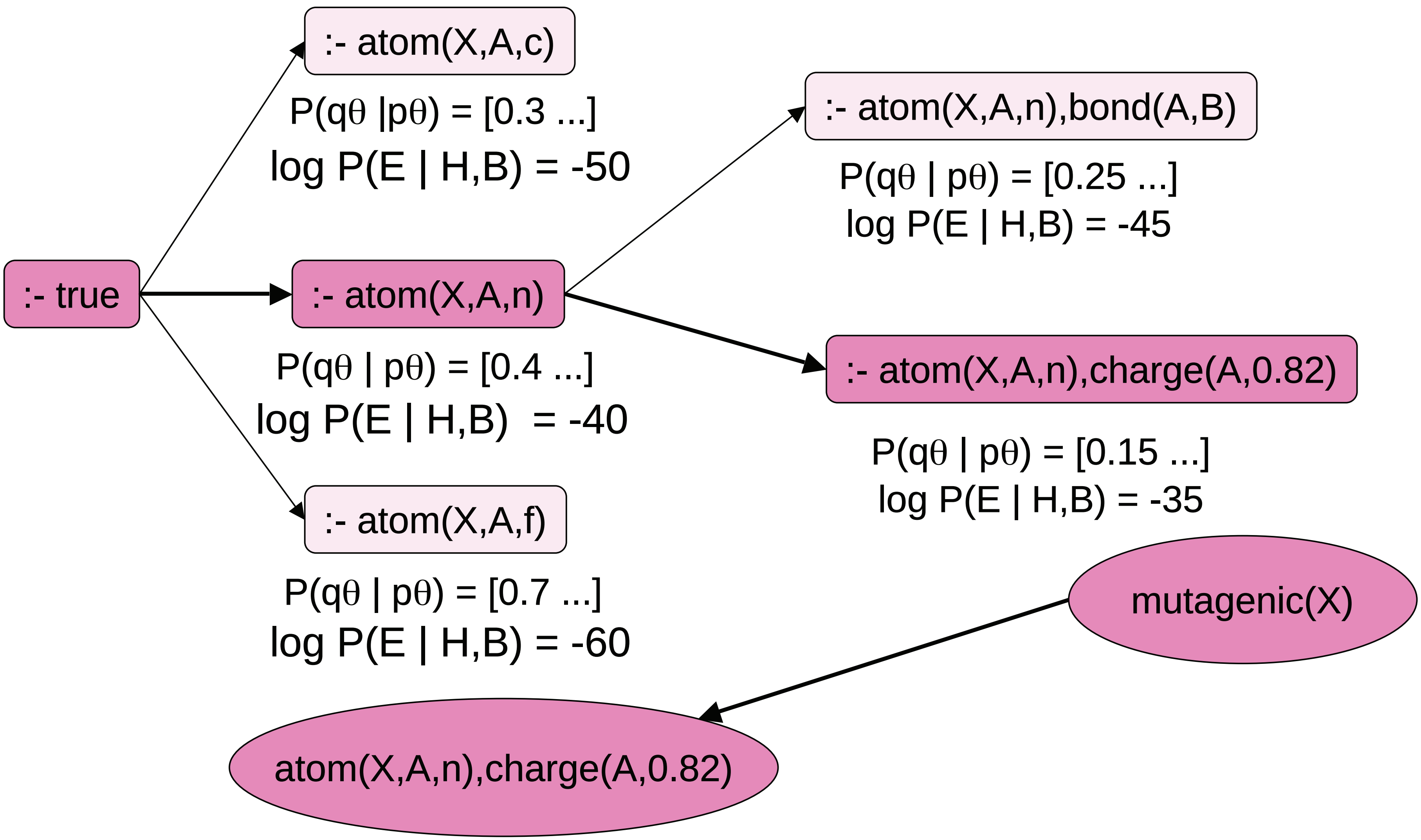
$$P(E \mid H,B) = \prod_{e \text{ in } E} \frac{\prod_i P(q_i\theta \mid p\theta) * P(p\theta)}{P(q_1\theta,\ldots,q_k\theta)}$$

where $$P(q_i\theta \mid p\theta) = \frac{\text{count}(q_i\theta, p\theta)}{\text{count}(p\theta)}$$

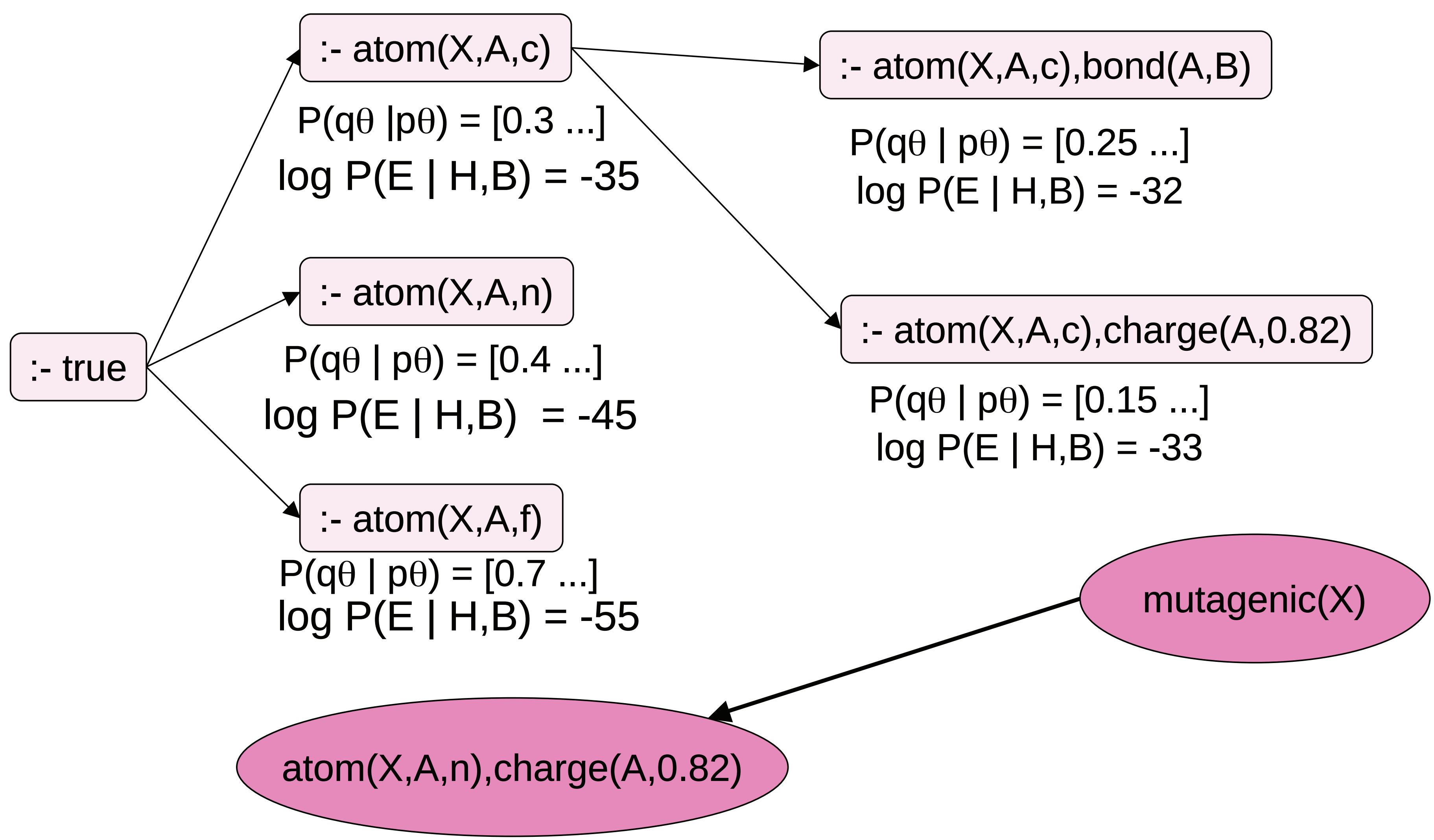
Learning (Example)



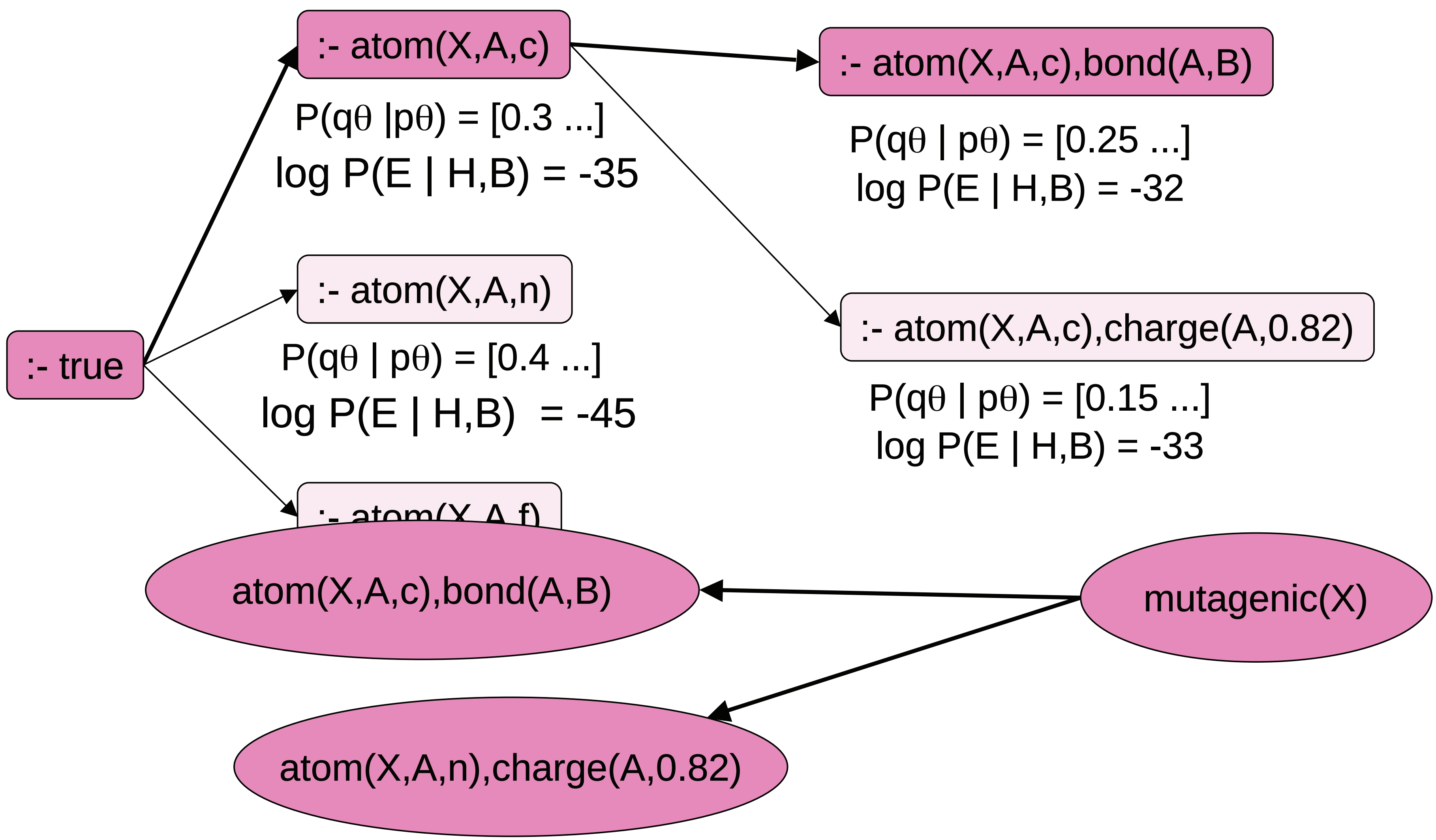
Learning (Example)



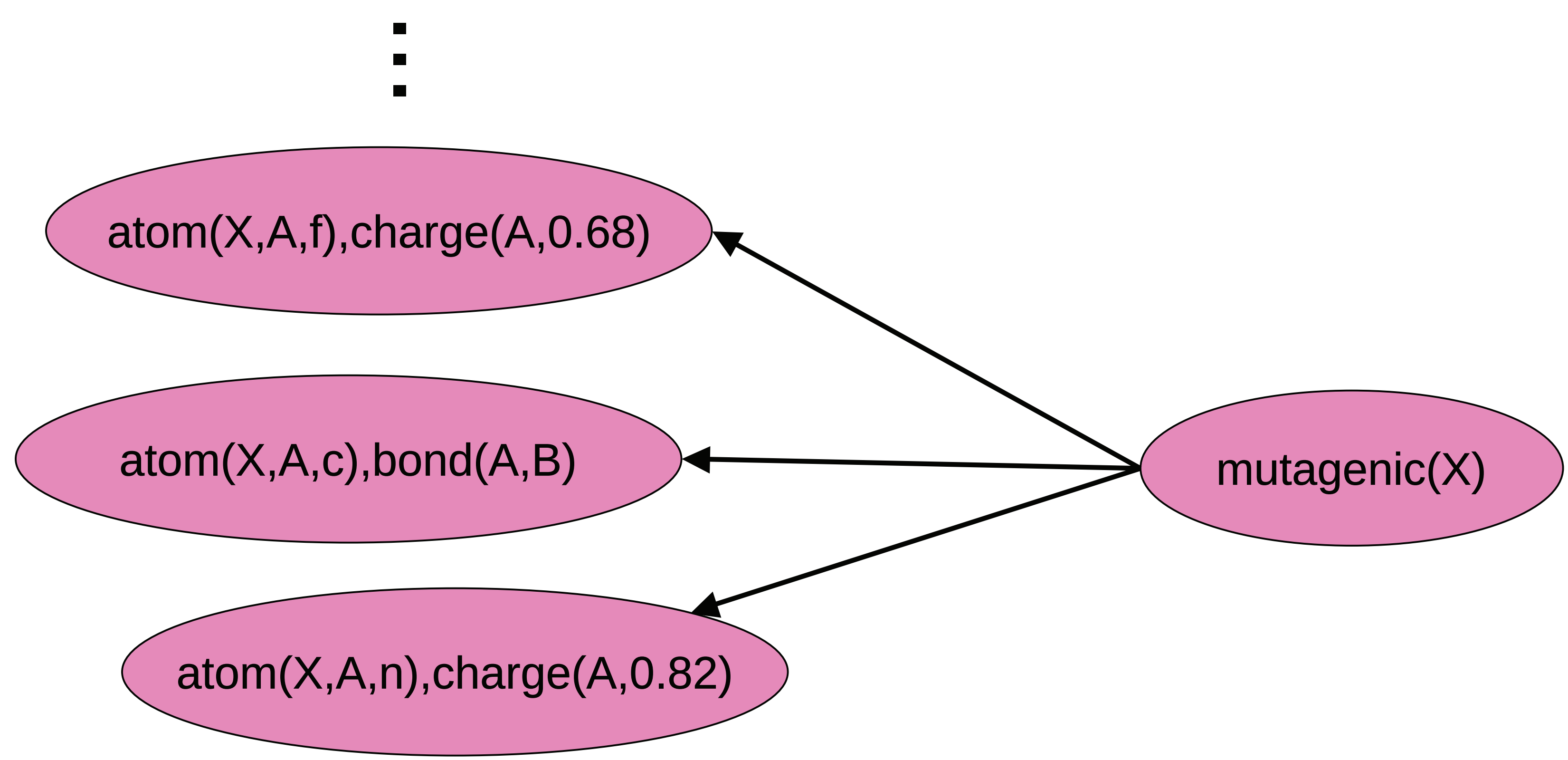
Learning (Example)



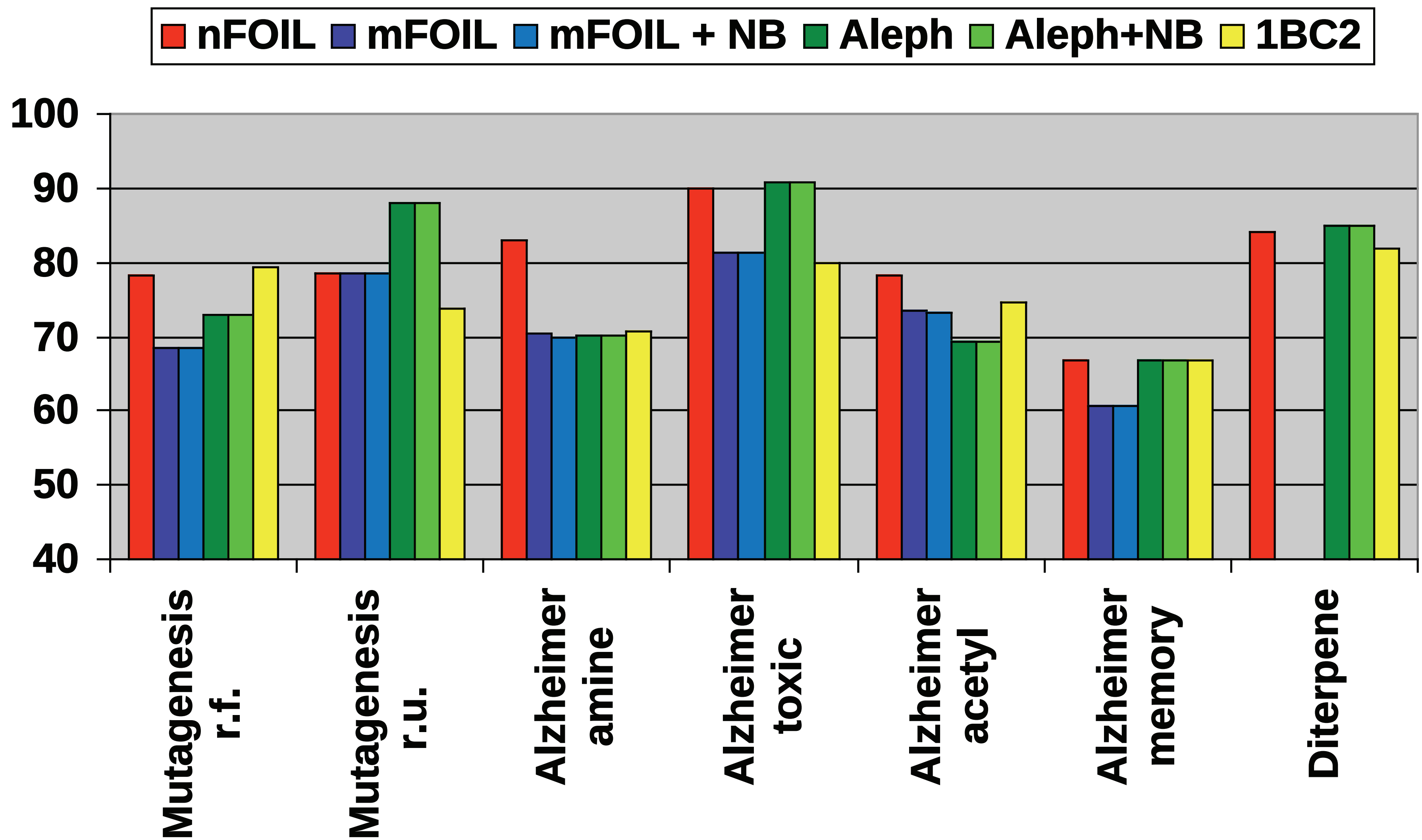
Learning (Example)



Learning (Example)



Experimental Results



Fisher Kernels
[Jaakkola, Haussler NIPS'99]

- Kernels based on probability models
- Idea: capture the change in parameters to accommodate new data point

Fisher score of observed data x

$$U_x = \nabla_\theta \log P(x \mid \theta^*, M) = \left(\frac{\partial \log P(x \mid \theta^*, M)}{\partial \theta_1}, \ldots, \frac{\partial \log P(x \mid \theta^*, M)}{\partial \theta_n} \right)^\top$$

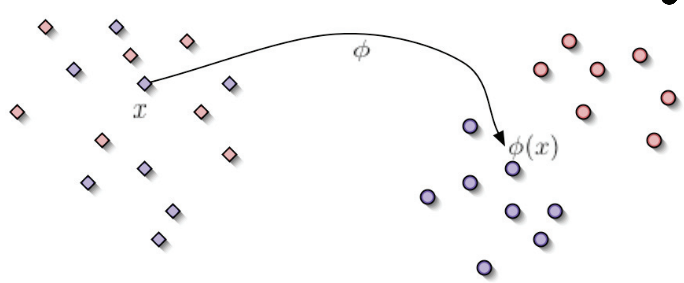
Fisher information matrix $J_\theta = E_x[U_x U_x^T]$

Fisher kernel $k(x, x') = U_x^T J_{\theta^*}^{-1} U_{x'}$

In practice: $k(x, x') = U_x^T U_{x'}$

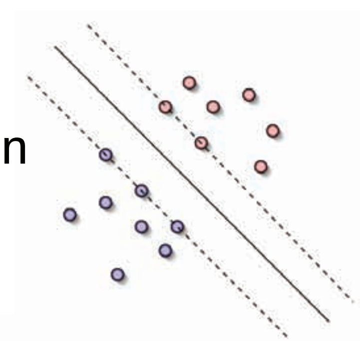
SVMs in a Nutshell
[Boser,Guyon,Vapnik COLT'92]

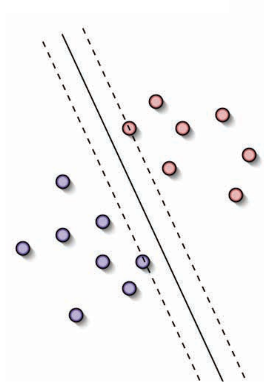
$$x \rightarrow \phi(x)$$



- With an appropriate mapping ϕ to a sufficiently high dimension, data from 2 classes can always be separated by a hyperplane
- Find optimal separating hyperplane by maximizing the margin of separation
- Computationally effective algorithm exist to solve this optimization problem (in higher dimensional feature space).

Larger Margin



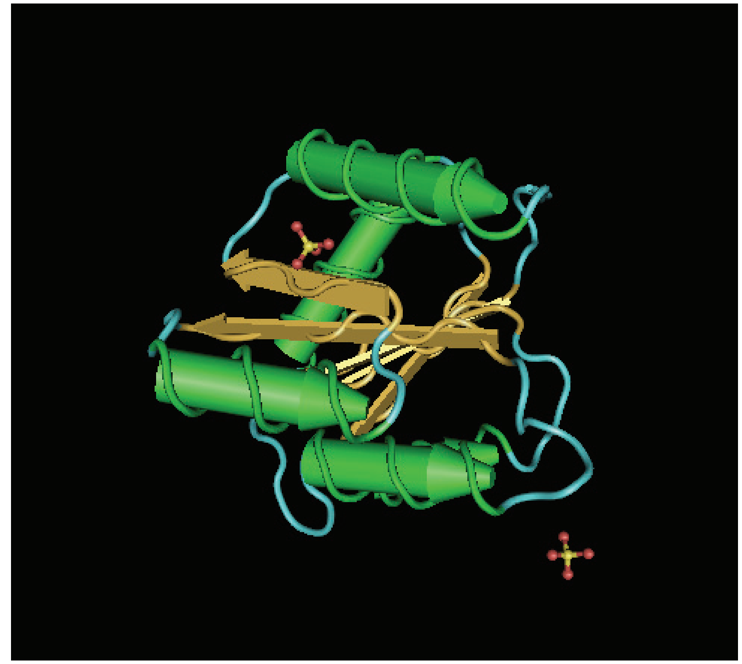


Smaller Margin

Protein Fold Recognition

[Kersting et al.; Kersting, Gaertner]

- Comparison of protein structure is fundamental to biology, e.g. function prediction
- Two proteins show sufficient sequence similarity = essentially adopt the same structure.



- If one of the two similar proteins has a known structure, can build a rough model of the protein of unknown structure.

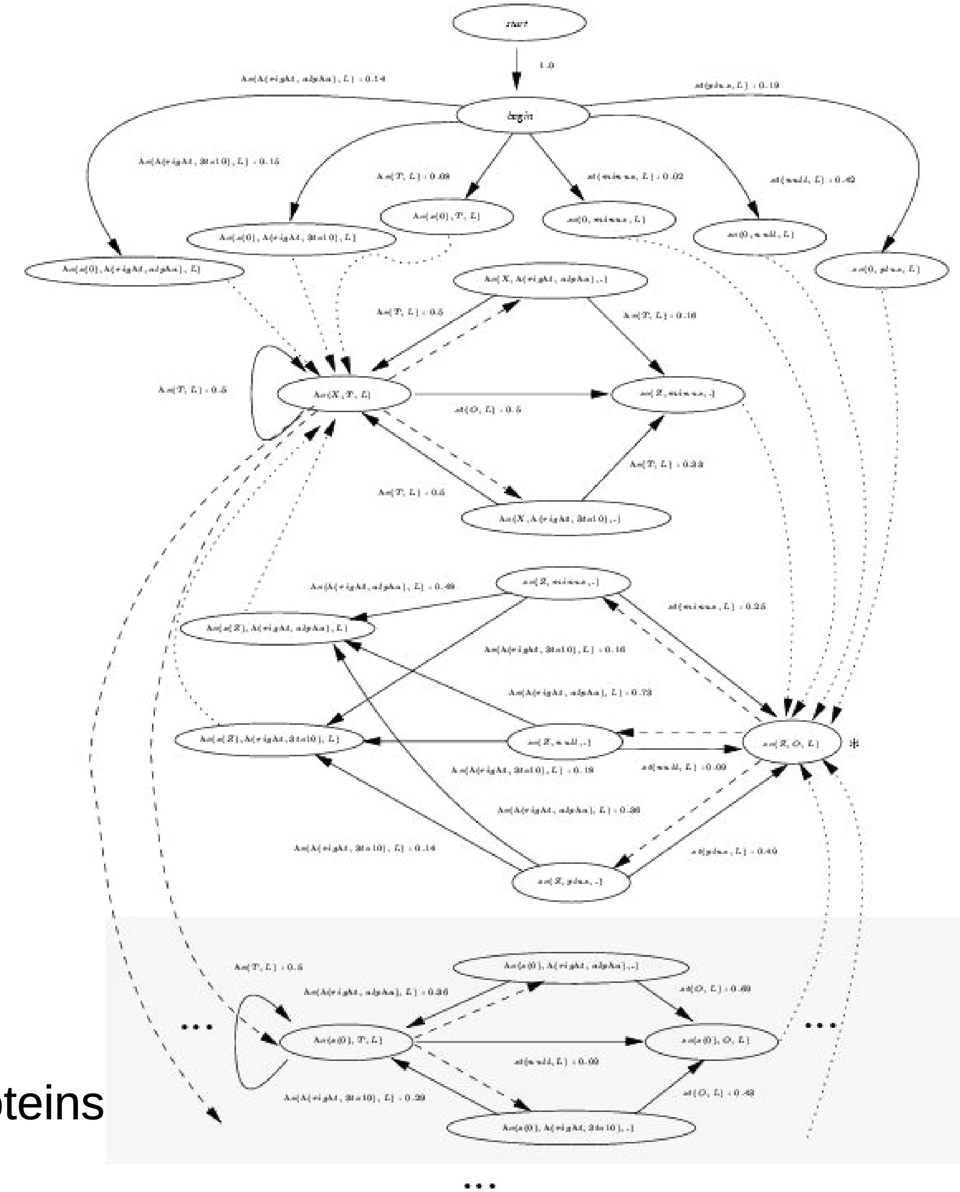
Model

[Kersting et al.]

~120 parameters

vs.

over 62000 parameters



Secondary structure of domains of proteins (from PDB and SCOP)

fold1: TIM beta/alpha barrel fold, fold2: NAD(P)-binding Rossman-fold fold23: Ribosomal protein L4, fold37: glucosamine 6-phosphate deaminase/isomerase old fold55: leucine aminopeptidas fold. 3187 logical sequences (> 30000 ground atoms)

Results

[Kersting et al.; Kersting, Gaertner]

- Accuracy: 74% vs. 82.7% (1622 vs. 1809 / 2187)
- Majority vote: 43%

	fold1	fold2	fold23	fold37	fold55
precision	0.86 / 0.89	0.69 / 0.86	0.56 / 0.82	0.72 / 0.70	0.66 / 0.74
recall	0.78 / 0.87	0.67 / 0.81	0.71 / 0.85	0.66 / 0.72	0.96 / 0.86

New class of probabilistic relational Kernels

mRNA

[Kersting et al.; Kersting, Gaertner]

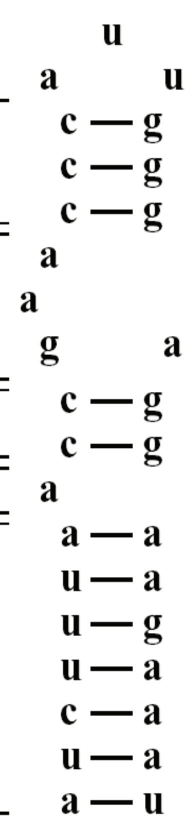
- Identifying subsequences in mRNA that are responsible for biological functions.

- Secondary structures of mRNAs form tree structures: not easily for HMMs

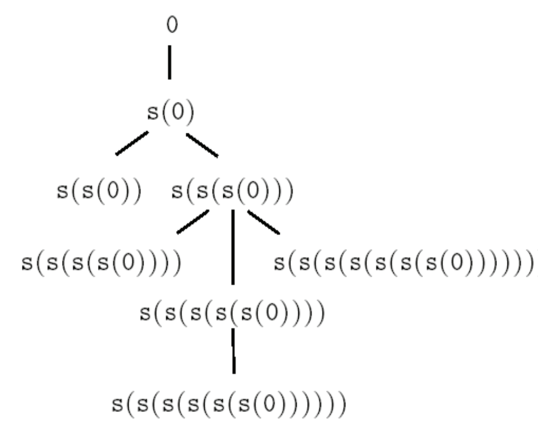
mRNA

[Kersting et al.; Kersting, Gaertner]

```
nucleotide_pair((c, g)).
nucleotide_pair((c, g)).
nucleotide_pair((c, g)).
helical(s(s(s(s(s(0))))), s(s(s(0))), [c], stem, n(n(n(0)))).
nucleotide(a).
nucleotide(a).
nucleotide(g).
single(s(s(s(s(0)))), s(s(s(0))), [], bulge5, n(n(0))).
nucleotide_pair((c, g)).
nucleotide_pair((c, g)).
helical(s(s(s(0))), s(0), [c, c, c], stem, n(n(0))).
nucleotide(a).
single(s(s(0)), s(0), [], bulge5, n(0)).
nucleotide_pair((a, a)).
nucleotide_pair((u, a)).
nucleotide_pair((u, g)).
nucleotide_pair((u, a)).
nucleotide_pair((c, a)).
nucleotide_pair((u, a)).
nucleotide_pair((u, u)).
helical(s(0), 0, [c, c], stem, n(n(n(n(n(n(0))))))).
root(0, root, [c]).
```




```
single(s(s(s(s(s(s(0)))))), s(s(s(s(0)))),
    [], hairpin, n(n(n(0)))).
nucleotide(a).
nucleotide(u).
nucleotide(u).

single(s(s(s(s(s(s(s(0))))))), s(s(s(0))),
    [], bulge3, n(0)).
nucleotide(a).
```



mRNA

[Kersting et al.; Kersting, Gaertner]

93 logical sequences (in total 3122 ground atoms)
- 15 and 5 SECIS (Selenocysteine Insertion Sequence),
- 27 IRE (Iron Responsive Element),
- 36 TAR (Trans Activating Region) and
- 10 histone stemloops.

Leave-one-out crossvalidation:

Plug-In Estimates:	4.3 % error
Fisher kernels SVM:	2.2 % error

Outline

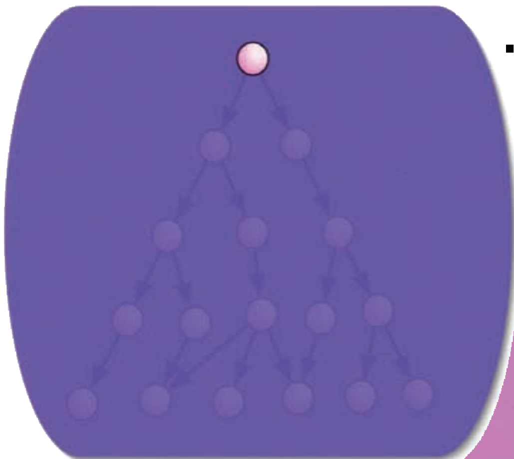
1. Motivation / Introudction
2. Inductive Logic Programming (ILP)
   - Logic
   - Learning setting, cover relation
   - Learning from entailment, interpretations, and traces/proofs
3. Probabilistic ILP
   - Learning setting, probabilistic cover relation
4. Probabilistic Learning from
   - Interpretations, entailment, and traces/proofs
5. Discriminative ILP
6. Conclusions

Conclusions

- A flavor of Probabilistic ILP or Statistical Relational Learning from a logical perspective
- A definition of Probabilistic ILP
  – based on a probabilistic coverage notion and annotated logic programs
- Different Probabilistic ILP settings (as for ILP)
  – Learning from entailment: Parameter Est. SLPs/Prism
  – Learning from interpretations: BLPs, PRMs, RMNs, MLNs
  – Learning from traces or proofs: SLPs, RMMs, LOHMMs
  – Discriminative ILP: nFOIL
- Different settings have different complexities

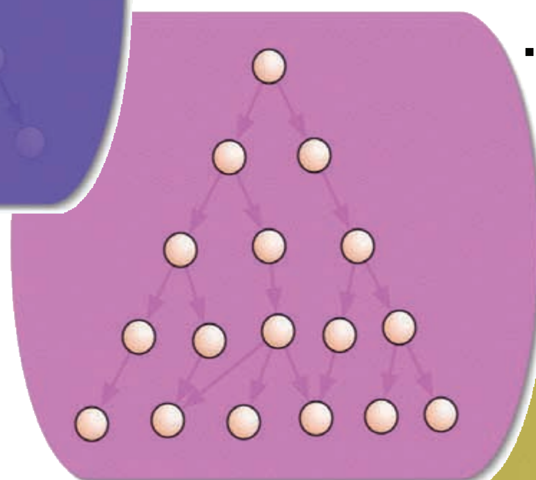
Use of different Prob. ILP Settings

Probabilistic Learning from …



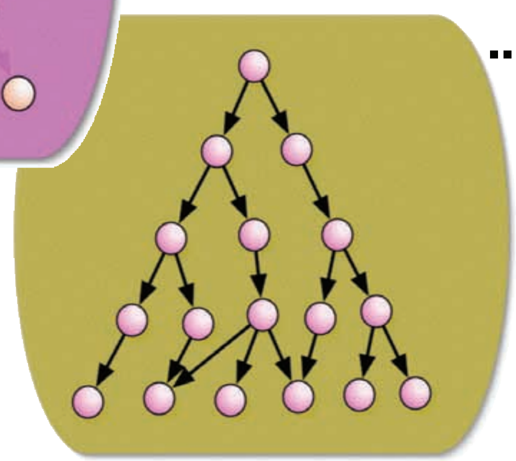
… Entailment

SLPs, PRISM



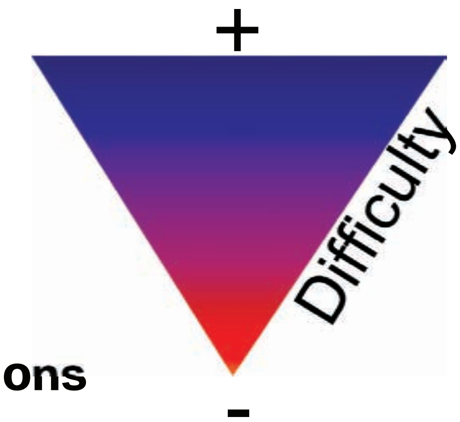
... Interpretations

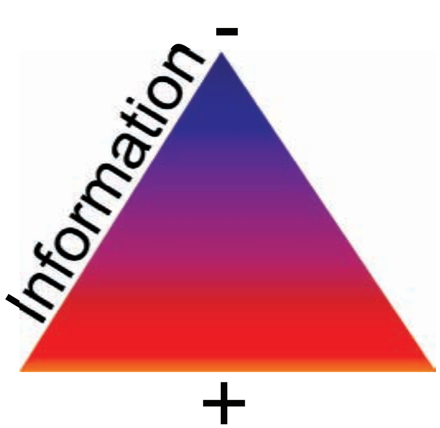
PRMs, BLPs, RMNs, MLNs



... Proofs/Traces

RMMs, LOHMMs, SLPs





Conclusions

Many interesting problems left !

Thank you for your attention !

Please join PILP / SRL !

http://www.aprill.org

http://www.informatik.uni-freiburg.de/~kersting/plmr

Special thanks to the APrIL II consortium

- „Application of Probabilistic ILP“
- 3 years EU project
- 5 institutes
- www.aprill.org