Advanced Artificial Intelligence

Part II. Statistical NLP

Probabilistic Logic Learning

Wolfram Burgard, Luc De Raedt, Bernhard Nebel, Lars Schmidt-Thieme

Many slides taken from Kristian Kersting and for Logic
From Peter Flach's Simply Logical

Overview

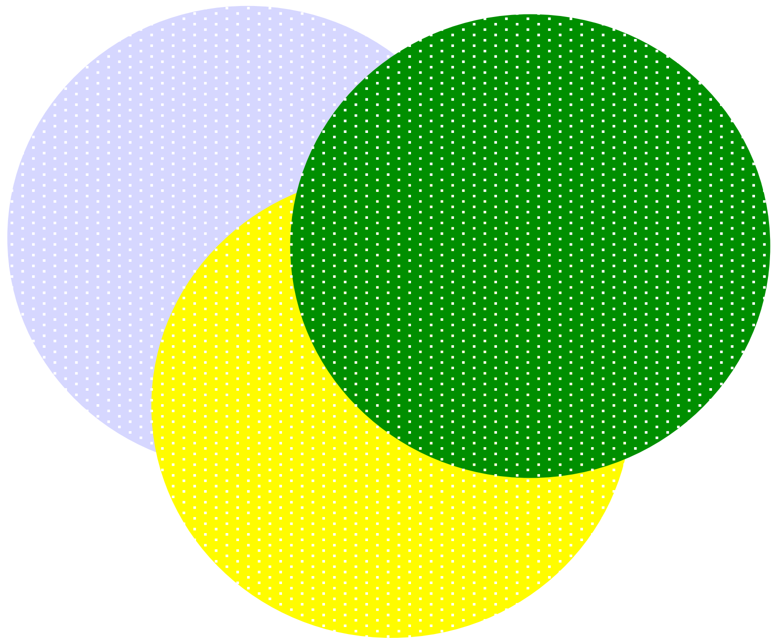
- Expressive power of PCFGs, HMMs, BNs still limited
  - First order logic is more expressive
- Why not combine logic with probabilities ?
  - Probabilistic logic learning
- Short recap of logic (programs)
- Stochastic logic programs
  - Extend PCFGs
- Bayesian logic programs
  - Extend Bayesian Nets
- Logical HMMs
  - Extend HMMs

Context

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.

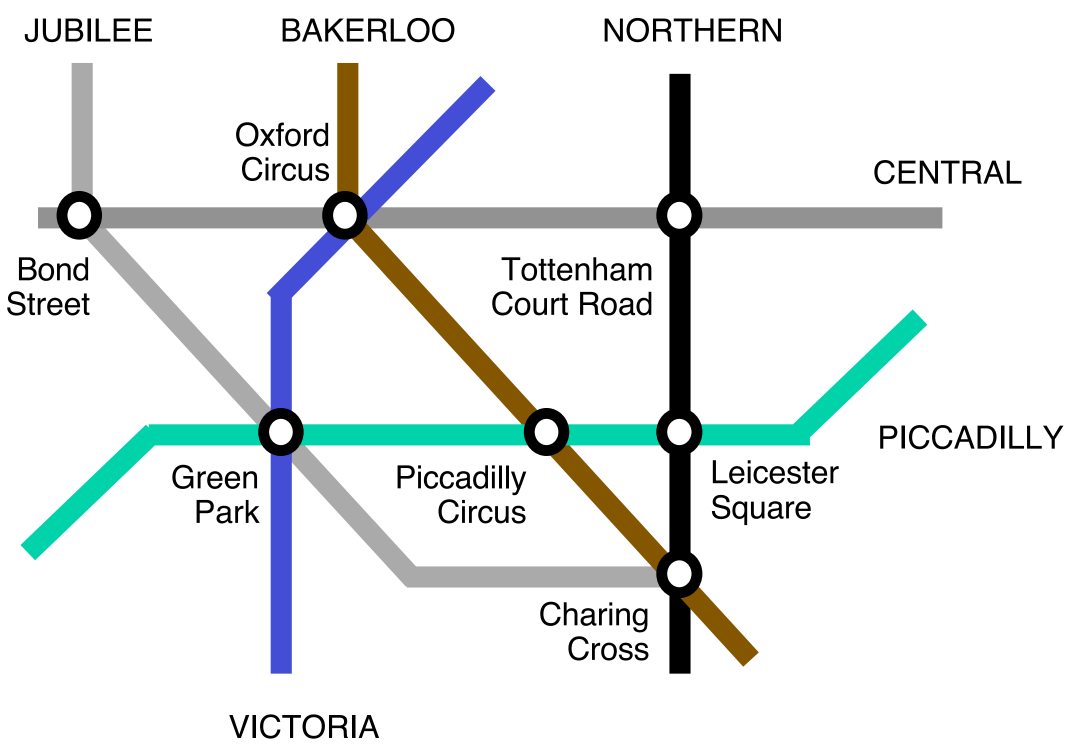


Sometimes called Statistical Relational Learning

So far

- We have largely been looking at probabilistic representations and ways of learning these from data
  - BNs, HMMs, PCFGs
- Now, we are going to look at their expressive power, and make traditional probabilistic representations more expressive using logic
  - Probabilistic First Order Logics
  - Lift BNs, HMMs, PCFGs to more expressive frameworks
  - Upgrade also the underlying algorithms

London Underground example



UNDERGROUND

LRT Registered User No. 94/1954

London Underground in Prolog (1)

```
connected(bond_street,oxford_circus,central).
connected(oxford_circus,tottenham_court_road,central).
connected(bond_street,green_park,jubilee).
connected(green_park,charing_cross,jubilee).
connected(green_park,piccadilly_circus,piccadilly).
connected(piccadilly_circus,leicester_square,piccadilly).
connected(green_park,oxford_circus,victoria).
connected(oxford_circus,piccadilly_circus,bakerloo).
connected(piccadilly_circus,charing_cross,bakerloo).
connected(tottenham_court_road,leicester_square,northern).
connected(leicester_square,charing_cross,northern).
```

Symmetric facts now shown !!!

London Underground in Prolog (2)

Two stations are nearby if they are on the same line with at most one other station in between (symmetric facts not showr

```
nearby(bond_street,oxford_circus).
nearby(oxford_circus,tottenham_court_road).
nearby(bond_street,tottenham_court_road).
nearby(bond_street,green_park).
nearby(green_park,charing_cross).
nearby(bond_street,charing_cross).
nearby(green_park,piccadilly_circus).
```

or better

```
nearby(X,Y):-connected(X,Y,L).
nearby(X,Y):-connected(X,Z,L),connected(Z,Y,L).
```

Facts: unconditional truths
Rules/Clauses: conditional truths

Both definitions are equivalent.

"Peter likes anybody who is his student."

atoms

clause

```
likes(peter,S):-student_of(S,peter).
```

constant variable

terms

Clauses are universally quantified !!!

:- denotes implication

Recursion (2)

A station is reachable from another if they are on the same line, or with one, two, … changes:

```
reachable(X,Y):-connected(X,Y,L).
reachable(X,Y):-connected(X,Z,L1),connected(Z,Y,L2).
reachable(X,Y):-connected(X,Z1,L1),connected(Z1,Z2,L2),
                connected(Z2,Y,L3).
…
```

or better

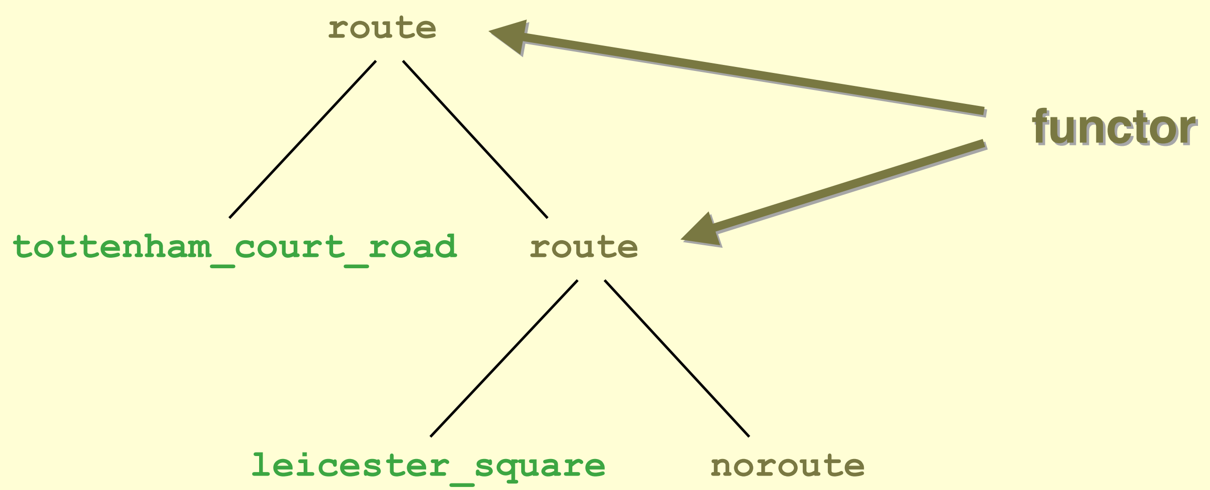
```
reachable(X,Y):-connected(X,Y,L).
reachable(X,Y):-connected(X,Z,L),reachable(Z,Y).
```

Substitutions

- A ***substitution*** maps variables to terms:
  - {S->maria}
- A substitution can be ***applied*** to a clause:
  - likes(peter,maria):-student_of(maria,peter).
- The resulting clause is said to be an ***instance*** of the original clause, and a ***ground instance*** if it does not contain variables.
- Each instance of a clause is among its logical consequences.

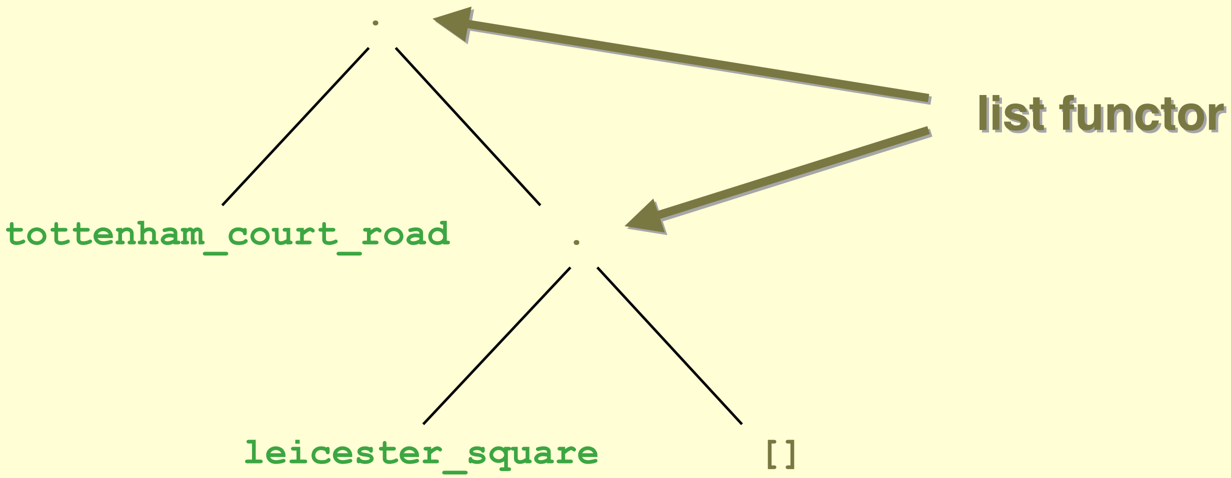
Structured terms (2)

```
reachable(X,Y,noroute):-connected(X,Y,L).
reachable(X,Y,route(Z,R)):-connected(X,Z,L),
                           reachable(Z,Y,R).

?-reachable(oxford_circus,charing_cross,R).
R = route(tottenham_court_road,route(leicester_square,noroute));
R = route(piccadilly_circus,noroute);
R = route(picadilly_circus,route(leicester_square,noroute))
```



Lists (3)

```
reachable(X,Y,[]):-connected(X,Y,L).
reachable(X,Y,[Z|R]):-connected(X,Z,L),
                      reachable(Z,Y,R).

?-reachable(oxford_circus,charing_cross,R).
R = [tottenham_court_road,leicester_square];
R = [piccadilly_circus];
R = [picadilly_circus,leicester_square]
```



Answering queries (1)

- Query:
  which station is nearby Tottenham Court Road?

  `?- nearby(tottenham_court_road, W).`
- Prefix `?-` means it‘s a query and not a fact.
- Answer to query is:

  `{W -> leicester_square}`

  a so-called *substitution*.
- When nearby defined by facts, substitution found by unification.

Fig.1.2, p.7

Proof tree

```
?-nearby(tottenham_court_road,W)
                          nearby(X1,Y1):-connected(X1,Y1,L1)
      {X1->tottenham_court_road, Y1->W}
?-connected(tottenham_court_road,W,L1)
                          connected(tottenham_court_road,
                               leicester_square,northern)
      {W->leicester_square, L1->northern}
[]
```

query

clause

answer substitution

fact

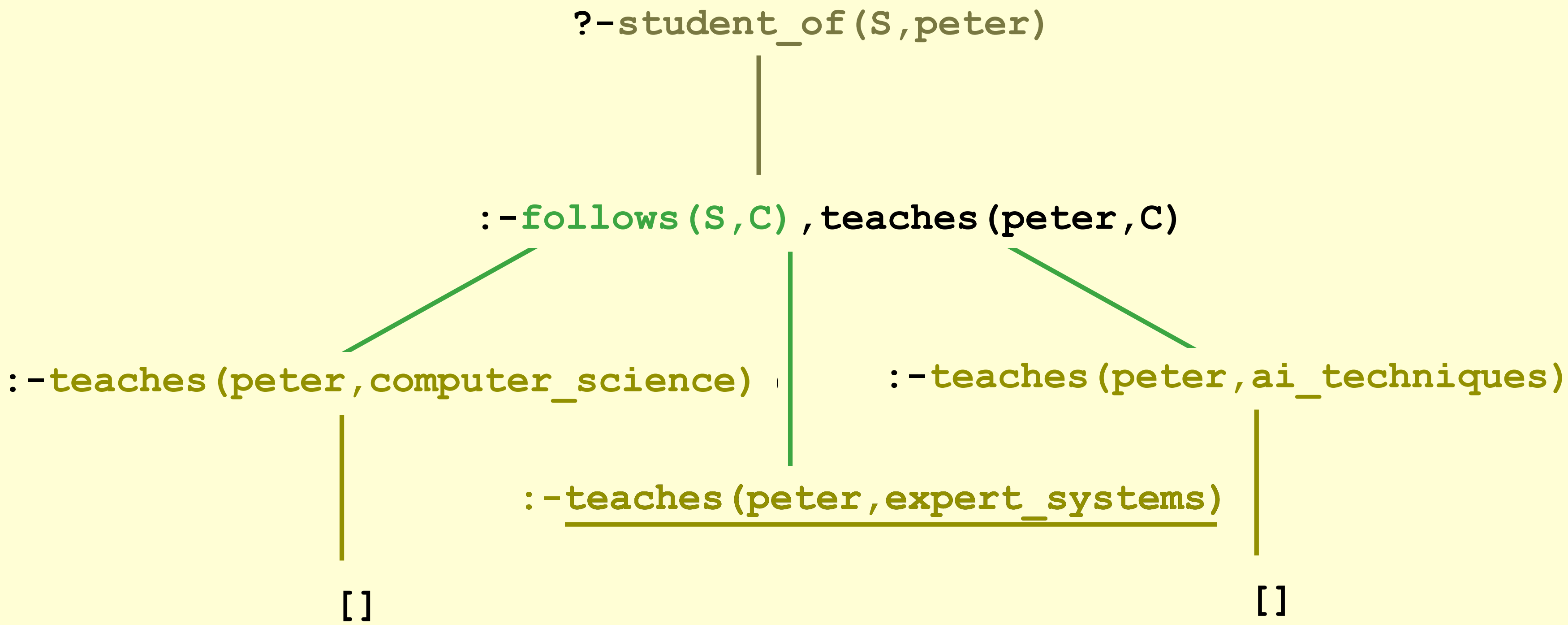
empty query

substitution

Recall from AI course

- Unification to unify two different terms
- Resolution inference rule
- Refutation proofs, which derive the empty clause
- SLD-tree, which summarizes all possible proofs (left to right) for a goal

```
student_of(X,T):-follows(X,C),teaches(T,C).
follows(paul,computer_science).
follows(paul,expert_systems).
follows(maria,ai_techniques).
teaches(adrian,expert_systems).
teaches(peter,ai_techniques).
teaches(peter,computer_science).
```



SLD-tree: one path for each proof-tree

The least Herbrand model

- Definition:
  - The set of all ground facts that are logically entailed by the program
- All ground facts not in the LHM are false …
- LHM be computed as follows:
  - M0 = {}; M1 = { true }; i:=0
  - while Mi =\= Mi+1 do
    - i := i +1;
    - Mi := { h θ | h:- b1, …, bn is clause and there is a substitution θ such that all bi θ $\in$ Mi-1 }
  - Mi contains all true facts, all others are false

Example LHM

KB: p(a,b). a(X,Y) :- p(X,Y).

p(b,c). a(X,Y) :- p(X,Z), a(Z,Y).

M0 = emtpy;

M1 = { true }

M2 = { true, p(a,b), p(b,c) }

M3 = M2 U {a(a,b), a(b,c) }

M4 = M3 U { a(a,c) }

M5 = M4

...

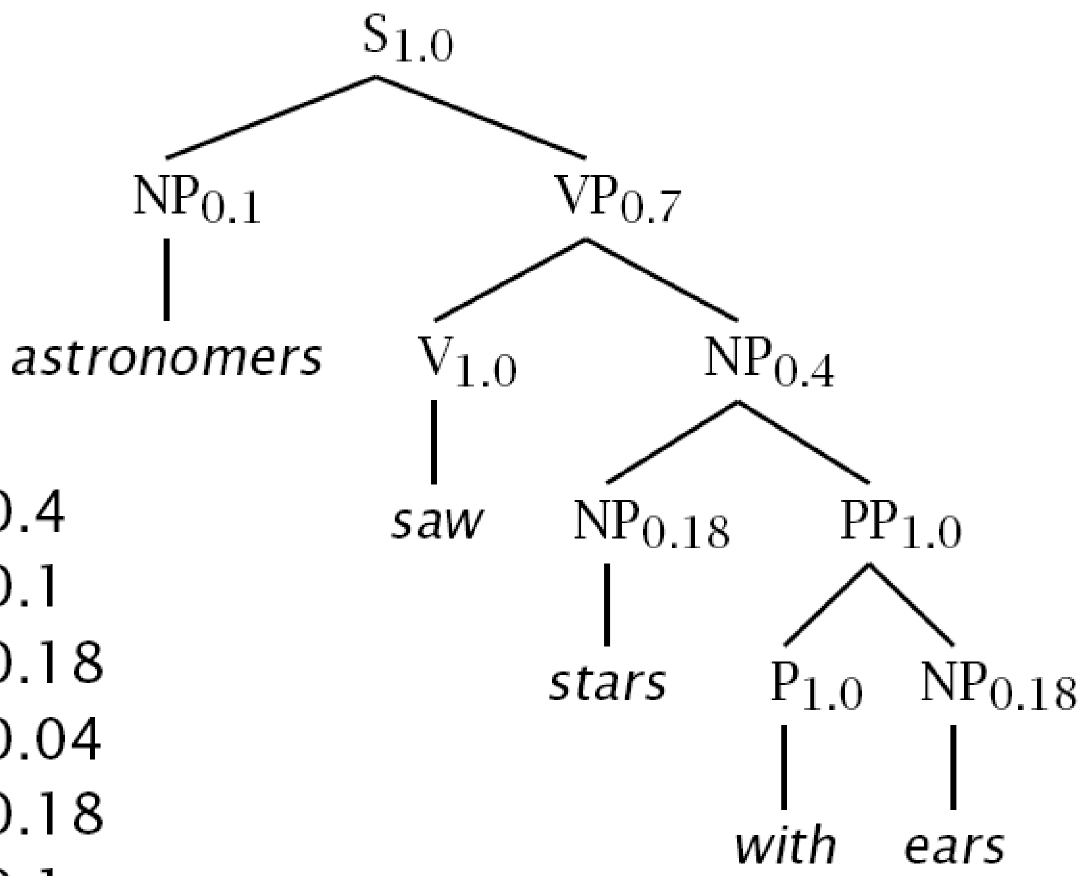
Stochastic Logic Programs

- Recall :
  - Prob. Regular Grammars
  - Prob. Context-Free Grammars
- What about Prob. Turing Machines ? Or Prob. Grammars ?
  - Stochastic logic programs combine probabilistic reasoning in the style of PCFGs with the expressive power of a programming language.

Recall PCFGs

S → NP VP	1.0
PP → P NP	1.0
VP → V NP	0.7
VP → VP PP	0.3
P → *with*	1.0
V → *saw*	1.0

NP → NP PP	0.4
NP → *astronomers*	0.1
NP → *ears*	0.18
NP → *saw*	0.04
NP → *stars*	0.18
NP → *telescopes*	0.1



We defined

- $P(tree|G) = \prod_i p_i^{c_i}$ where i ranges over all rules i used to derive tree, and c_i is the number of times they were applied
- $P(w_{1m}|G) = \sum_j P(tree_j|G)$ where j ranges over all possible parse trees for w_{1m}.
- Key concept : probabilities over derivations !!!

Stochastic Logic Programs

- Correspondence between CFG - SLP
  - Symbols - Predicates
  - Rules - Clauses
  - Derivations - SLD-derivations/Proofs
- So,
  - a stochastic logic program is an annotated logic program.
  - Each clause has an associated probability label. The sum of the probability labels for clauses defining a particular predicate is equal to 1.

An Example

$1 : \text{card}(X, Y) \leftarrow \text{rank}(X), \text{suit}(Y)$

$0.25 : \text{suit}(d) \leftarrow$

$0.25 : \text{suit}(h) \leftarrow$

$0.125 : \text{rank}(a) \leftarrow$

$0.125 : \text{rank}(7) \leftarrow$

$0.125 : \text{rank}(8) \leftarrow$

$0.125 : \text{rank}(9) \leftarrow$

$0.25 : \text{suit}(c) \leftarrow$

$0.25 : \text{suit}(s) \leftarrow$

$0.125 : \text{rank}(10) \leftarrow$

$0.125 : \text{rank}(k) \leftarrow$

$0.125 : \text{rank}(q) \leftarrow$

$0.125 : \text{rank}(f) \leftarrow$

:-card(a,s)

|

:-rank(a), suit(s)

|

:-suit(s)

|

[]

Prob derivation
= 1 . 0.125 . 0.25

Example

$1 : \mathrm{sentence}(A, B) \leftarrow \mathrm{noun} - \mathrm{phrase}(C, A, D), \mathrm{verb} - \mathrm{phrase}(C, D, B).$
$1 : \mathrm{noun} - \mathrm{phrase}(A, B, C) \leftarrow \mathrm{article}(A, B, D), \mathrm{noun}(A, D, C).$
$1 : \mathrm{verb} - \mathrm{phrase}(A, B, C) \leftarrow \mathrm{intransitive} - \mathrm{verb}(A, B, C).$
$1/3 : \mathrm{article}(\mathrm{singular}, A, B) \leftarrow \mathrm{terminal}(A, a, B).$
$1/3 : \mathrm{article}(\mathrm{singular}, A, B) \leftarrow \mathrm{terminal}(A, \mathrm{the}, B).$
$1/3 : \mathrm{article}(\mathrm{plural}, A, B) \leftarrow \mathrm{terminal}(A, \mathrm{the}, B).$
$1/2 : \mathrm{noun}(\mathrm{singular}, A, B) \leftarrow \mathrm{terminal}(A, \mathrm{turtle}, B).$
$1/2 : \mathrm{noun}(\mathrm{plural}, A, B) \leftarrow \mathrm{terminal}(A, \mathrm{turtles}, B).$
$1/2 : \mathrm{intransitive} - \mathrm{verb}(\mathrm{singular}, A, B) \leftarrow \mathrm{terminal}(A, \mathrm{sleeps}, B).$
$1/2 : \mathrm{intransitive} - \mathrm{verb}(\mathrm{plural}, A, B) \leftarrow \mathrm{terminal}(A, \mathrm{sleep}, B).$
$1 : \mathrm{terminal}([A|B], A, B) \leftarrow$

s([the,turtle,sleeps],[]) ?

SLPs : Key Ideas

- let us consider goals g (atoms, queries) of the form $p(X_1, ..., X_n)$ with the X_i different variables; corresponds to a non-terminal in a PCFG
- $P_D(der\ g|SLP) = \prod_i p_i^{c_i}$ where i ranges over all clauses i used in the derivation $der\ g$ for goal g and c_i is the number of times i was applied; so far this is similar as for PCFGs
- **key difference with PCFGs**:
  - derivations in PCFGs always succeed,
  - proofs in SLPs can fail;
  - *refutations* are successful proofs
  - we are interested in the conditional probability of a derivation given that we know it is succesful / a refutation
- Therefore, define$P_R(ref\ g|SLP) = \frac{P_D(ref\ g)}{\sum_j P_D(ref_j\ g|SLP)}$, the probability P_R of refutation ref of g; where j ranges over all refutations ref_j of the goal g
- For ground atoms $g\theta$ (where θ is the substitution grounding g), define : $P_A(g\theta|SLP) = \sum_j P_R(ref_j\ g\theta|SLP)$ where ref_j ranges over all possible refutations for $g\theta$.

Example

- Cards :
  - card(R,S) - no proof with R in {a,7,8,9…} and S in { d,h,s,c} fails
  - For each card, there is a unique refutation
  - So,

$$P_A(card(r,s)) = P_D(derivation\ card(r,s))$$
$$= P_R(refutation\ card(r,s)) = 1/32$$

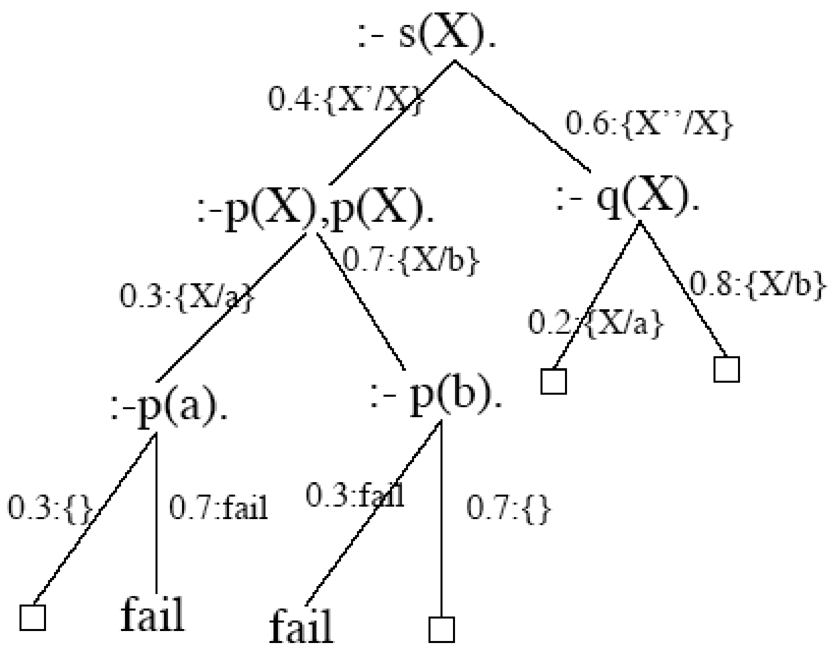
Consider

- same_suit(S,S) :-
  suit(S), suit(S).
- In total 16 possible derivations, only 4 will succeed, so

$$P_D(derivation\ samesuit(S1, S2)) = 1/16$$
$$P_R(refutation\ samesuit(s, s)) = 1/4$$
$$P_A(samesuit(s, s)) = 1/4$$

Another example (due to Cussens)

```
0.4:s(X) :- p(X), p(X).          0.3:p(a).     0.2:q(a).
0.6:s(X) :- q(X).                0.7:p(b).     0.8:q(b).
```



$$P_A(s(a)) = (0.4 \times 0.3 \times 0.3 + 0.6 \times 0.2)/0.832 = 0.1875$$

$$P_A\, s(b)) = (0.4 \times 0.7 \times 0.7 + 0.6 \times 0.8)/0.832 = 0.8125$$

Questions we can ask (and answer) about SLPs

- Compute the probability $P_A(a)$ of a ground atom a
- Find the most likely refutation r of a goal g, i.e. $argmax_{ref} P_R(ref\ g)$
- For a given SLP and set of atoms At for a goal g, compute the maximum likelihood parameters λ , i.e. $argmax_\lambda(\prod_{a \in At} P_A(a|SLP(\lambda))$

Answers

- The algorithmic answers to these questions, again extend those of PCFGs and HMMs, in particular,
  - Tabling is used (to record probabilities of partial proofs and intermediate atoms)
  - Failure Adjusted EM (FAM) is used to solve parameter re-estimation problem
    - Additional hidden variables range over
      - Possible refutations and derivations for observed atoms
      - Topic of recent research
      - Freiburg : learning from refutations (instead of atoms), combined with structure learning

Sampling

- PRGs, PCFGs, and SLPs can also be used for sampling sentences, ground atoms that follow from the program
- Rather straightforward. Consider SLPs:
  - Probabilistically explore SLD-tree
  - At each step, select possible resolvents using the probability labels attached to clauses
  - If derivation succeeds, return corresponding (ground) atom
  - If derivation fails, then restart.

Bayesian Networks

[Pearl 91]

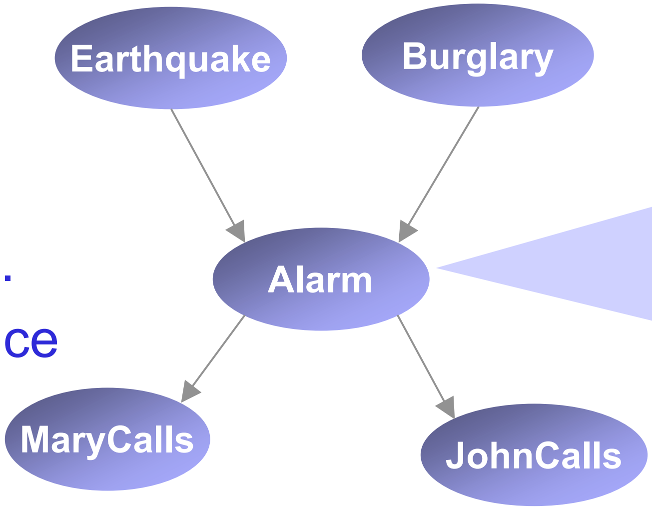
Compact representation of joint probability distributions

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

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

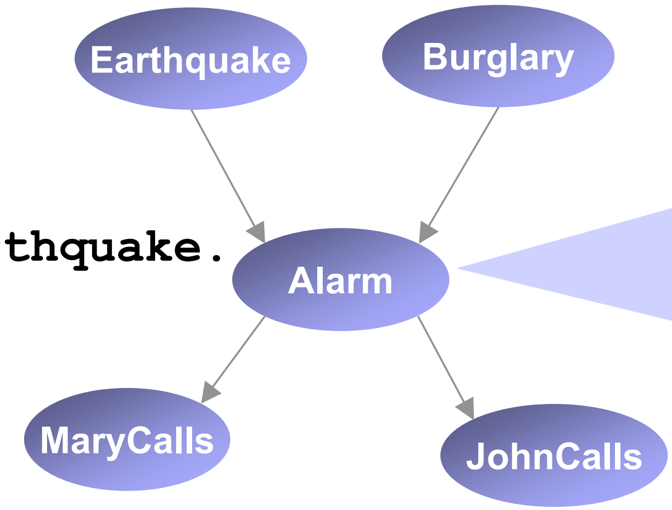
Together:
Define a unique distribution in a compact, factored form

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

Bayesian Networks

[Pearl 91]

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



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

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

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

...

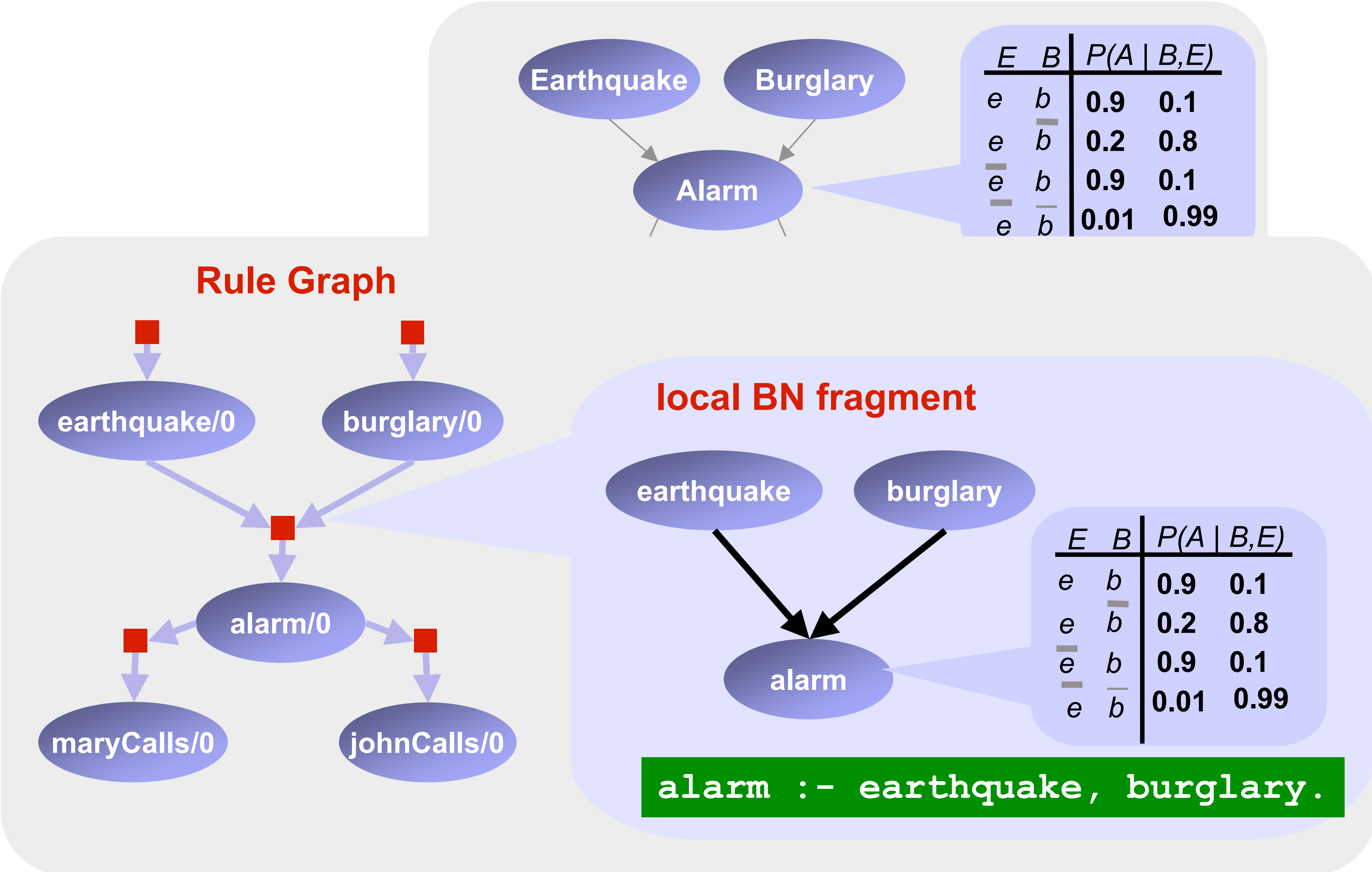
$$+ P(j|\bar{a}) * P(\bar{m}|\bar{a}) * P(\bar{a}|\bar{e},\bar{b}) * P(\bar{e}) * P(\bar{b})$$

Expressiveness Bayesian Nets

- A Bayesian net defines a probability distribution over a propositional logic
- Essentially, the possible states (worlds) are propositional interpretations
- But propositional logic is severely limited in expressive power, therefore consider combining BNs with logic programs
  - Bayesian logic programs
  - Actually, a BLP + some background knowledge generates a BN
  - So, BLP is a kind of BN template !!!

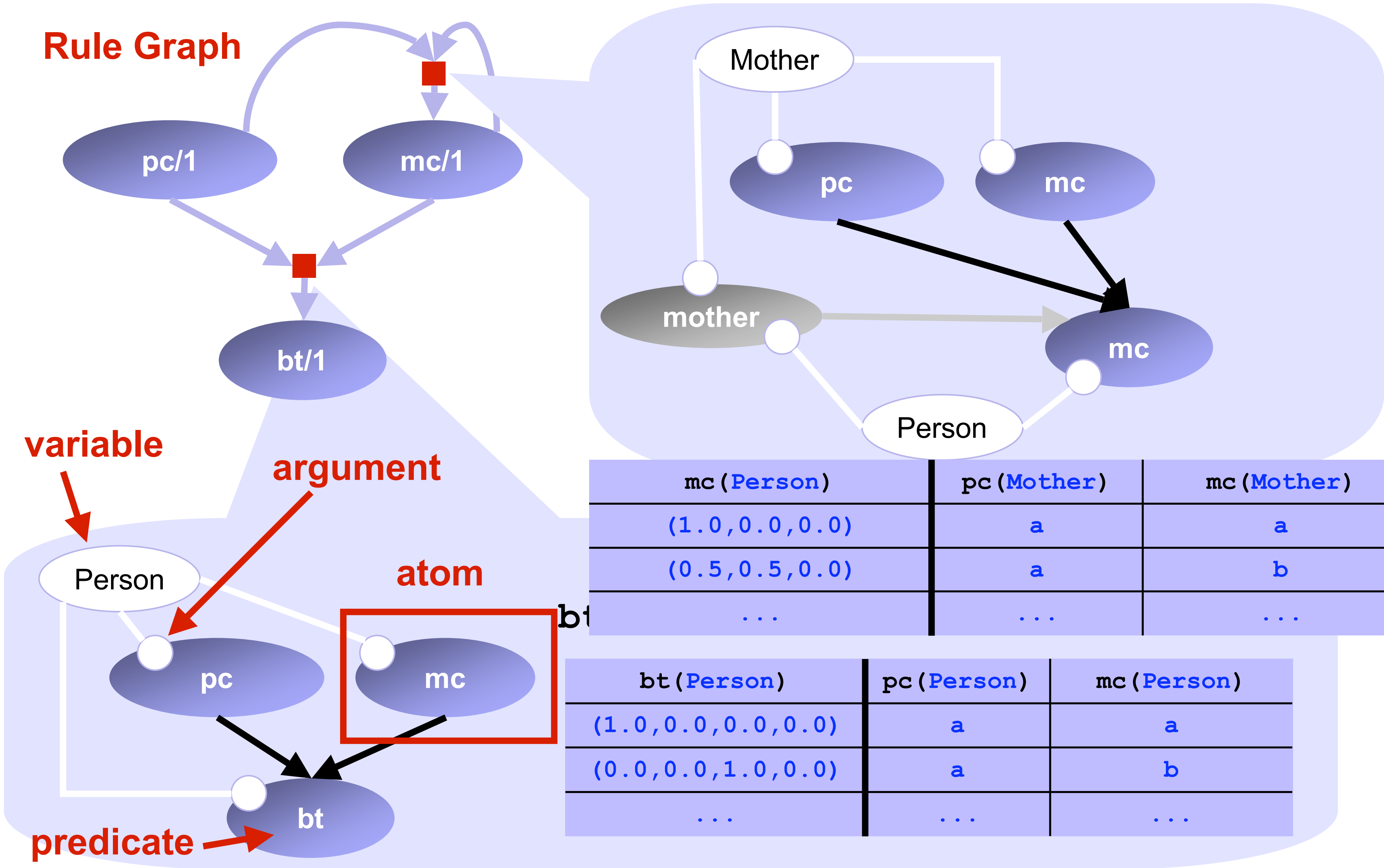
[Kersting, De Raedt]

Bayesian Logic Programs (BLPs)



[Kersting, De Raedt]

Bayesian Logic Programs (BLPs)

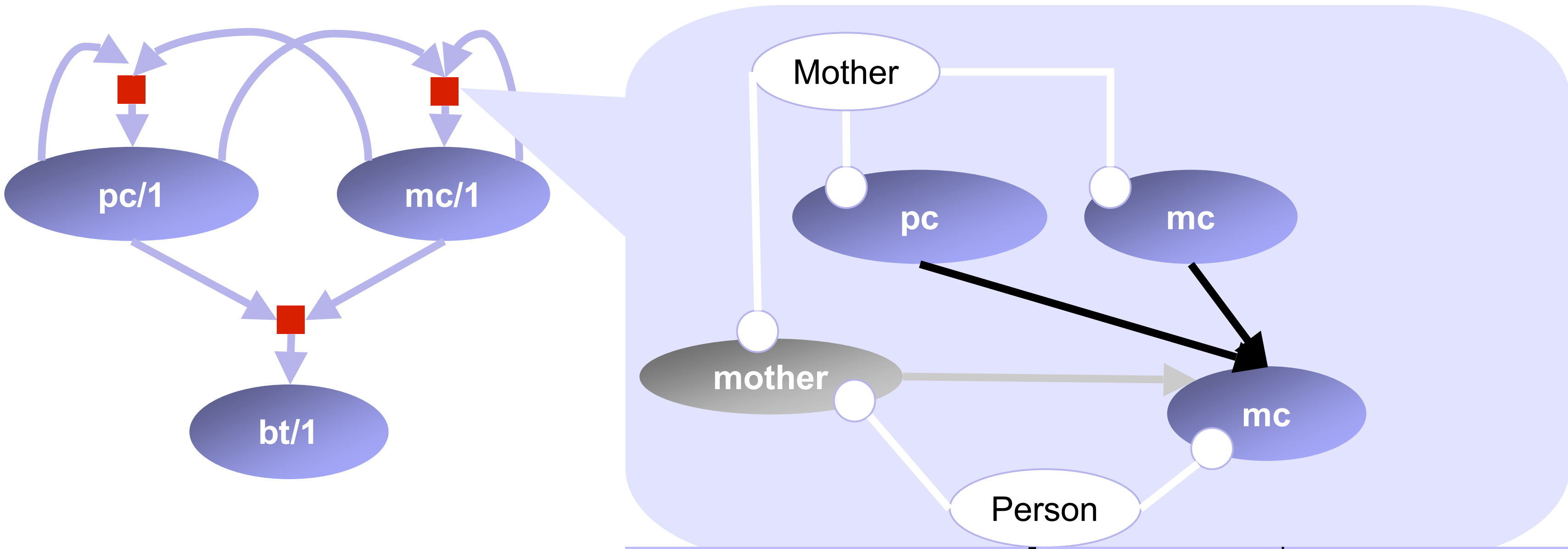


mc(Person)	pc(Mother)	mc(Mother)
(1.0,0.0,0.0)	a	a
(0.5,0.5,0.0)	a	b
...	...	...

bt(Person)	pc(Person)	mc(Person)
(1.0,0.0,0.0,0.0)	a	a
(0.0,0.0,1.0,0.0)	a	b
...	...	...

[Kersting, De Raedt]

Bayesian Logic Programs (BLPs)



mc(Person)	pc(Mother)	mc(Mother)
(1.0,0.0,0.0)	a	a
(0.5,0.5,0.0)	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).
```

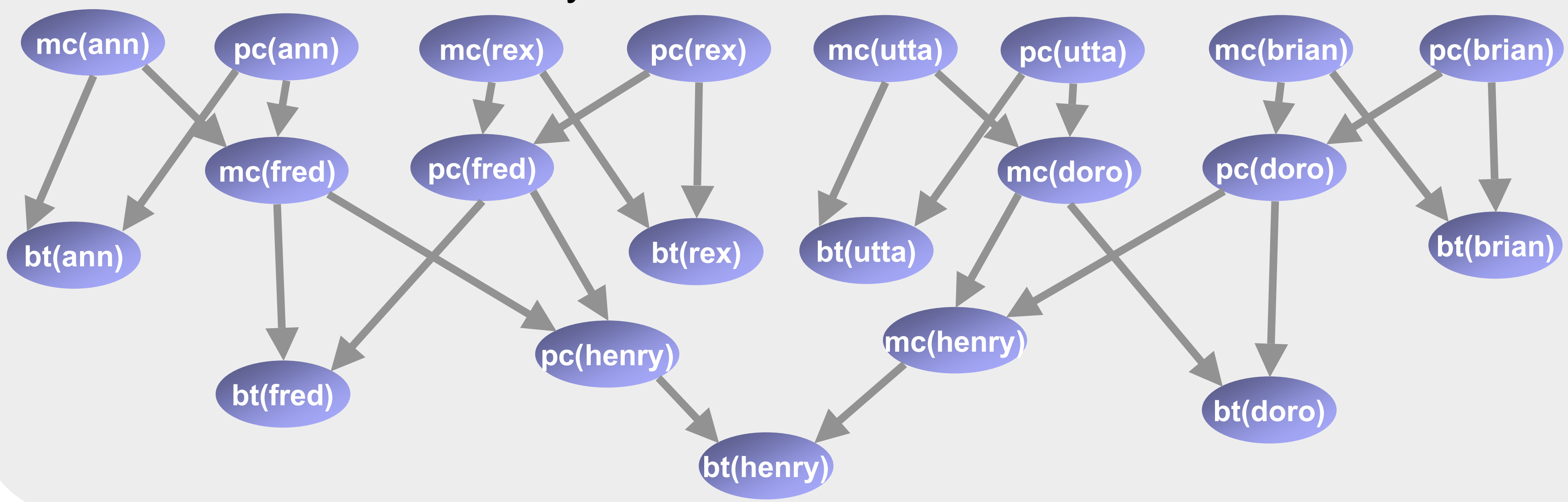
Bayesian Logic Programs (BLPs)

```
mc(ann).
pc(ann).
mc(utta).
pc(utta).
mc(brian).
pc(brian).
mc(rex).
pc(rex).
```

```
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).
```

Bayesian Network induced over least Herbrand model



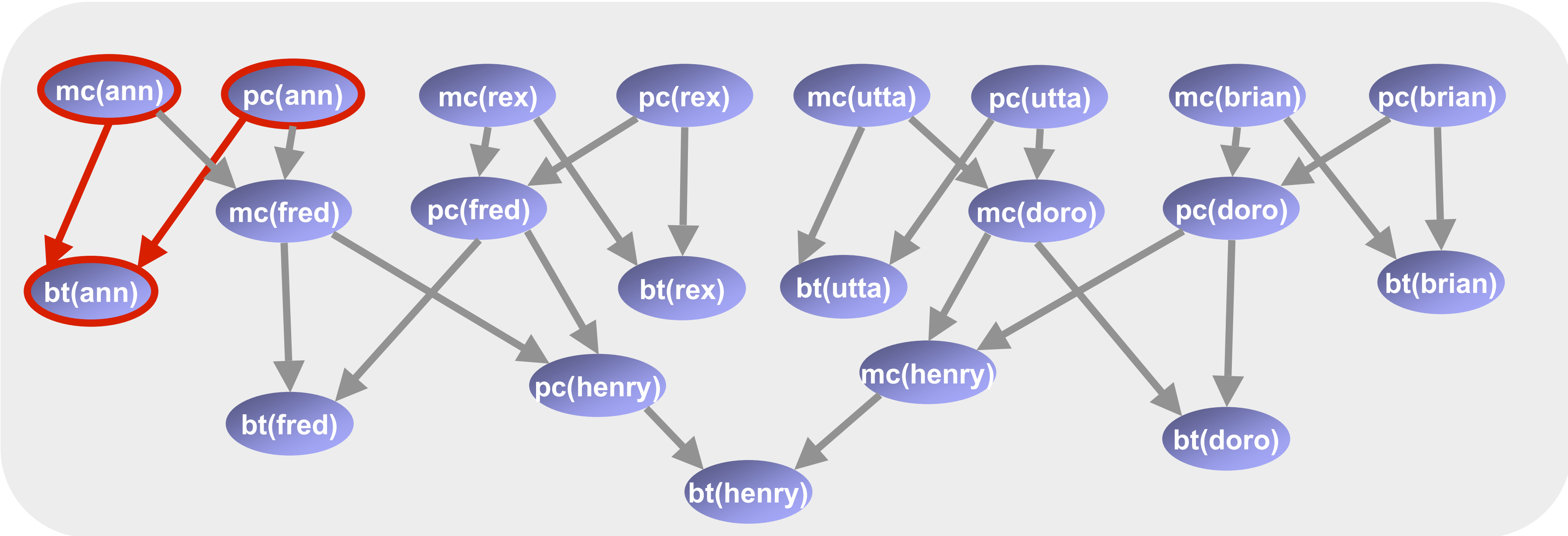
Bayesian logic programs

- Computing the ground BN (the BN that defines the semantics)
  - Compute the least Herbrand Model of the BLP
  - For each clause H | B1, … BN with CPD
    - if there is a substitution θ such that {H θ, B1 θ, …,BN θ} subset LHM, then H θ's parents include B1 θ, …,BN θ, and with CPD specified by the clause
  - Delete logical atoms from BN (as their truth-value is known) - e.g. mother, father in the example
  - Possibly apply aggregation and combining rules

♣ For specific queries, only part of the resulting BN is necessary, the support net, cf. Next slides

Procedural Semantics

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

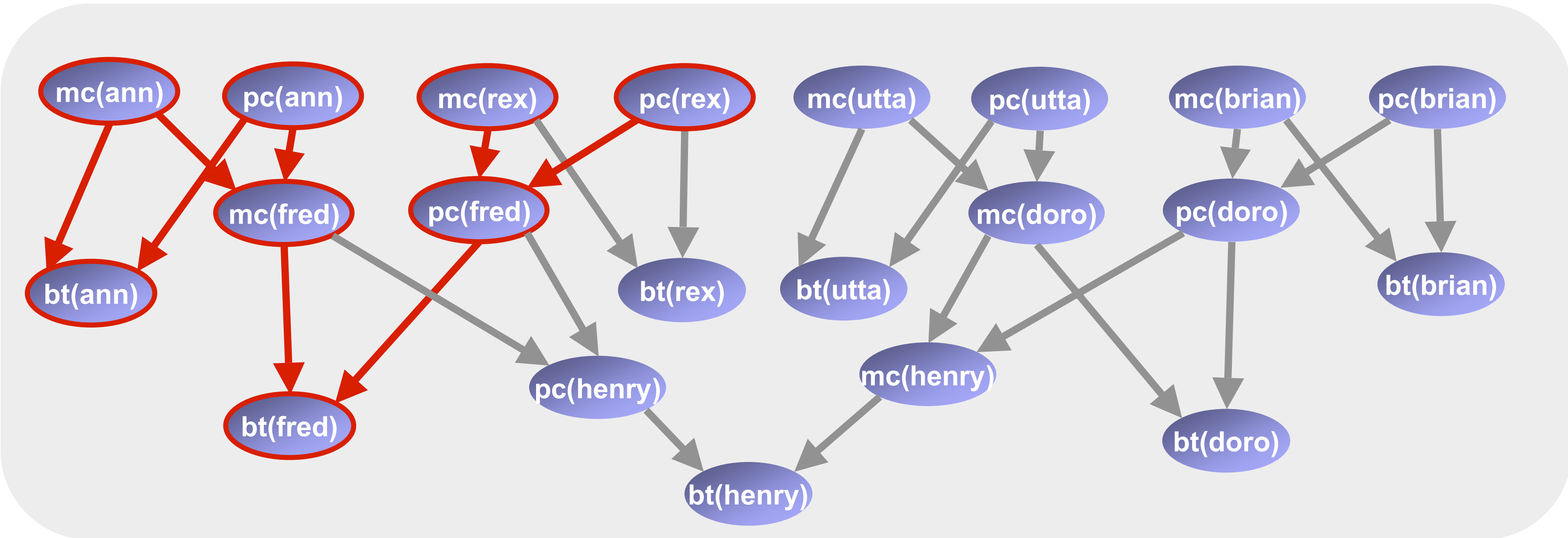


Procedural Semantics

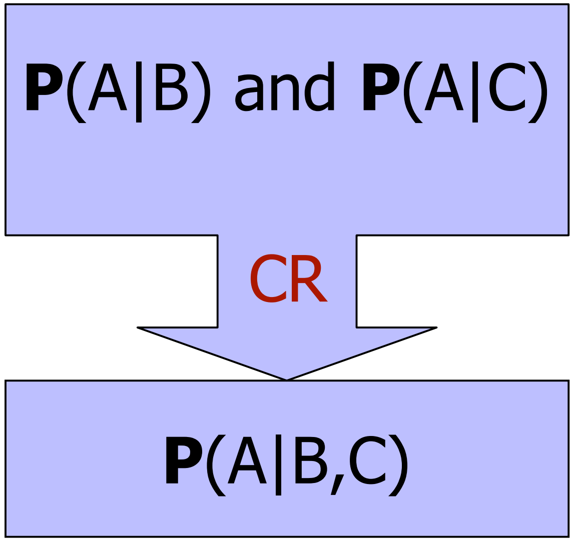
Bayes' rule

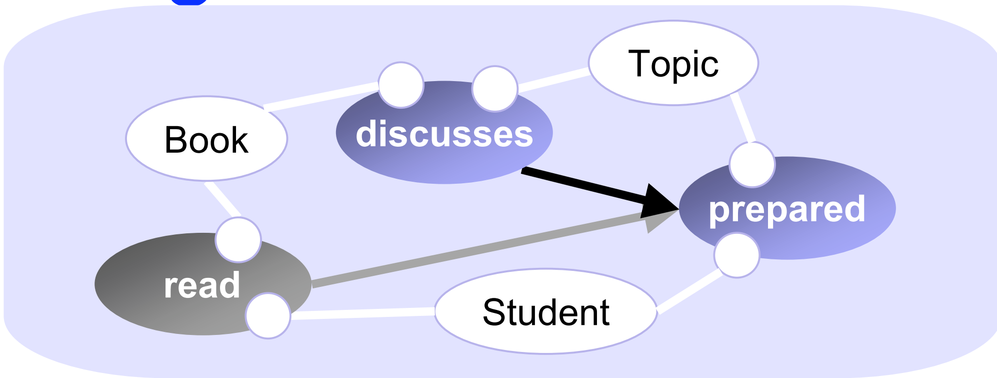
$$P(\texttt{bt(ann)} \mid \texttt{bt(fred)}) = \frac{P(\texttt{bt(ann)},\texttt{bt(fred)})}{P(\texttt{bt(fred)})}$$

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



Combining Rules





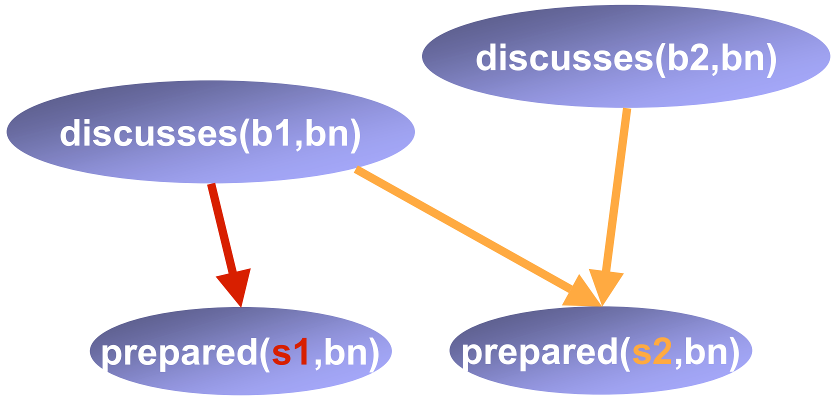

```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

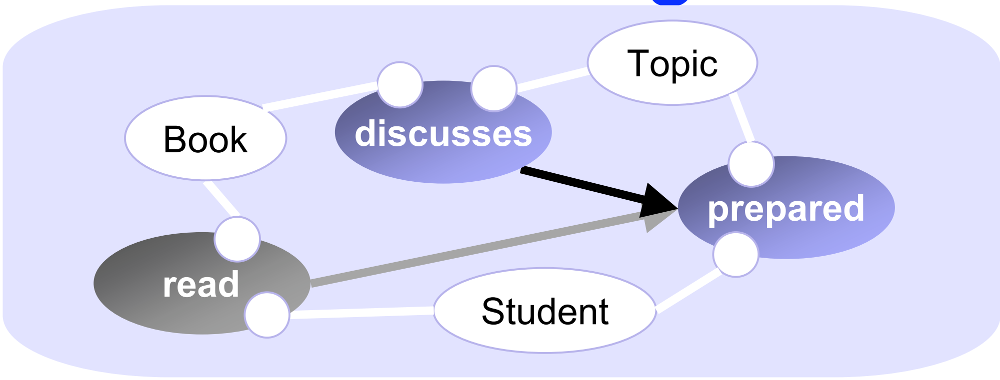
- Any algorithm which
  - has an empty output if and only if the input is empty
  - combines a set of CPDs into a single (combined) CPD
- E.g. noisy-or

Noisy-or: $P(A = true|V_1, ..., V_n) = 1 - \prod_{V_i = true} P(A = false|V_i = true)$

all causes can be independently inhibited

Combining Partial Knowledge

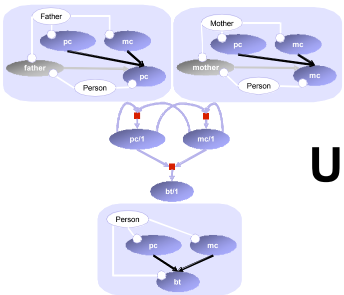





```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

- 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

Summary BLPs

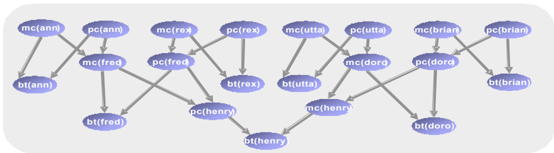



```
mc(Person) | mother(Mother,Person), pc(Mother),mc(Mother).
pc(Person) | father(Father,Person), pc(Father),mc(Father).
bt(Person) | pc(Person),mc(Person).
```

Underlying logic pogram

+ Consequence operator

If the body holds then the head holds, too.

= 

Conditional independencies encoded in the induced BN structure

Local probability models

+ (macro) CPDs

mc(Person)	pc(Mother)	mc(Mother)
(1.0,0.0,0.0)	a	a
(0.5,0.5,0.0)	a	b
...	...	...

+ CRs noisy-or, ...

= Joint probability distribution over the least Herbrand interpretation

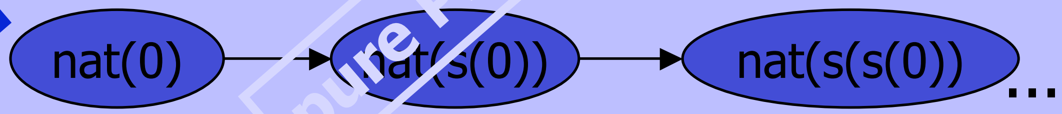
E

Bayesian Logic Programs - Examples

pure Prolog

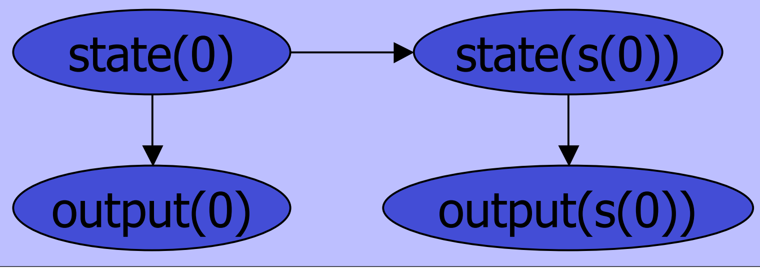
MC

```
% apriori nodes
nat(0).

% aposteriori nodes
nat(s(X)) | nat(X).
```



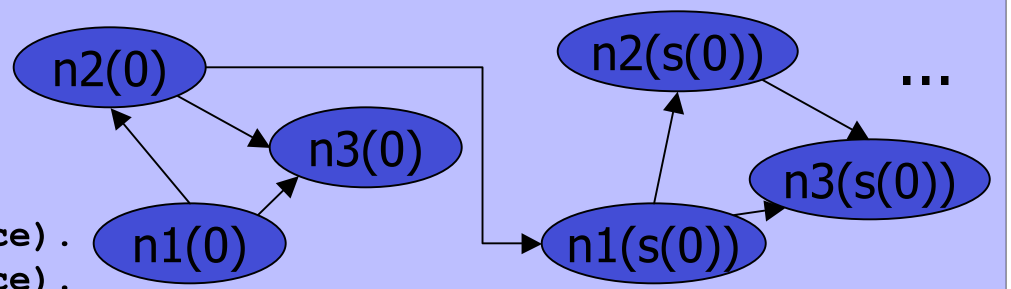
HMM

```
% apriori nodes
state(0).

% aposteriori nodes
state(s(Time)) | state(Time).
output(Time)   | state(Time)
```



DBN

```
% apriori nodes
n1(0).

% aposteriori nodes
n1(s(TimeSlice) | n2(TimeSlice).
n2(TimeSlice)   | n1(TimeSlice).
n3(TimeSlice)   | n1(TimeSlice), n2(TimeSlice).
```



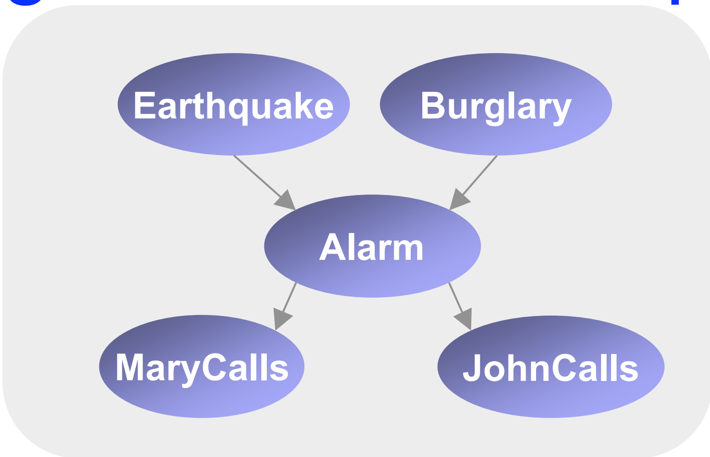
[Kersting, De Raedt]

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 ...
- Turing-complete programming language
- Learning

E

Learning BLPs from Interpretations



Model(1)

earthquake=yes,

burglary=no,

alarm=?,

marycalls=yes,

johncalls=no

Model(2)

earthquake=no,

burglary=no,

alarm=no,

marycalls=no,

johncalls=no

Model(3)

earthquake=?,

burglary=?,

alarm=yes,

marycalls=yes,

johncalls=yes

Learning BLPs from Interpretations

Data case:

- Random Variable + States = (partial) Herbrand interpretation

Model(1)

pc(brian)=b,

bt(ann)=a,

bt(brian)=?,

bt(dorothy)=a

Background

m(ann,dorothy),

f(brian,dorothy),

m(cecily,fred),

f(henry,fred),

f(fred,bob),

m(kim,bob),

...

Model(3)

pc(rex)=b,

bt(doro)=a,

bt(brian)=?

Model(2)

bt(cecily)=ab,

pc(henry)=a,

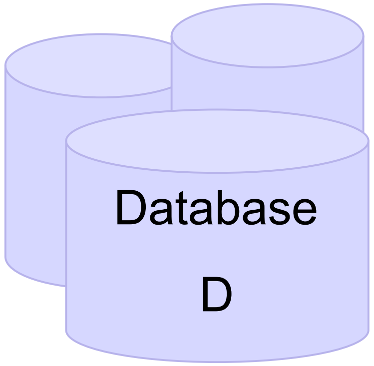
mc(fred)=?,

bt(kim)=a,

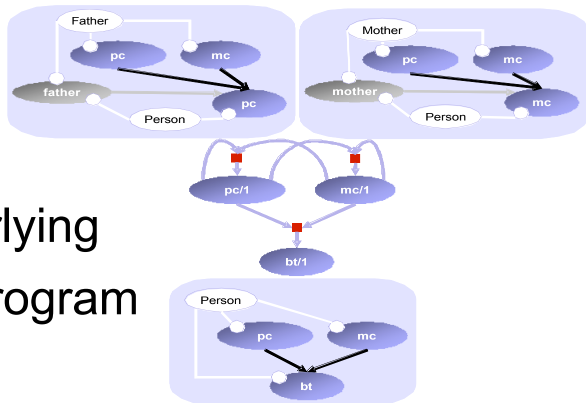
pc(bob)=b

Bloodtype example

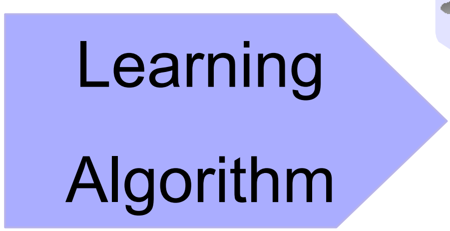
Parameter Estimation - BLPs

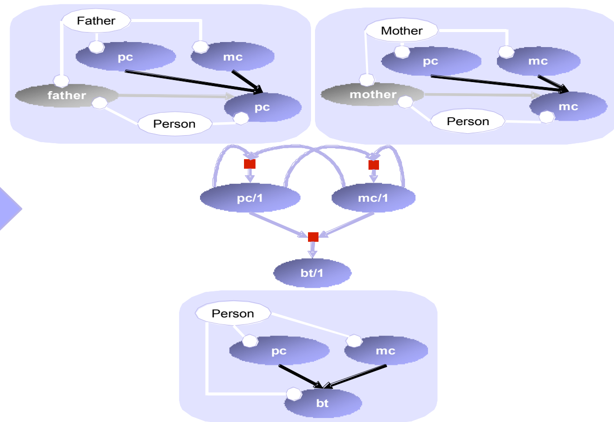


+



Underlying
Logic program
L





Parameter Θ

mc(Person)	pc(Mother)	mc(Mother)
(1.0,0.0,0.0)	a	a
(0.5,0.5,0.0)	a	b
...	...	...

Parameter Estimation – BLPs

- Estimate the CPD θ entries that best fit the data

- „Best fit“: ML parameters θ^*

  $\theta^* =$ argmax$_\theta$ P(data | logic program, θ)

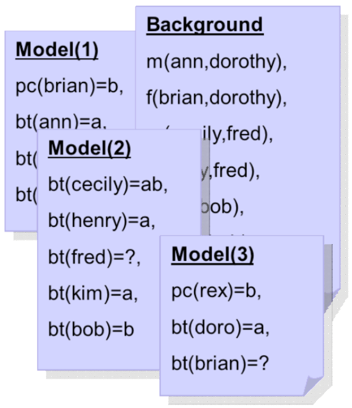
  $=$ argmax$_\theta$ log P(data | logic program, θ)

- Reduces to problem to estimate parameters of a Bayesian networks:

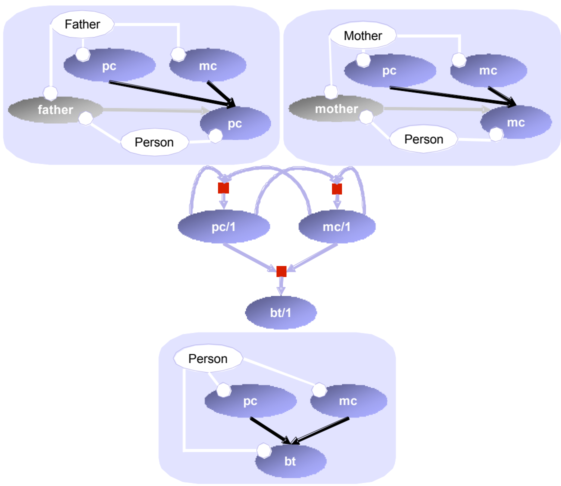
  given structure,

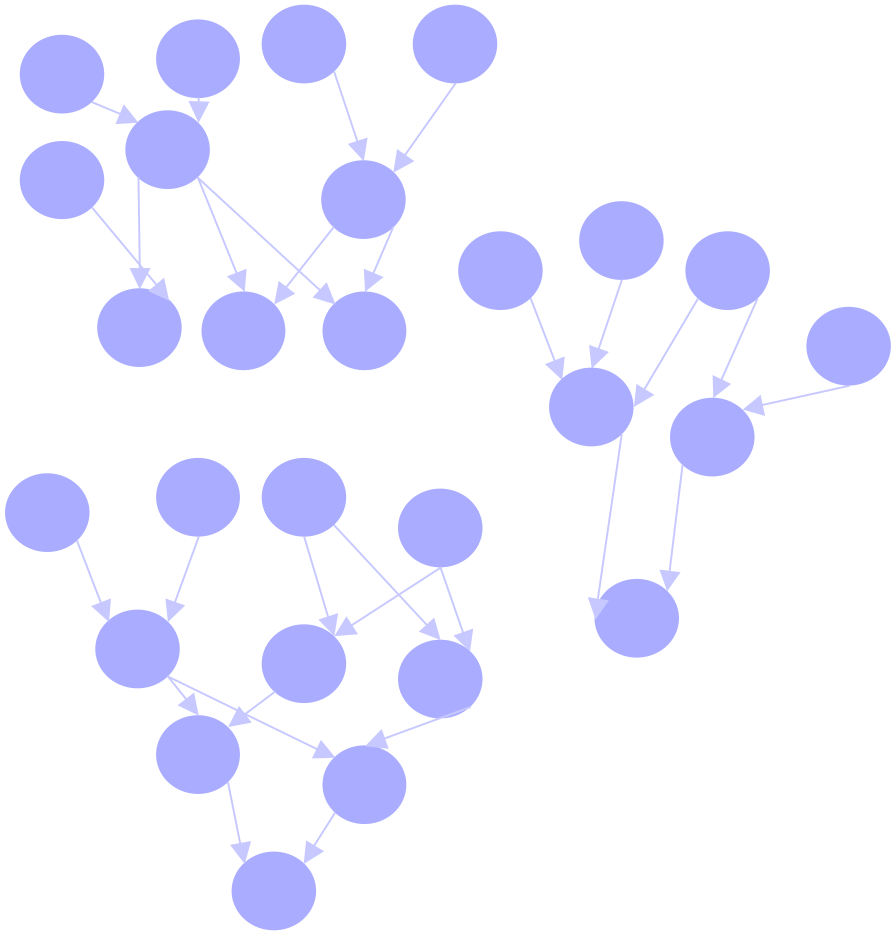
  partially observed random variables

Parameter Estimation – BLPs

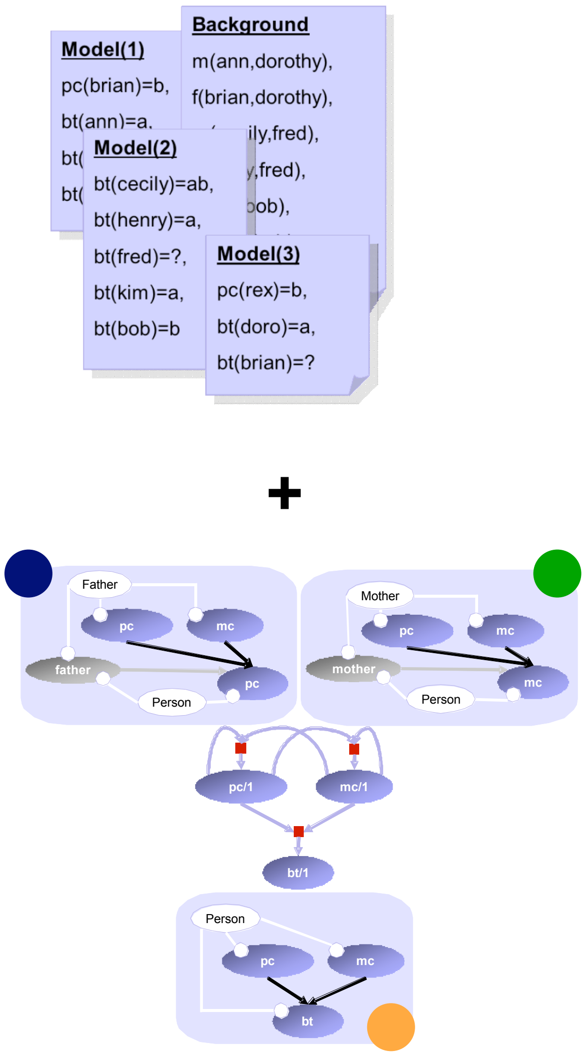


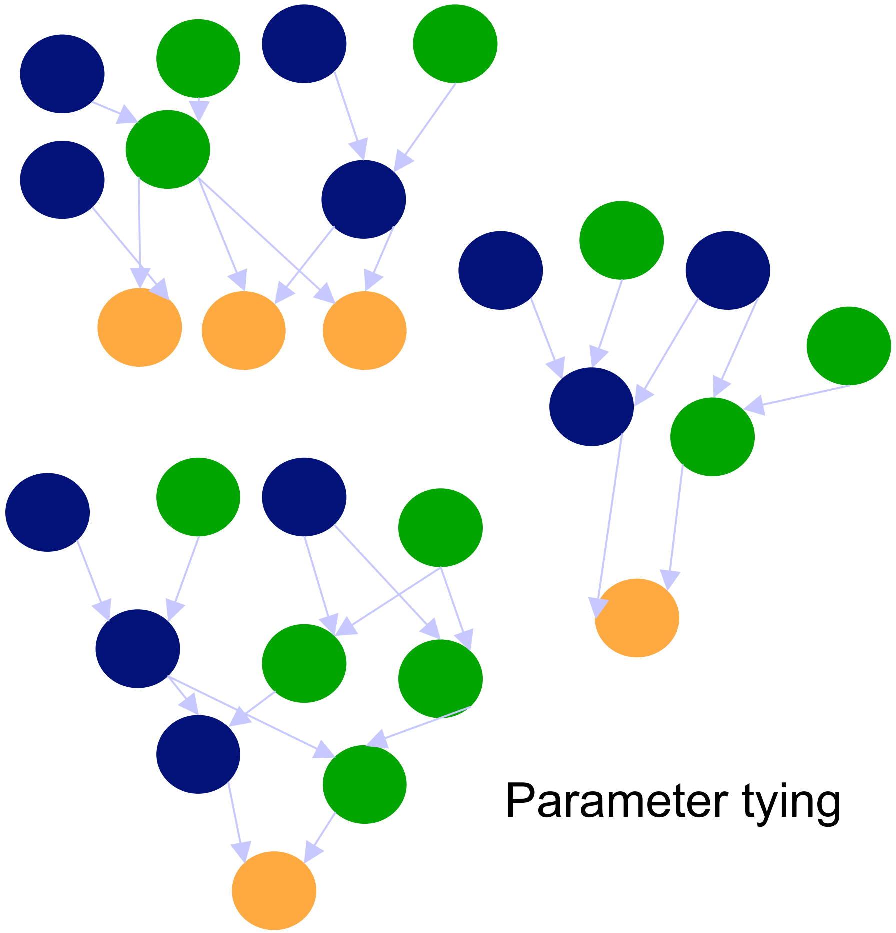
+





Parameter Estimation – BLPs



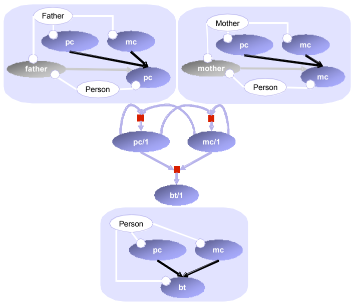


Parameter tying

EM – BLPs

EM-algorithm:
iterate until convergence

Logic Program L



Initial Parameters θ0

Current Model
(M,θk)

Model(1)
pc(brian)=b,
bt(ann)=a,
bt(
bt(

Background
m(ann,dorothy),
f(brian,dorothy),
ily,fred),
y,fred),
bob),

Model(2)
bt(cecily)=ab,
bt(henry)=a,
bt(fred)=?,
bt(kim)=a,
bt(bob)=b

Model(3)
pc(rex)=b,
bt(doro)=a,
bt(brian)=?

Expectation

Inference

Expected counts of a clause

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

Maximization

Update parameters
(ML, MAP)

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