Statistical Relational Learning

[With notes from Domingos]

Components

- We have the elements:
  - **Probability** for handling uncertainty
  - **Logic** for representing types, relations, and complex dependencies between them
  - **Inference** and **Learning** algorithms for each
- Figure out how to put them together

Markov Networks

A Markov Network is a model of the joint probability distribution over a set $\mathbf{X} = (X_1, ..., X_N) \in \mathcal{X}$ of variables. A Markov network is an undirected graph $\mathcal{G}$, with N nodes (each corresponding to one of the stochastic variables) and M edges, and a set of potential functions

$$\Phi_k \ (1 \leq k \leq K)$$

Each Φ_k is associated to a different clique in the graph.

Let $\mathbf{x} = (x_1, ..., x_N) \in \mathcal{X}$ be a set of values that the variables can take on. A Markov network is associated to the following probability distribution:

$$Pr(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{k=1}^{K} \Phi_k(\mathbf{x}^{(k)}),$$

$x^{(k)}$ is the state of the system (instantiation of the variables) in the k-th clique. Z is the partition function and can be computed as follows:

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} \prod_{k=1}^{K} \Phi_k(\mathbf{x}^{(k)})$$

Hammersley-Clifford Theorem

If Distribution is strictly positive (P(x) > 0)

And Graph encodes conditional independences

Then Distribution is product of potentials over cliques of graph

Inverse is also true.

("Markov network = Gibbs distribution")

Markov Networks

The U function is a weigthed feature of the clique --> log-linear model

$$Pr(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} e^{\sum_{k=1}^{K} w_k f_k(\mathbf{x}^{(k)})}$$

Each function $f_k(x^{(k)})$ is a feature of the k-th clique, and w_k is its weight. In a clique, there is one feature for each possible state $x^{(k)}$ of the clique.

Binary features: $f_j(x^{(k)}) \in \{0, 1\}$. Moreover, $w_k = \ln \Phi_k ((x^{(k)})$

A Markov Network is derived from a Markov Logical Network (MLN) and a set C = $\{a_1, ..., a_C\}$ of constants. A Markov Logical Network MLN = $\{(F_i, w_i) | 1 \le i \le M\}$ is a set of pairs (F_i, w_i), consisting of a logical formula F_i, belonging to a knowledge base $\mathcal{K}$, and an associated real number w_i.

The logical language is function-free.

Example - Markov Logical Network

Let $\mathcal{K}$ be a knowledge base:

F_1 : R(X, Y) : – P(x), Q(Y)

F_2 : S(X, Y) : – P(X), Q(Y)

F_3 : V (X, Y) : – Q(X), Q(Y)

F_4 : R(X, Y) : – V (X , Y)

Set of constants

C = {a,b}

Herbrand base $HB_{\mathcal{K}}$:

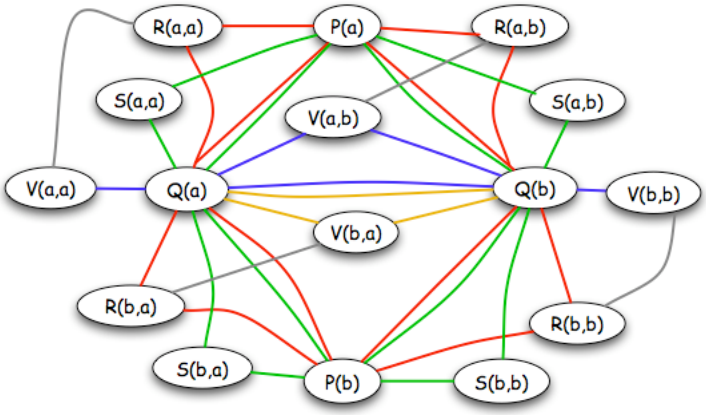
{ P(a), P(b), Q(a), Q(b), R(a, a), R(a, b), R(b, a), B(b, b), S(a, a), S(a, b), S(b, a), S(b, b), V (a, a), V (a, b), V (b, a), V (b, b) }

Markov Logical network:

MLN = $\{(F_1, w_1), (F_2, w_2), (F_3, w_3), (F_4, w_4)\}$

w = (w_1, w_2, w_3, w_4) = (2, 1.5, 0.8, 1.7)

Markov Network



Probability Distribution

Cliques$_2$ = {(Q(a), V (a, a)), (Q(b), V (b, b)), (V (a, a), R(a, a)), (V (a, b), R(a, b)), (V (b, a), R(b, a)), (V (b, b), R(b, b)) }

Of the sixteen lenght-3 cliques, only the following 10 are true groundings:

Cliques$_3$ = {(P(a), Q(a), R(a, a)), (P(a), Q(a), S(a, a)), (P(a), Q(b), R(a, b)), (P(a), Q(b), S(a, b)), (P(b), Q(a), R(b, a)), (P(b), Q(a), S(b, a)), (P(b), Q(b), R(b, b), (P(b), Q(b), S(b, b)), (Q(a), Q(b), V (a, b)), (Q(b), Q(a), V (b, a)) }

$$Pr(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} e^{\sum_{i=1}^{M} w_i n_i(\mathbf{x})} = \frac{1}{Z} \prod_{i=1}^{M} e^{w_i n_i(\mathbf{x})}$$

Z = 3.022509 · $_{10}{}^{14}$ = 3.023 · $_{10}{}^{14}$

$n_i(x)$ is the number of true groundings of formula F_i in x, or the number of features, corresponding to F_i , that equal to 1 in x.

Cliques

F_i	Formula	Features	Weigth w_i
F_1	R(X,Y) :- P(X), Q(Y)	(P(a), Q(a), R(a,a))	2.0
		(P(a), Q(b), R(a,b))	2.0
		(P(b), Q(a), R(b,a))	2.0
		(P(b), Q(b), R(b,b))	2.0
F_2	S(X,Y) :- P(X), Q(Y)	(P(a), Q(a), S(a,a))	1.5
		(P(a), Q(b), S(a,b))	1.5
		(P(b), Q(a), S(b,a))	1.5
		(P(b), Q(b), S(b,b))	1.5
F_3	V(X,Y) :- Q(X), Q(Y)	(Q(a), V(a,a))	0.8
		(Q(a), Q(b), V(a,b))	0.8
		(Q(b), Q(a), V(b,a))	0.8
		(Q(b), V(b,b))	0.8
F_4	R(X,Y) :- V(X,Y)	(R(a,a), V(a,a))	1.7
		(R(a,b), V(a,b))	1.7
		(R(b,a), V(b,a))	1.7
		(R(b,b), V(b,b))	1.7

Special cases

$$Pr(\mathbf{X}=\mathbf{x}) = \frac{1}{Z}\prod_{k=1}^{4} e^{w_k n_k(\mathbf{x})} = \frac{1}{Z}\, e^{2\, n_1(\mathbf{x})} \cdot e^{1.5\, n_2(\mathbf{x})} \cdot e^{0.8\, n_3(\mathbf{x})} \cdot e^{1.7\, n_4(\mathbf{x})}$$

If all the variables are true (let $\mathbf{x}_1$ be the corresponding world), or all the variables are false (let $\mathbf{x}_0$ be the corresponding world)

we have $n_1(\mathbf{x}_1) = n_2(\mathbf{x}_1) = n_3(\mathbf{x}_1) = n_4(\mathbf{x}_1) = 4$,

$$Pr(\mathbf{X}=\mathbf{x}_1) = \frac{1}{Z}\, e^{4\,(2+1.5+0.8+1.7)} = \frac{e^{24}}{3.023 \cdot 10^{14}} = 8.76 \cdot 10^{-5}$$

World $\mathbf{x}_2$ in which P(a) = 0, P(b) = 1, Q(a) = 0, Q(b) = 1, R(a,b) = 1, S(a,b) = 1, R(b,b) = 0, S(b,b) = 0, R(b,a) = 1, S(b,a) = 1, R(a,a) = 0, S(a,a) = 1, V(a,a) = 0, V(b,b) = 1, V(a,b) = 0, V(b,a) = 1.

In this world, we have $n_1(\mathbf{x}_2) = 4$, $n_2(\mathbf{x}_2) = 4$, $n_3(\mathbf{x}_2) = 2$, $n_4(\mathbf{x}_2) = 3$,

$$Pr(\mathbf{X}=\mathbf{x}_2) = \frac{1}{Z}\, e^{2\cdot 4+1.5\cdot 4+0.8\cdot 2+1.7\cdot 3} = \frac{e^{20.7}}{3.023 \cdot 10^{14}} = 3.23 \cdot 10^{-6}$$

Computing Probabilities

- P(Formula|MLN,C) = ?
- MCMC: Sample worlds, check formula holds
- P(Formula1|Formula2,MLN,C) = ?
- If Formula2 = Conjunction of ground atoms
  - First construct min subset of network necessary to answer query (generalization of KBMC)
  - Then apply MCMC (or other)
- Can also do lifted inference [Braz et al, 2005]

Inference

What is the probability that a ground formula F_1 is true, knowing than another ground formula F_2 is true?

$$Pr(F_1|F_2,\mathbf{MLN},\mathbf{C}) = \frac{Pr(F_1 \wedge F_2|\mathbf{MLN},\mathbf{C})}{Pr(F_2|\mathbf{MLN},\mathbf{C})} = \frac{\sum_{\mathbf{x}\in X_{F_1}\cap X_{F_2}} Pr(\mathbf{X}=\mathbf{x}|\mathbf{MLN},\mathbf{C})}{\sum_{\mathbf{x}\in X_{F_2}} Pr(\mathbf{X}=\mathbf{x}|\mathbf{MLN},\mathbf{C})}$$

$F_1 \equiv$ [R(a, b) : – P(a), Q(b)] $F_2 \equiv$ [V (b, a) : – Q(a), Q(b)].

F_2 is true when either Q(a) = 0, or Q(b) = 0, or Q(a) = 1, Q(b) = 1, V (b, a) = 1

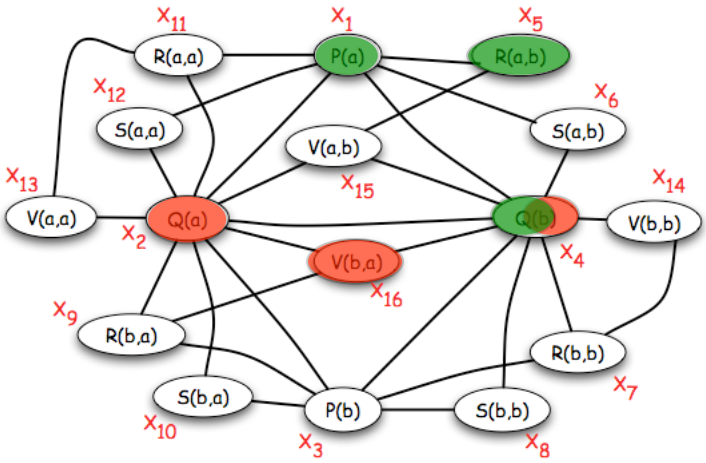
F_1 is true when either P(a) = 0, or Q(b) = 0, or P(a) = 1, Q(b) = 1, R(a, b) = 1

$Pr(F_2 \mid MLN, C) = 0.984$

$Pr(F_1 \wedge F_2 \mid MLN, C) = 0.632$

$Pr(F_1 \mid F_2, MLN, C) = 0.642$

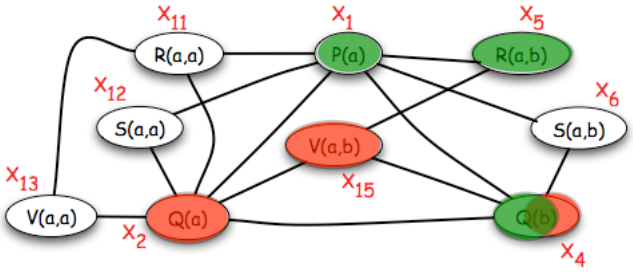
Query and Evidence



Ground Network Construction

network ← Ø
queue ← query nodes
repeat
  node ← front(*queue*)
  remove *node* from *queue*
  add *node* to *network*
  if *node* not in evidence **then**
    add neighbors(*node*) to queue
until queue = Ø

Reduced Network



$Pr(X' = x') \equiv Pr(X_1 = x_1 , X_2 = x_2 , X_4 = x_4 , X_5 = x_5 , X_6 = x_6 , X_{11} = x_{11} ,$

$X_{12} = x_{12} , X_{13} = x_{13} , X_{15} = x_{15}) =$

$= (1/ Z') \, e^{2 \, n1'(\mathbf{x}')} \, e^{1.5 \, n2'(\mathbf{x}')} \, e^{0.8 \, n3'(\mathbf{x}')} \, e^{1.7 \, n4'(\mathbf{x}')}$

$Z' = 1.287 \, 10^8$

Markov Blanket

Pprobability that X is true, given the state s of $\mathcal{B}(X)$, can be estimated through Gibb sampling:

$$Pr(X = 1 | \mathcal{B}(X) = \mathbf{s}) = \frac{e^{\sum_{i=1}^{M} w_i \cdot n_i''(X=1, \mathcal{B}(X)=\mathbf{s})}}{e^{\sum_{i=1}^{M} w_i \cdot n_i''(X=1, \mathcal{B}(X)=\mathbf{s})} + e^{\sum_{i=1}^{M} w_i \cdot n_i''(X=0, \mathcal{B}(X)=\mathbf{s})}}$$

Node X_5 = R(a, b). Its Markov blanket is the set B (X5) = $\{X_1 , X_{15} , X_4\}$.

X_5 belongs to the cliques (X_1, X_4, X_5) of rule F_1, and

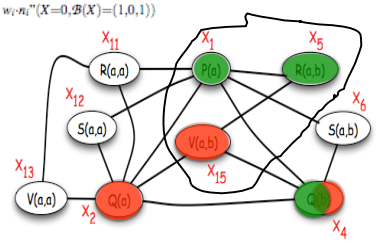
(X_5, X_{15}) of rule F_4. Let s = (1, 0, 1) be the state of B(X_5):

$$Pr(X_5 = 1 | \mathcal{B}(X_5) = (1,0,1)) = \frac{e^{\sum_{i=1}^{4} w_i \cdot n_i''(X=1, \mathcal{B}(X)=(1,0,1))}}{e^{\sum_{i=1}^{4} w_i \cdot n_i''(X=1, \mathcal{B}(X)=(1,0,1))} + e^{\sum_{i=1}^{4} w_i \cdot n_i''(X=0, \mathcal{B}(X)=(1,0,1))}}$$

$X_5 = 1$ --> $n_1 = 1$, $n_2 = 0$, $n_3 = 0$, $n_4 = 1$

$X_5 = 0$ --> $n_1 = 0$, $n_2 = 0$, $n_3 = 0$, $n_4 = 1$

$Pr(X_5 = 1 \, | \mathcal{B}(X_5) = (1, 0, 1)) = 0.881$



MCMC: Gibbs Sampling

state ← random truth assignment
for *i* ← 1 **to** *num-samples* **do**
  for each variable *x*
    sample *x* according to P(*x*|*neighbors*(*x*))
    state ← *state* with new value of *x*
P(*F*) ← fraction of states in which *F* is true

Learning Markov Networks

- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (features)

Weight Learning

- Parameter tying: Groundings of same clause

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

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

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

- Generative learning: Pseudo-likelihood
- Discriminative learning: Cond. likelihood, use MC-SAT or MaxWalkSAT for inference

Generative Weight Learning

- Maximize likelihood or posterior probability
- Numerical optimization (gradient or 2nd order)
- No local maxima

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

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

Expected no. times feature *i* is true according to model

- Requires inference at each step (slow!)

Pseudo-Likelihood

$$PL(x) \equiv \prod_i P(x_i \mid neighbors(x_i))$$

- Likelihood of each variable given its neighbors in the data
- Does not require inference at each step
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains

Discriminative Weight Learning

- Maximize conditional likelihood of query (y) given evidence (x)

$$\frac{\partial}{\partial w_i} \log P_w(y \mid x) = n_i(x, y) - E_w[n_i(x, y)]$$

No. of true groundings of clause *i* in data

Expected no. true groundings according to model

- Approximate expected counts by counts in MAP state of y given x

Structure Learning

- Start with atomic features
- Greedily conjoin features to improve score
- Problem: Need to reestimate weights for each new candidate
- Approximation: Keep weights of previous features constant

Example: Friends & Smokers

Smoking causes cancer.

Friends have similar smoking habits.

Example: Friends & Smokers

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

Example: Friends & Smokers

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

Example: Friends & Smokers

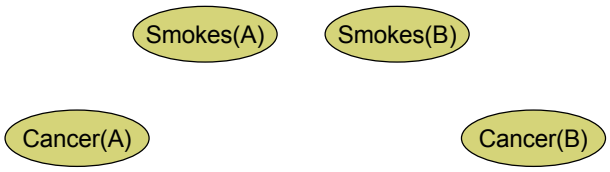
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)

Example: Friends & Smokers

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

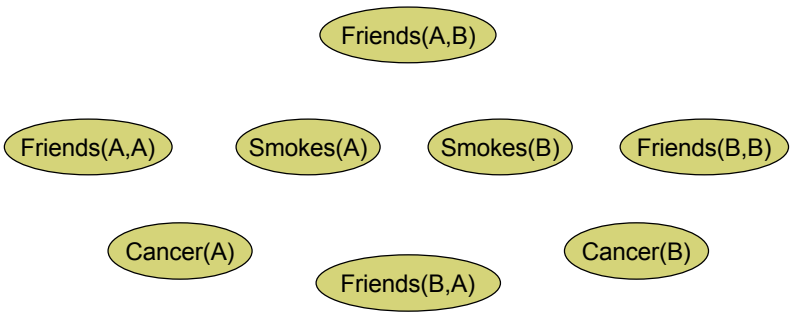
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

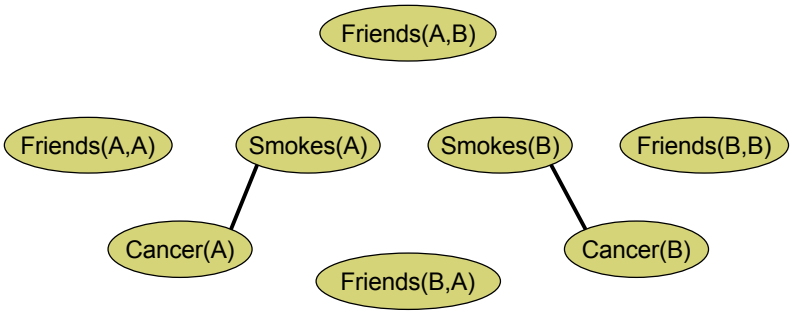
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

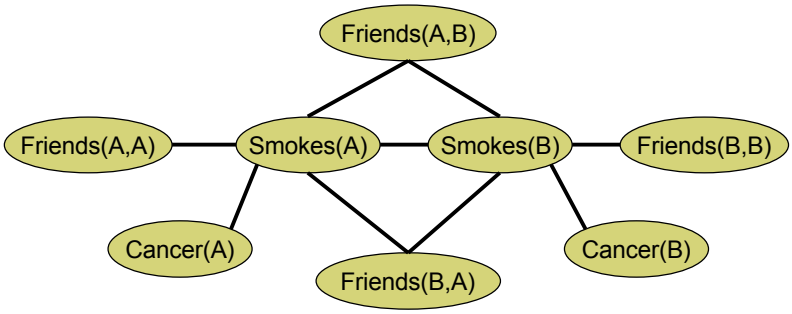
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

Two constants: **Anna** (A) and **Bob** (B)



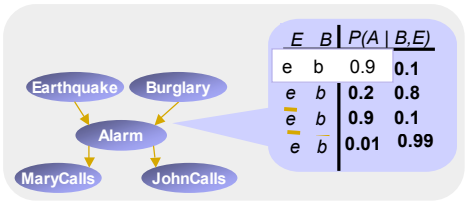
Markov Nets vs. Bayes Nets

Property	Markov Nets	Bayes Nets
Form	Prod. potentials	Prod. potentials
Potentials	Arbitrary	Cond. probabilities
Cycles	Allowed	Forbidden
Partition func.	Z = ?	Z = 1
Indep. check	Graph separation	D-separation
Indep. props.	Some	Some
Inference	MCMC, BP, etc.	Convert to Markov

Probabilistic Logic Programs (PLPs)

[Haddawy, Ngo]

- Atoms = set of similar RVs
- First arguments = RV
- Last argument = state
- Clause = CPD entry



Probability of being true | RV | State

- Probability distribution over Herbrand interpretations

```
0.1 : burglary(true).        0.9 : burglary(false).
0.01 : earthquake(true).     0.99: earthquake(false).
0.9 : alarm(true) :- burglary(true), earthquake(true).
...
                    burglary(true), burglary(false).
                    ue); burglary(false) :- true.
                    :- earthquake(true), earthquake(false).
...
```

burglary(true) and burglary(false) true in the same interpretation ?

Probabilistic Logic Programs (PLPs)

[Haddawy, Ngo]

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

Context

```
1.0 : mc(P,a) :- mother(M,P), pc(M,a),mc(M,a).
0.0 : mc(P,b) :- mother(M,P), pc(M,a),mc(M,a).
...
0.5 : pc(P,a) :- father(F,P), pc(F,0),mc(F,a).
0.5 : pc(P,0) :- father(F,P), pc(F,0),mc(F,a).
...
1.0 : bt(P,a) :- mc(P,aa),pc(P,aa)
```

Probabilities

```
false :- pc(P,a),pc(P,b), pc(P,0).
pc(P,a);pc(P,b); pc(P,0)´:- person(P).
...
```

Constraints

Probabilistic Logic Programs (PLPs)

[Haddawy, Ngo]

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

Qualitative Part

Quantitative Part

RV State

```
1.0 : mc(P,a) :- mother(M,P), pc(M,a),mc(M,a).
0.0 : mc(P,b) :- mother(M,P), pc(M,a),mc(M,a).
...
0.5 : pc(P,a) :- father(F,P), pc(F,0),mc(F,a).
0.5 : pc(P,0) :- father(F,P), pc(F,0),mc(F,a).
...
1.0 : bt(P,a) :- mc(P,aa),pc(P,aa)
```

Dependency

```
false :- pc(P,a),pc(P,b), pc(P,0).
pc(P,a);pc(P,b); pc(P,0) :- person(P).
...
```

Probabilistic Logic Programs (PLPs)

[Haddawy, Ngo]

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

Qualitative Part

Quantitative Part

RV State

```
1.0 : mc(P,a) :- mother(M,P), pc(M,a),mc(M,a).
0.0 : mc(P,b) :- mother(M,P), pc(M,a),mc(M,a).
...
0.5 : pc(P,a) :- father(F,P), pc(F,0),mc(F,a).
0.5 : pc(P,0) :- father(F,P), pc(F,0),mc(F,a).
...
1.0 : bt(P,a) :- mc(P,aa),pc(P,aa)
```

Dependency

Variable Binding

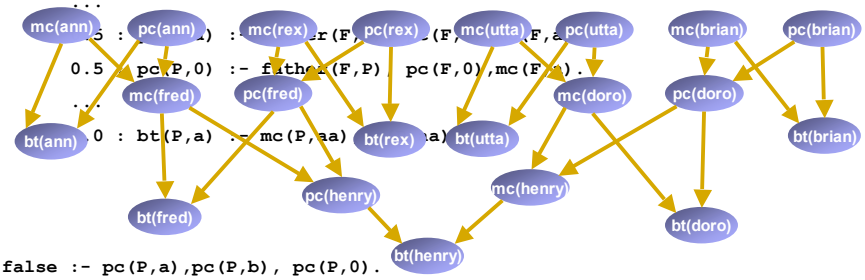
```
false :- pc(P,a),pc(P,b), pc(P,0).
pc(P,a);pc(P,b); pc(P,0) :- person(P).
...
```

Probabilistic Logic Programs (PLPs)

[Haddawy, Ngo]

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

     1.0 : mc(P,a) :- mother(M,P), pc(M,a),mc(M,a).
     0.0 : mc(P,b) :- mother(M,P), pc(M,a),mc(M,a).
     ...
     0.5 : pc(P,a) :- father(F,P), pc(F,a),mc(F,a).
     0.5 : pc(P,0) :- father(F,P), pc(F,0),mc(F,a).
     ...
     1.0 : bt(P,a) :- mc(P,a).
```




```
false :- pc(P,a),pc(P,b), pc(P,0).
pc(P,a);pc(P,b); pc(P,0) :- person(P).
...
```

Probabilistic Logic Programs (PLPs)

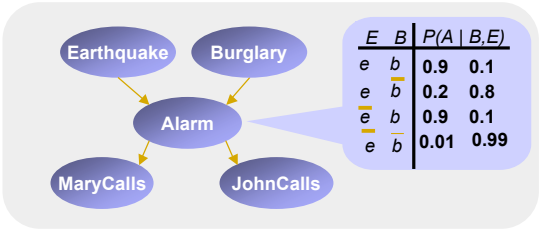
[Haddawy, Ngo]

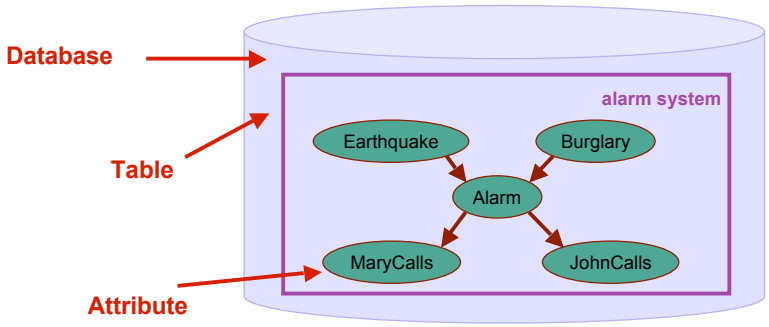
- Unique probability distribution over Herbrand interpretations
  - finite branching factor, finite proofs, no self-dependency
- Atoms = States
- Integrity constraints encode mutually excl. states
- Functors
- BN used to do inference
- Turing-complete programming language
- BNs, HMMs, DBNs, SCFGs, ...
- No learning

Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]

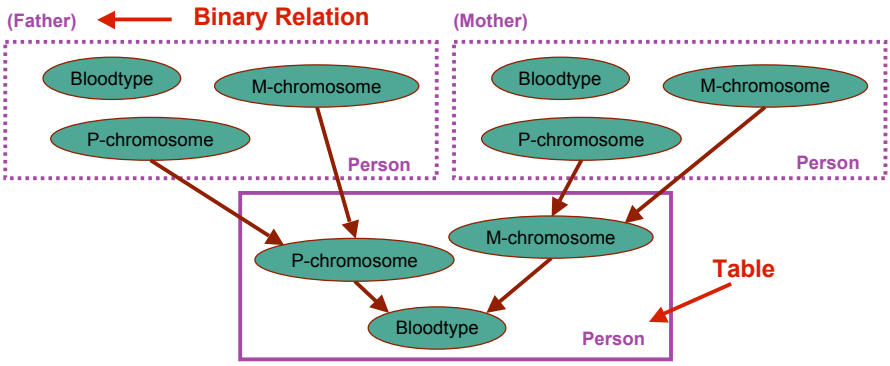
- Database theory
- Entity-Relationship Models
  - Attributes = RV





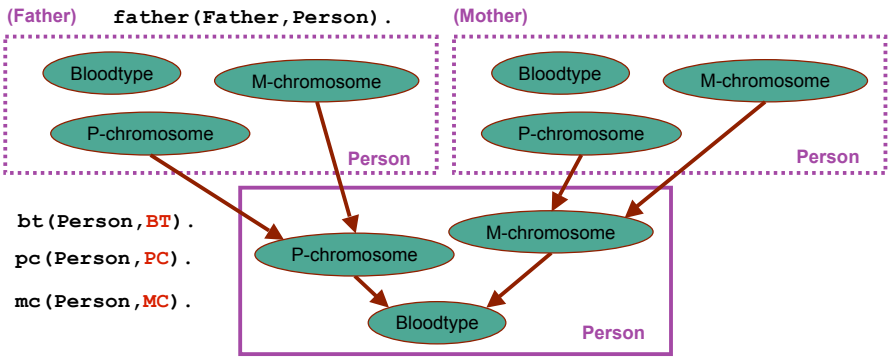
Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]



Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]



Dependencies :

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

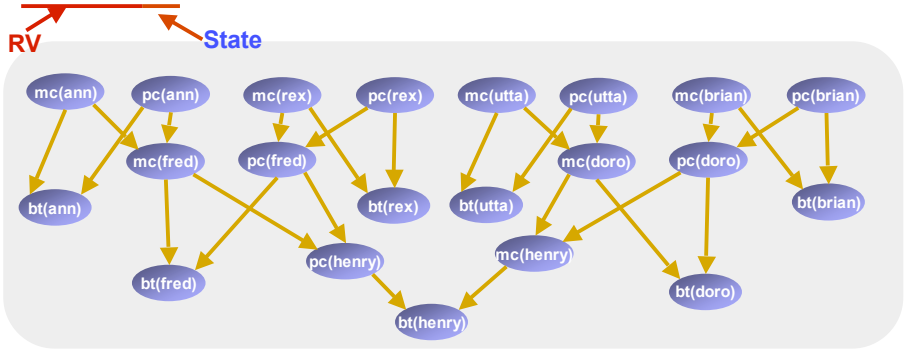
View :

```
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).
```



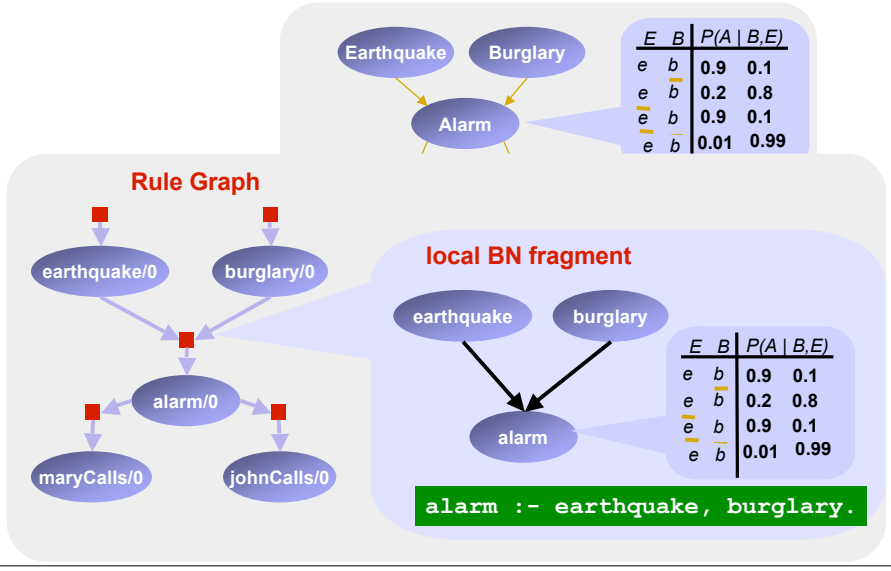
Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]

- Datalog
- Unique Probability Distribution over finite Herbrand interpretations
  - No self-dependency
- Discrete and continuous RV
- BN used to do inference
- Highlight Graphical Representation
- BNs
- Learning

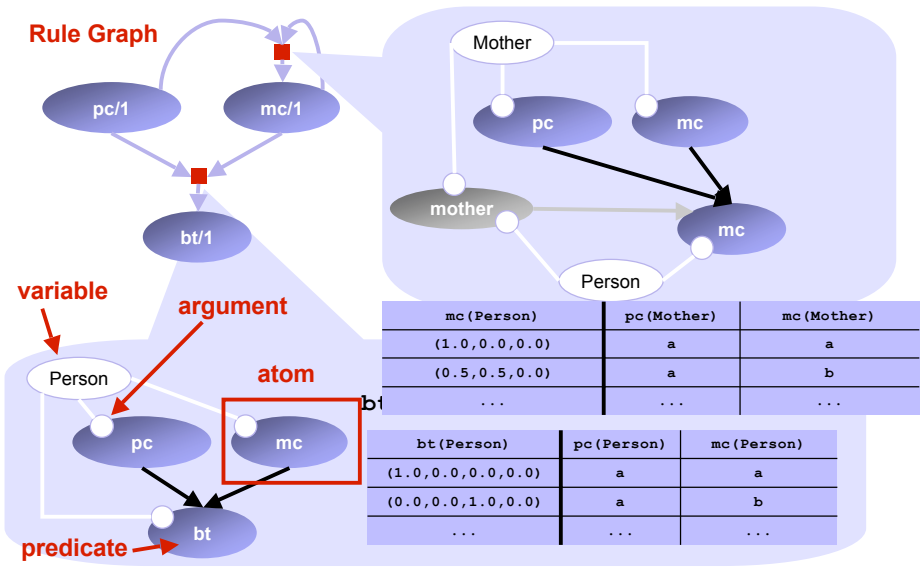
Bayesian Logic Programs (BLPs)

[Kersting, De Raedt]



Bayesian Logic Programs (BLPs)

[Kersting, De Raedt]

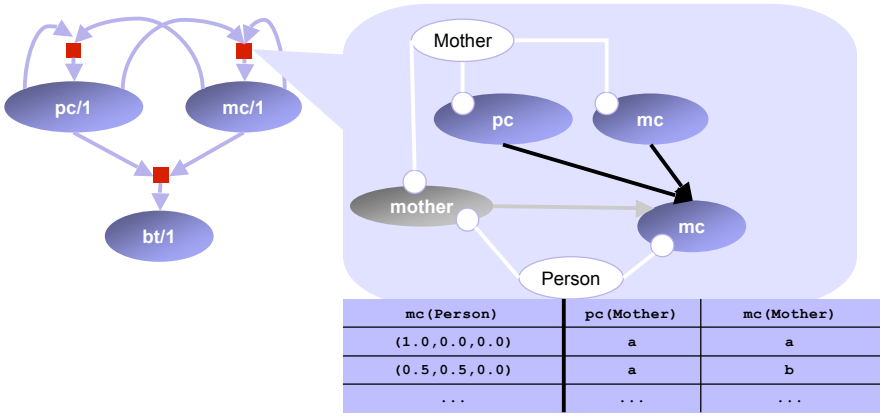


mc(Person)	pc(Mother)	mc(Mother)
(1.0,0.0,0.0)	a	a
(0.5,0.5,0.0)	a	b
...	...	...

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
...	...	...

Bayesian Logic Programs (BLPs)

[Kersting, De Raedt]



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

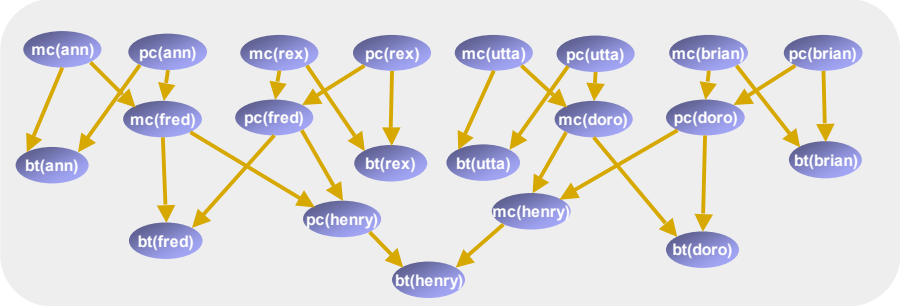
```
mc(Person) | mother(Mother,Person), pc(Mother),mc(Mother).
pc(Person) | father(Father,Person), pc(Father),mc(Father).
bt(Person) | pc(Person),mc(Person).
```

Bayesian Logic Programs (BLPs)

[Kersting, De Raedt]

```
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).
```

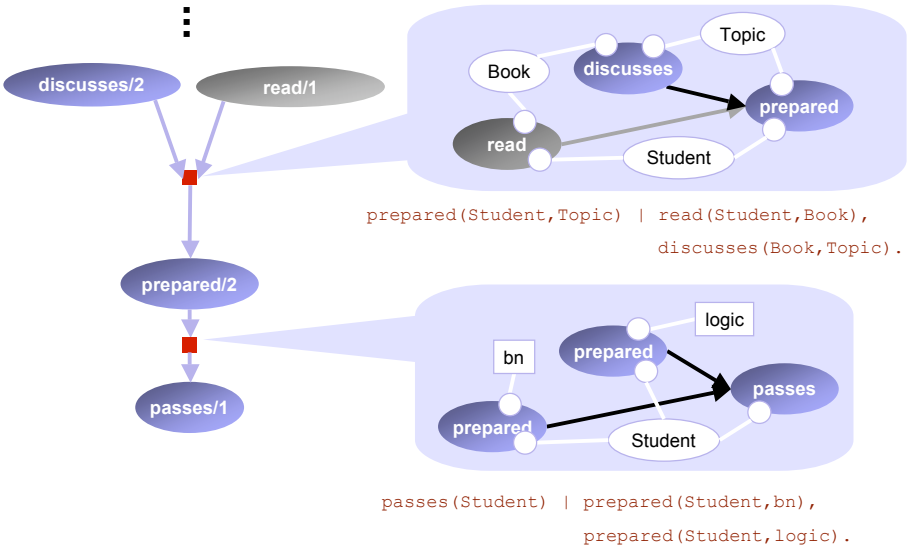


Bayesian Logic Programs (BLPs)

[Kersting, De Raedt]

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

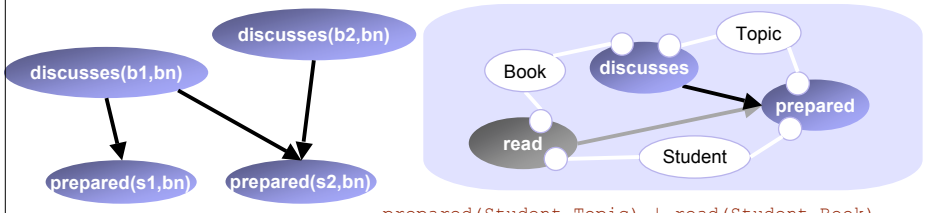
Combining Partial Knowledge




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

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

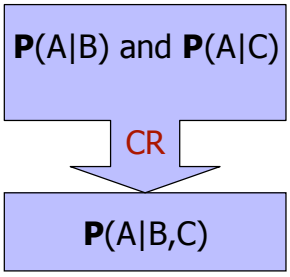
Combining Partial Knowledge

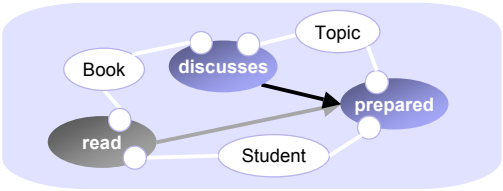



```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

- variable #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

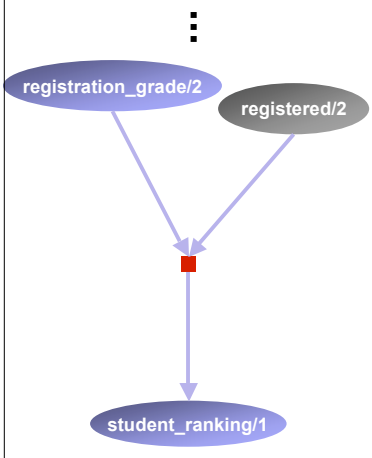





```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

- 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, regression, ...

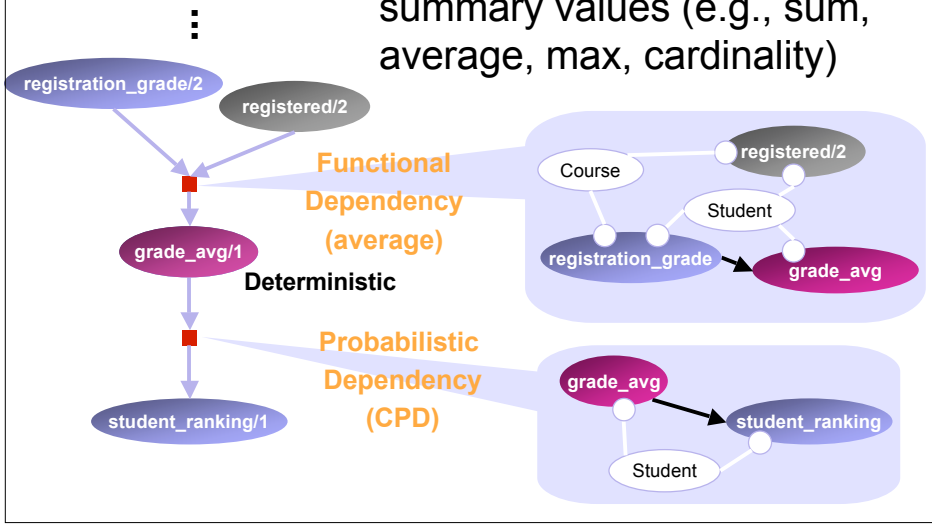
Aggregates



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)



Stochastic Relational Models (SRMs)

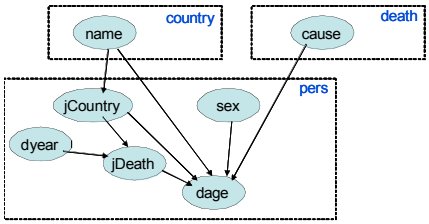
- Type I, i.e., frequencies in databases
- Probability that a select-join query succeeds:

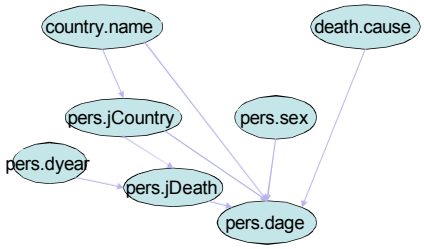
$$P_D(I_Q) = \frac{|\bowtie_Q (\sigma_Q(R_1 \times ... \times R_k))|}{|R_1| \times ... \times |R_k|}$$

- Independently sample tuples ri from Ri; select as values for Ai the values r.Ai

Stochastic Relational Models (SRMs)

WHO Mortality Database

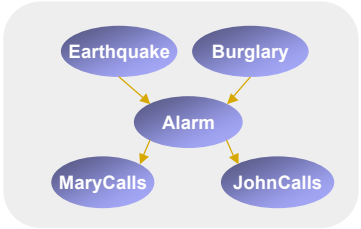




query(pers:dage='1-4y') = 0.00201
query(pers:dage='25-29y') = $7.1*10^{-5}$
query(pers:dage='75-79y') = 0.12
query(pers:dage='85-89y') = 0.176

query(pers:dage=,75-79y',death:cause=k) = 0.012
query(pers:dage=,85-89y',death:cause=k) = 0.0012
query(pers:dage=,75-79y',death:cause=r) = 0.02
query(pers:dage=,85-89y',death:cause=r) = 0.114

What is the data about? – Model Theoretic



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

What is the data about? – Model Theoretic

Data case:

- RV specified = (partial) Herbrand interpretation
- Aking to „learning from interpretations“ (ILP)

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

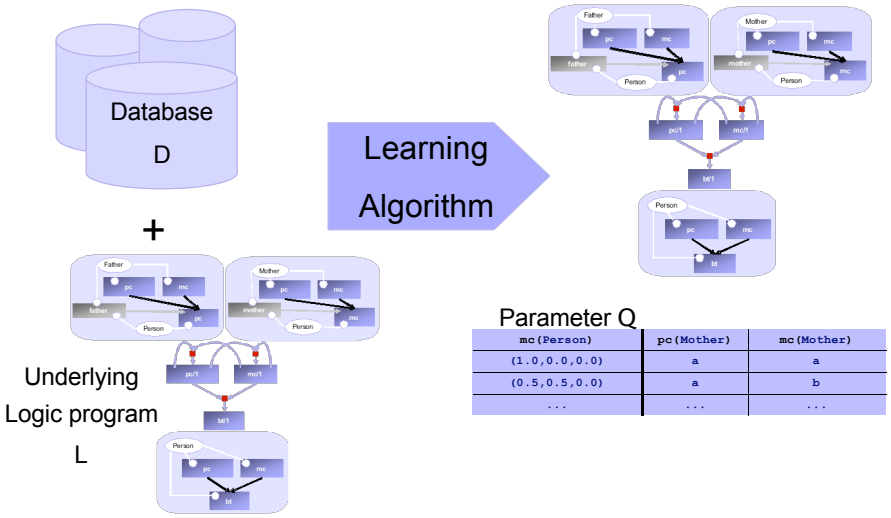
Background
m(ann,dorothy),
f(brian,dorothy),
m(cecily,fred),
f(henry,fred),
f(fred,bob),
m(kim,bob),
...

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

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

Bloodtype example

Parameter Estimation – Model Theoretic



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

Parameter Estimation – Model Theoretic

- Estimate the CPD q entries that best fit the data
- „Best fit“: ML parameters q^*

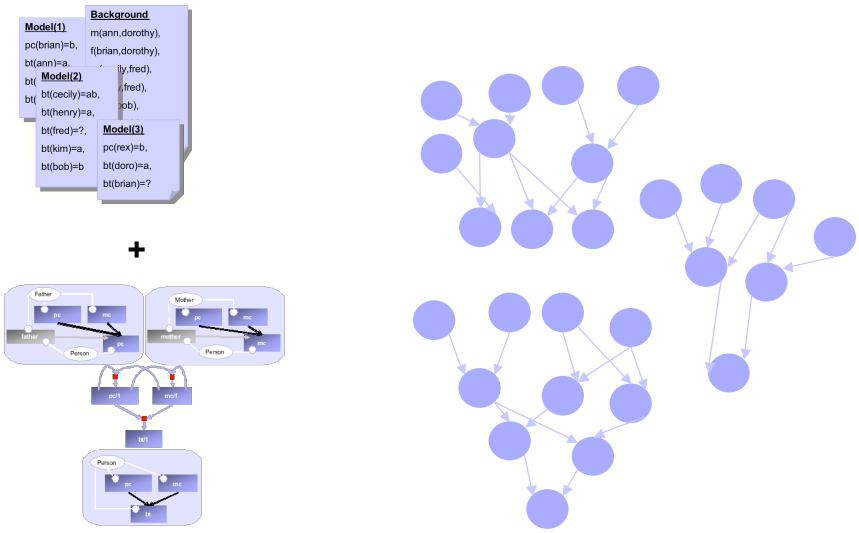
  $q^* = \text{argmax}_q P(\text{data} \mid \text{logic program}, q)$

  $= \text{argmax}_q \log P(\text{data} \mid \text{logic program}, q)$
- Reduces to problem within Bayesian networks:

  given structure,

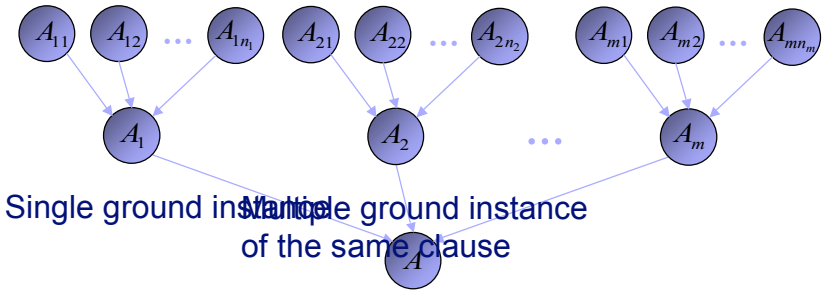
  partially observed random varianbles

Parameter Estimation – Model Theoretic



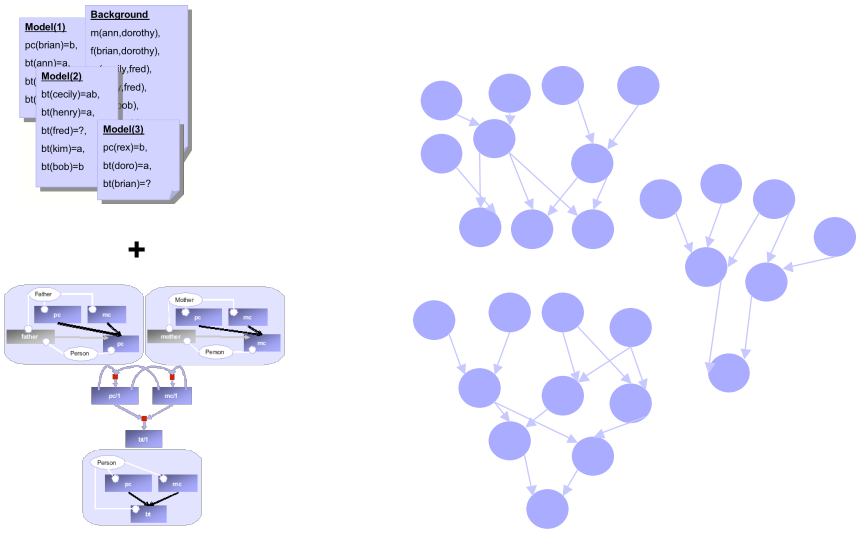
Excourse: Decomposable CRs

- Parameters of the clauses and not of the support network.

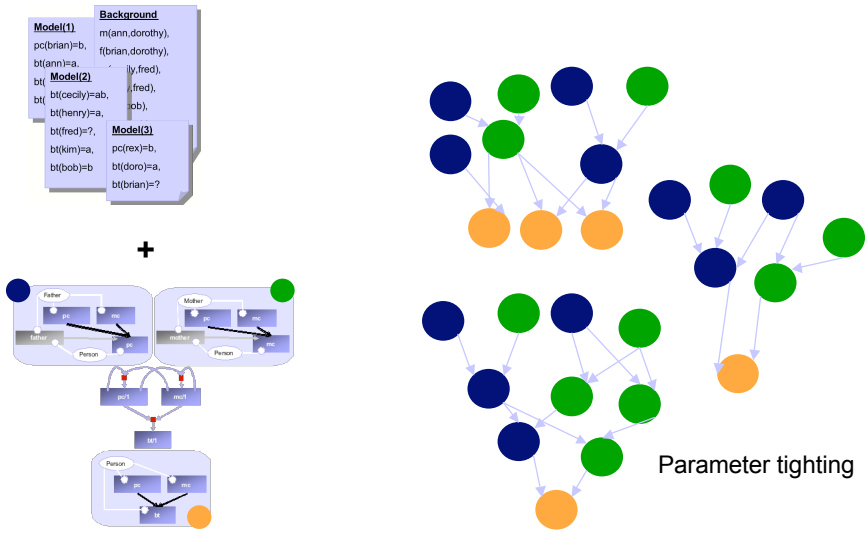


Deterministic CPD for Combining Rule

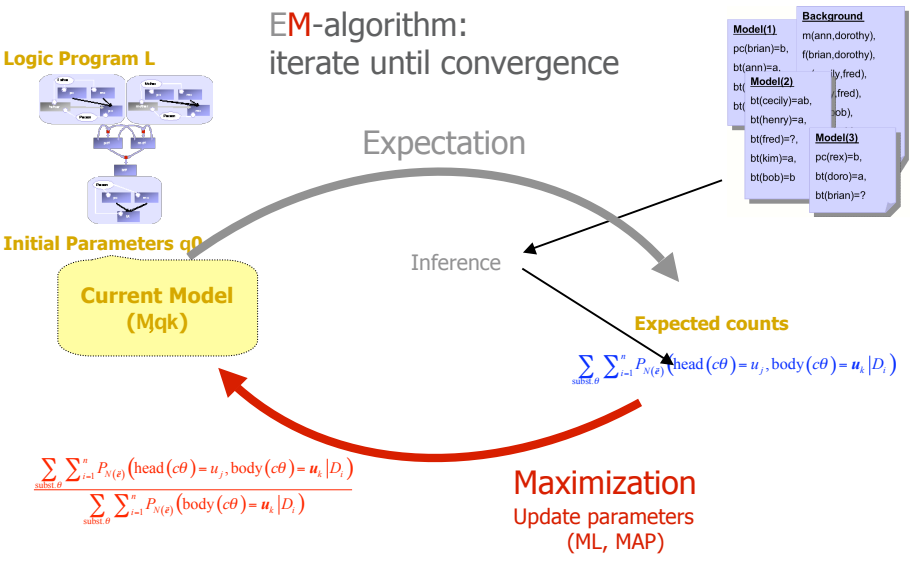
Parameter Estimation – Model Theoretic



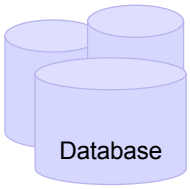
Parameter Estimation – Model Theoretic



EM – Model Theoretic



Model Selection – Model Theoretic



+

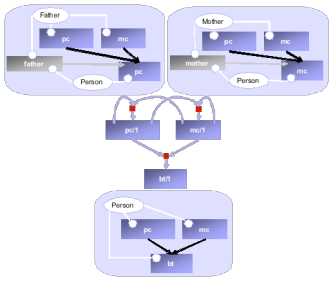
Language:

Bayesian bt/1, pc/1,mc/1

Background Knowledge:

Logical mother/2, father/2

Learning Algorithm



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

Model Selection – Model Theoretic

- 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* = argmax_M score(M,D).
- *Highlights*
  - *Refinement operators*
  - *Background knowledge*
  - *Language biase*
  - *Search bias*

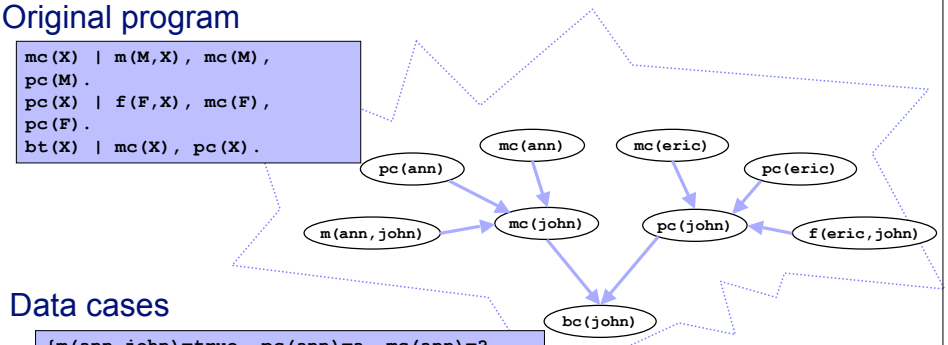
Refinement Operators

- Add a fact, delete a fact or refine an existing clause:
  - Specialization:
    - Add atom
    - apply a substitution { X / Y } where X,Y already appear in atom
    - apply a substitution { X / f(Y1, … , Yn)} where Yi new variables
    - apply a substitution {X / c } where c is a constant
  - Generalization:
    - delete atom
    - turn ‘term’ into variable
      - p(a,f(b)) becomes p(X,f(b)) or p(a,f(X))
      - p(a,a) becomes p(X,X) or p(a,X) or p(X,a)
    - replace two occurences of variable X into X1 and X2
      - p(X,X) becomes p(X1,X2)

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) = ? }

...
```

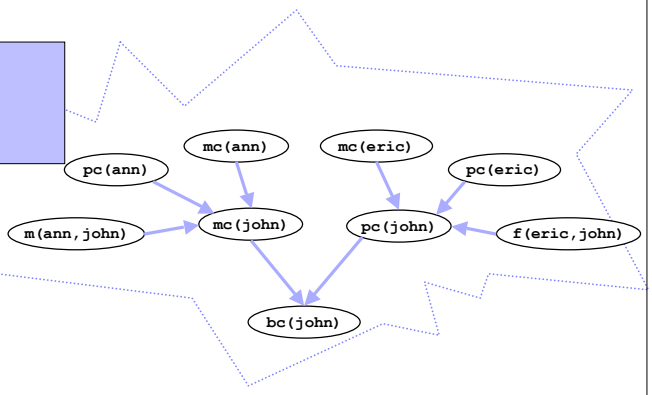
Example

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).
```



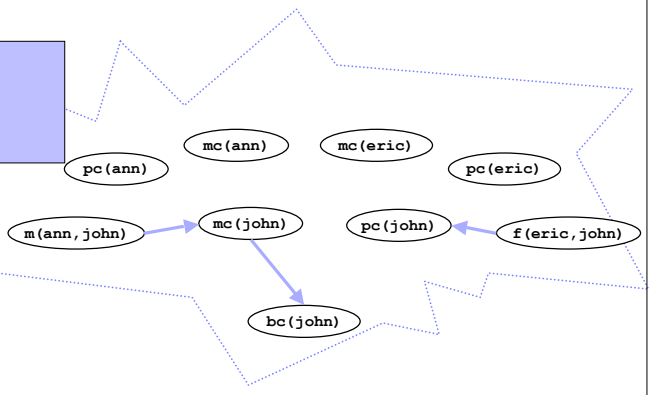
Example

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

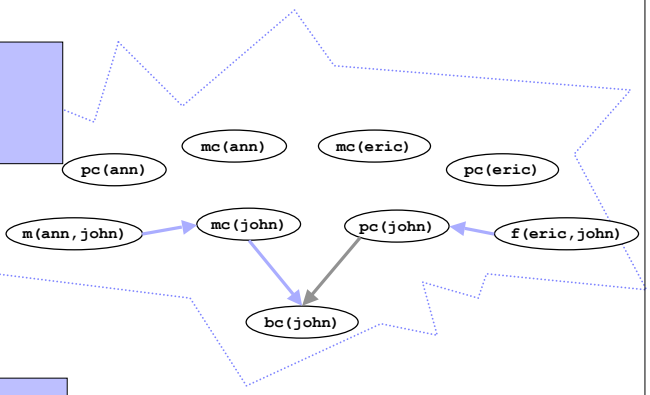
```
mc(X) | m(M,X), mc(M).
pc(M).
pc(X) | f(F,X), mc(F).
pc(F).
bt(X) | mc(X), pc(X).
```

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

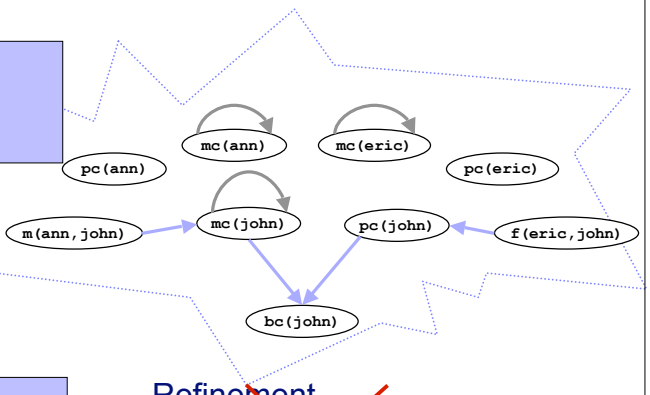
```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```

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).
```



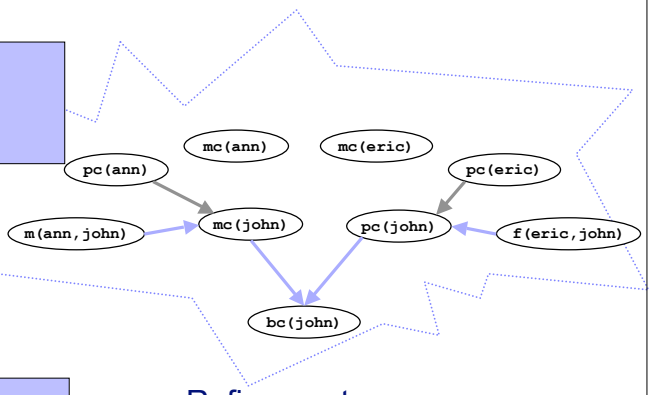
Example

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).
```

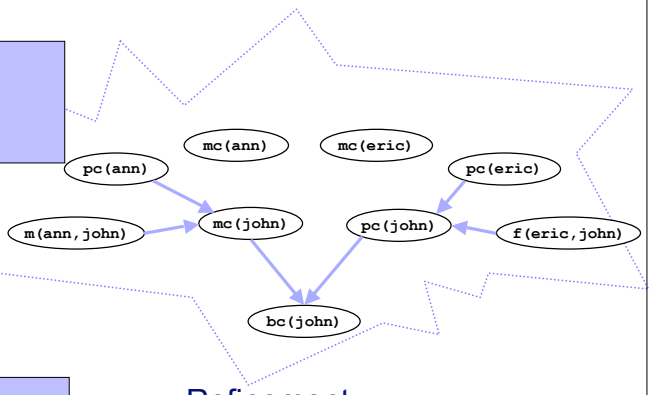
Example

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).
```