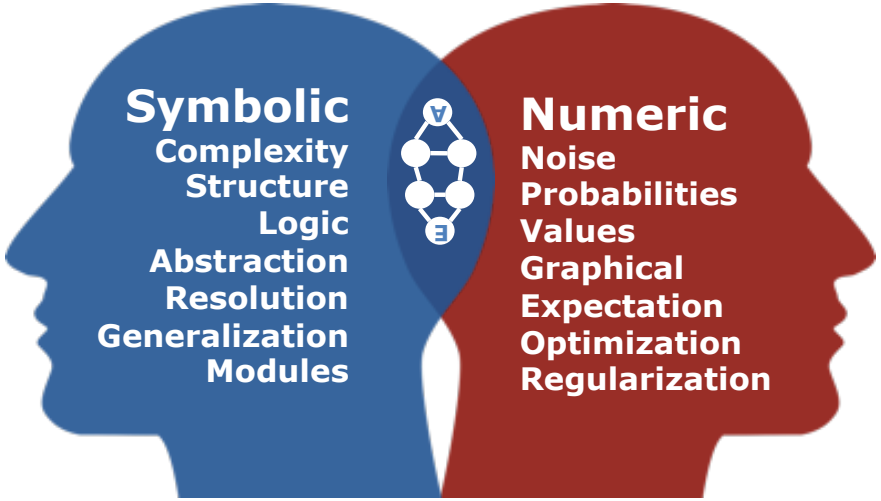
Statistical Relational AI

(Deep) Relational Probabilistic Models - Overview, Problog, and MLNs

TECHNISCHE UNIVERSITÄT DARMSTADT



Kristian Kersting

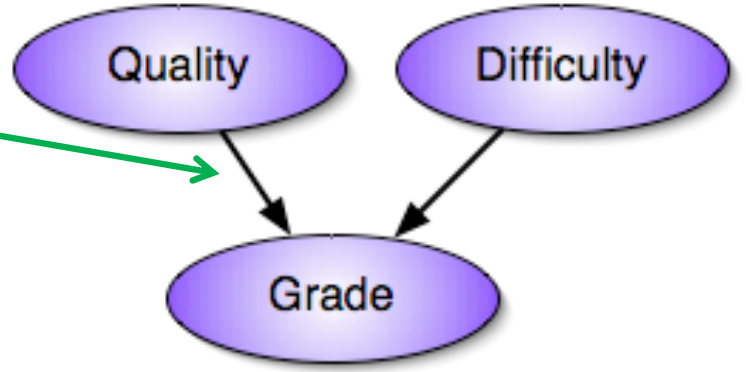
Thanks to Vincent Conitzer, Rina Dechter, Luc De Raedt, Pedro Domingos, Peter Flach, Dieter Fensel, Florian Ficher, Vibhav Gogate, Carlos Guestrin, Daphen Koller, Nir Friedman, Ray Mooney, Sriraam Natarajan, David Poole, Fabrizio Riguzzi, Dan Suciu, Guy van den Broeck, and many others for making their slides publically available

Let's consider a simple relational domain: Reviewing Papers

TECHNISCHE UNIVERSITÄT DARMSTADT

- The grade of a paper at a conference depends on the paper's quality and the difficulty of the conference.
  - **Good papers may get A's at easy conferences**
  - **Good papers may get D's at top conference**
  - **Weak papers may get B's at good conferences**
  - ...

(Reviewing) Bayesian Network



P(Qual)		
low	middle	high
0.3	0.5	0.2

P(Diff)		
low	middle	high
0.2	0.3	0.5

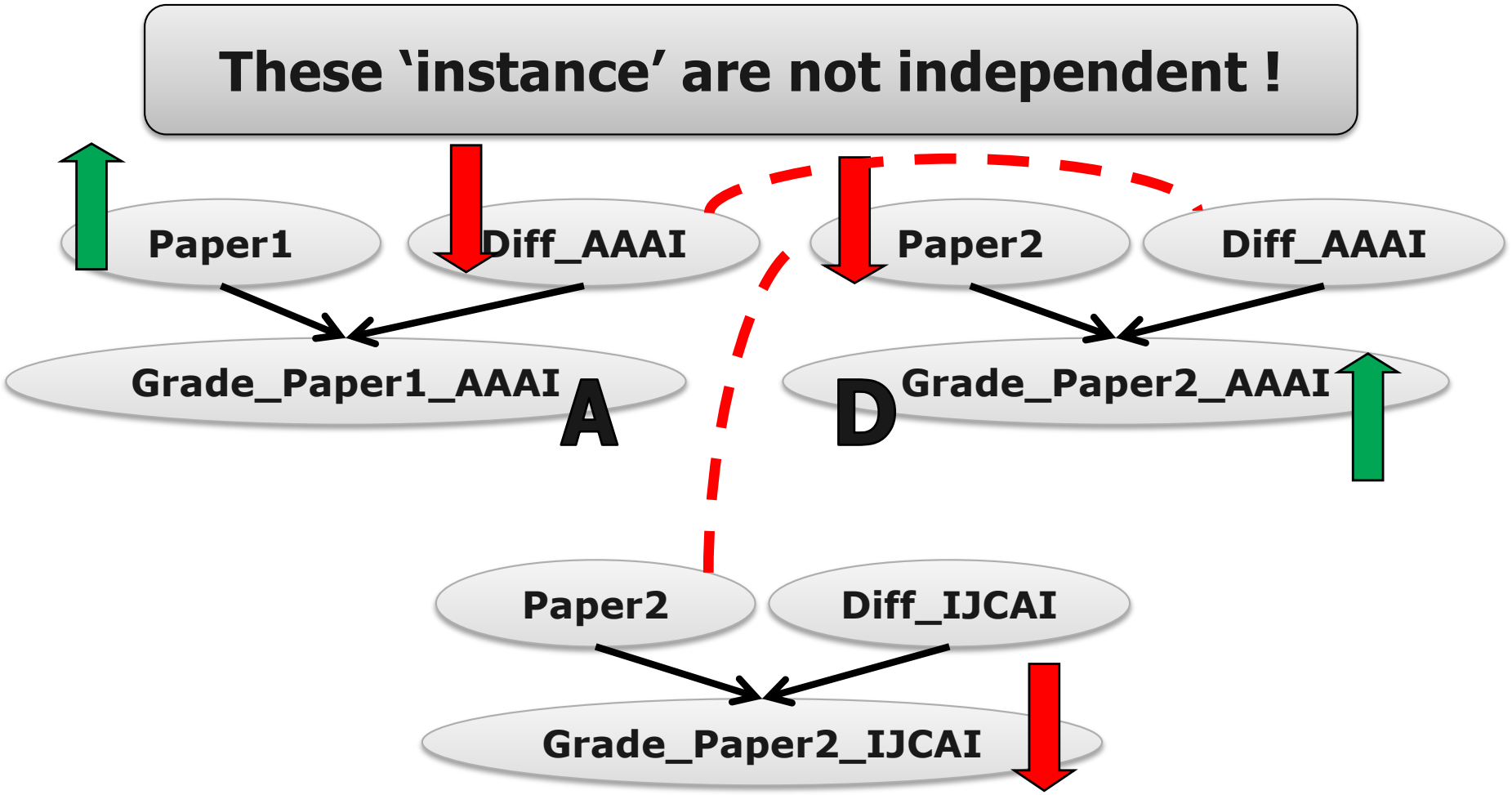
$$P(X_1,\ldots,X_n)=\prod_{i=1}^{n}P(X_i\,|\,X_{i-1},\ldots,X_1)$$

Qual	Diff	P(Grade)		
		c	b	a
low	low	0.2	0.5	0.3
low	middle	0.1	0.7	0.2
...				

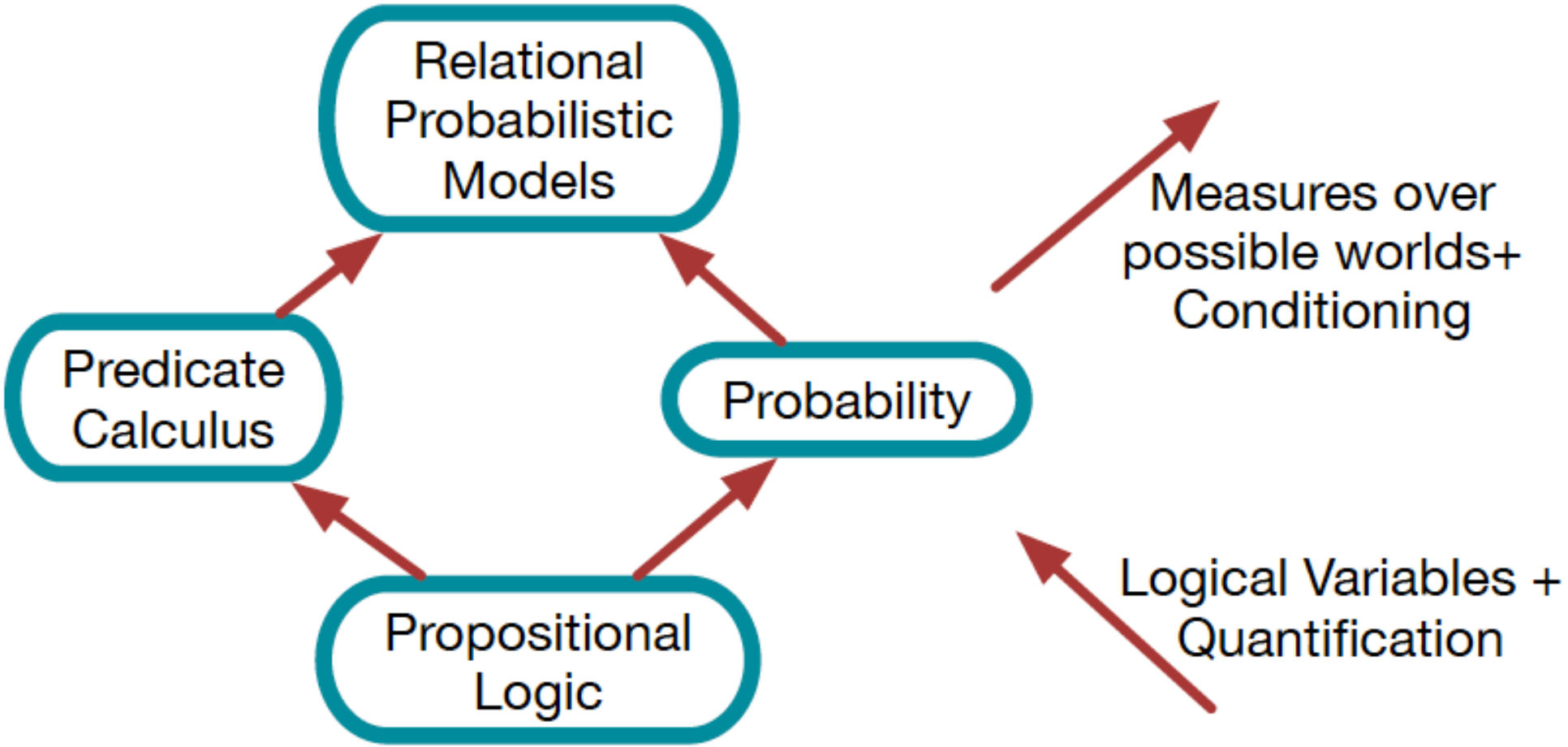
The real world, however, ...

[inspired by Friedman and Koller]

... has **inter-related objects**



Therefore we want to combine probabilities and logic



But first let us clarify how people view relations in this context

SO WHAT ARE RELATIONS?

What are Relations?

TECHNISCHE UNIVERSITÄT DARMSTADT

There are several types of relations and in turn there are several views on what (statitical) relational learning is

1. **Relations provide additional correlations/ regularization**
   - If two words occure frequently in the same context (page, paragraph, sentence, …) then there must be some semantic relation between them
2. **Often extensional (data) only, for one relation**
   - Covariance function, distance functions, kernel functions, graphs, tensors, random walks with restarts…

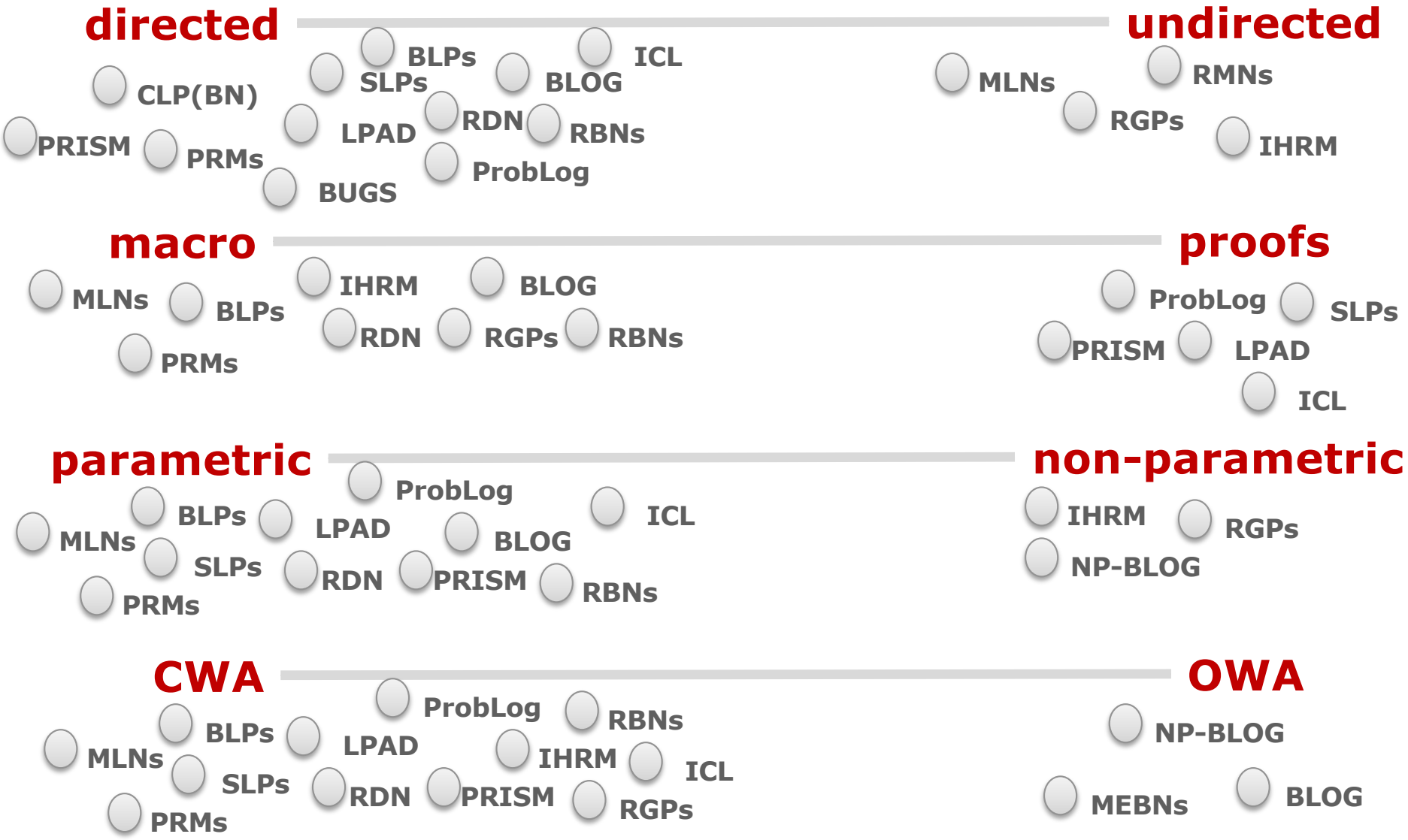
What are Relations?

TECHNISCHE UNIVERSITÄT DARMSTADT

3. **Relations are symmetries/redundancies in the model**
   - E.g. lifted inference based on bisimulation
4. **Hypergraph representations of data**
   - Multiple (extensional) relations
   - Random walks with restarts as similarity measure, produce path features, tensor-based embeddings
5. **Full-fledged relational (or logical) knoweldge as considered in this tutorial**
   - Multiple (often typed) relations
   - Intensional formulas (often Horn clauses)
     ancestor(X,Z) ^ parent(Z,Y) ⇒ ancestor(X,Y)

Different approaches exists that can be classified along several dimensions

TECHNISCHE UNIVERSITÄT DARMSTADT



And actually SRL spans the whole AI spectrum

TECHNISCHE UNIVERSITÄT DARMSTADT

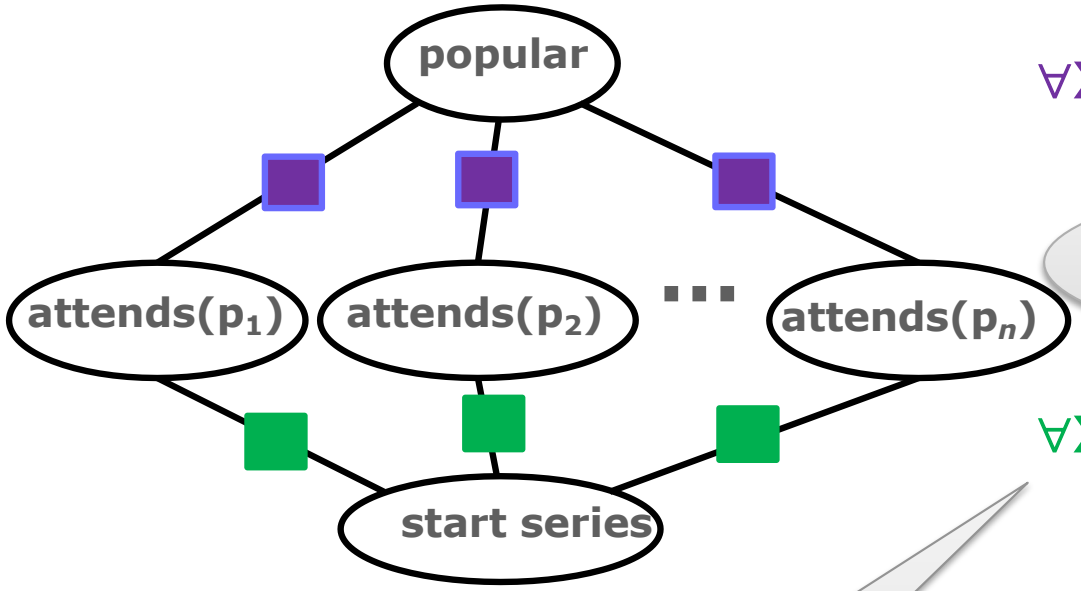
- Relational topic models
- Mixed-membership models
- Relational Gaussian processes
- Relational reinforcement learning
- (Partially observable) MDPs
- Systems of linear equations
- Kalman filters
- Declarative information networks
- …

General idea

TECHNISCHE UNIVERSITÄT DARMSTADT

- Given a relational model in your language of choice, a set of constants and a query, **compile** everything **into an intermediate respresentation**
  - **(logically parameterized) Factor graphs**
  - **BDDs, Probabilistic Circtuits, d-DNNFs, …**
  - **Weighted CNFs**
- **Run** (lifted) **inference**

Rules + Potential: Logically Parameterized Factors (parfactors) [Poole 2003; de Salvo Braz et al. 2005,…]



$\forall X.\ \phi_1(\text{popular}, \text{attends}(X))$

Logical Variables parameterize RV

$\forall X.\ \phi_2(\text{attends}(X), \text{series})$

Atoms represent a set of random variables

Parfactors parameterized factors

There can also be contraints to logical variables such as X=/=AAAI2017

[Domingos et al.; Van den Broeck et al. and many others]

Rules + Weights: Weighted CNF

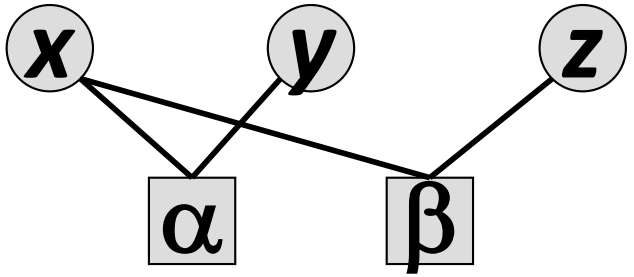
- Weighted MAX-SAT as mode finding for log-linear distributions
- Each configuration has a cost: the sum of the weights of the unsatisfied (ground) clauses.
- An infinite cost gives a 'hard' clause.
- Goal: find an assignment with minimal cost.

Weigthed CNF

$$\underbrace{(x \vee y)}_{\alpha} \wedge \underbrace{(\neg x \vee z)}_{\beta}$$

w1 w2

Factor Graph:



x	y	$f_\alpha(x,y)$
0	0	0
0	1	w1
1	0	w1
1	1	w1

x	z	$f_\beta(x,z)$
0	0	w2
0	1	w2
1	0	0
1	1	w2

[Darwiche et al.; Van den Broeck et al. and many others]

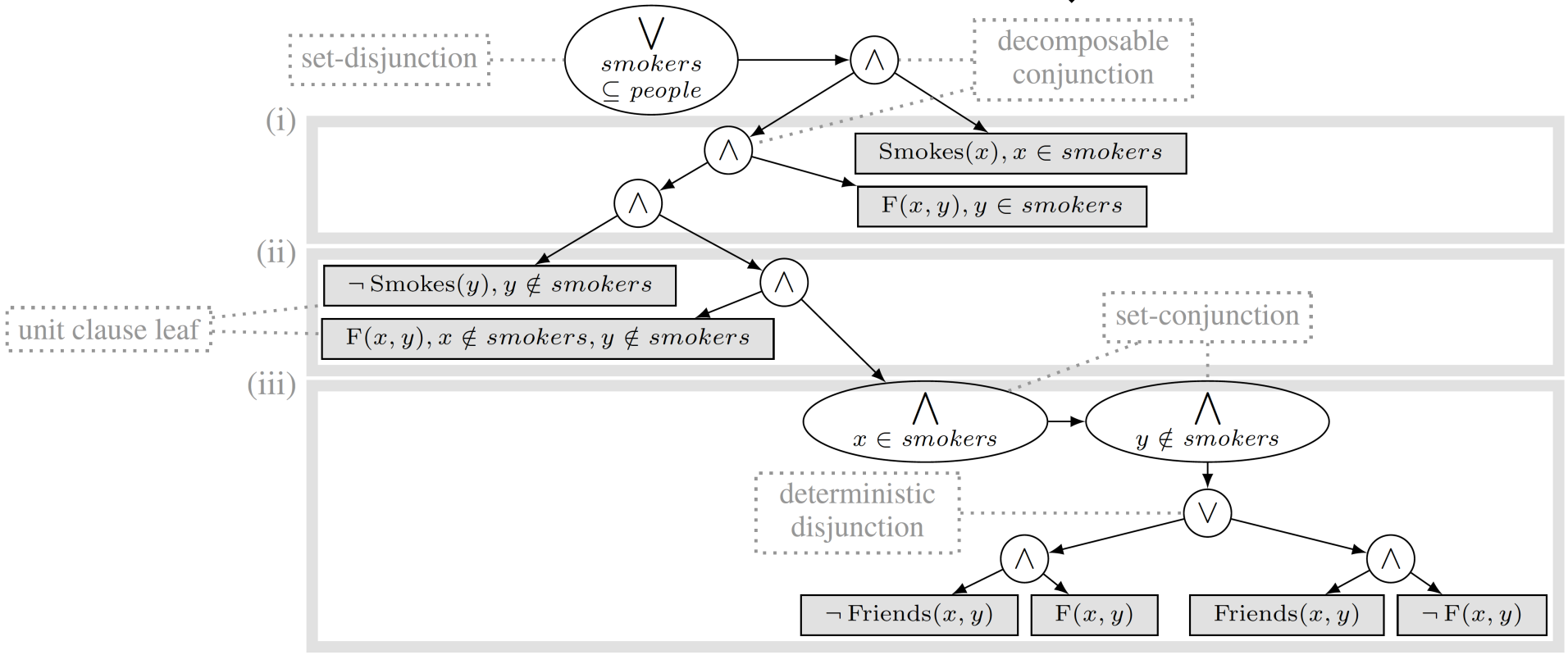
Rules + Weights: Knowledge Compilation

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

$$w \quad \text{F}(x, y)$$

$$\infty \quad \text{F}(x, y) \equiv [\text{Smokes}(x) \wedge \text{Friends}(x, y) \Rightarrow \text{Smokes}(y)] .$$

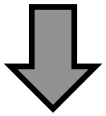
d-DNNF



[Darwiche et al.; Van den Broeck et al. and many others]

And even low-level C/C++ programs

$$\{\langle\{x,m\}, r(x,m) \wedge s(x,m), 1.2\rangle, \langle\{x,m\}, s(x,m) \wedge t(x), 0.2\rangle\}$$



```
1.  v2=0;
2.  for (int i = 0; i <= 2; i++){
3.      v5 = 0;
4.      for (int j = 0; j <= i; j++)
5.          v5 += choose(i, j) * exp(1.2 * j);
6.      v2 += choose(2, i) * pow(2, 2-i)
                    * v5 * (exp(0.2 * i) + 1);
7.  }
8.  v1 = pow(v2, 4);
```


What have we learnt?

TECHNISCHE UNIVERSITÄT DARMSTADT

- There are several ways to specify relational probabilistic models
- The goal is not to have a probabilistic characterization
- They highlight different aspects of relational modeling
- Semantically this soup boiled down to weighted CNFs, factor graphs, parfactors, diagrams and even program code

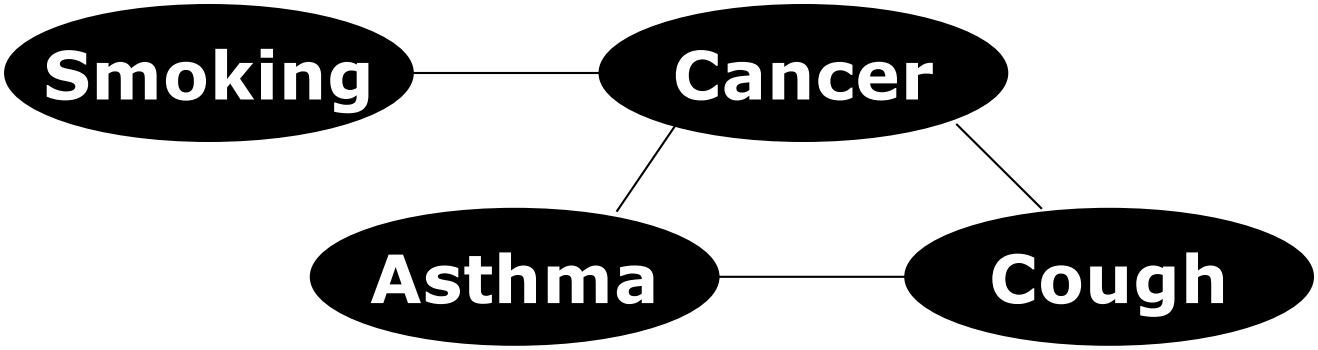
Now, let’s have a look at specific probabilistic relational model languages in more details.

MARKOV LOGIC NETWORKS

Markov Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Undirected graphical models



- Potential functions defined over cliques

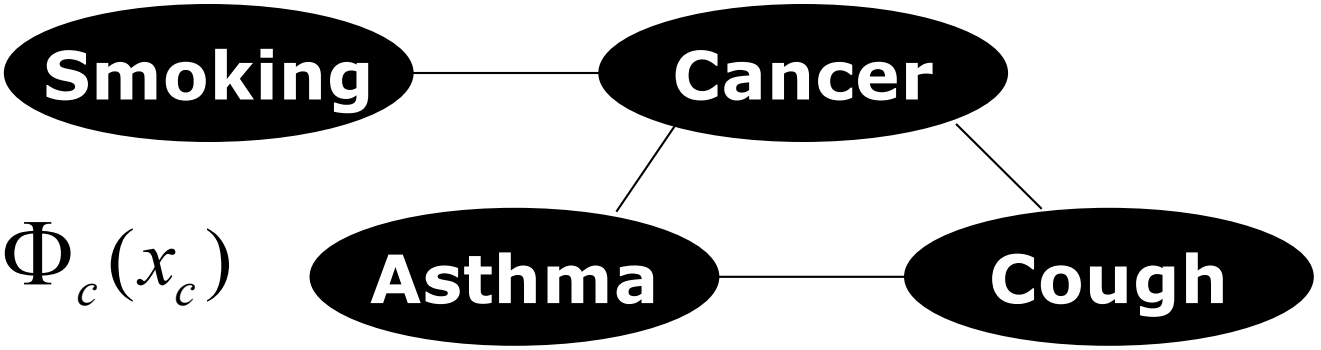
$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	Cancer	Φ(S,C)
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

Markov Networks

- Undirected graphical models



$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

- Log-linear model:

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

Weight of Feature *i*

Feature *i*

Allows us to put structure into undirected models

$$f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg \text{Smoking} \vee \text{Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 0.51$$

Markov Logic: Intuition

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Markov Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- A Markov Logic Network (MLN) is a set of pairs (F, w) where
  - F is a formula in first-order logic
  - w is a real number
- An MLN defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w

Markov Logic Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probability of a world x:

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

Weight of formula *i*

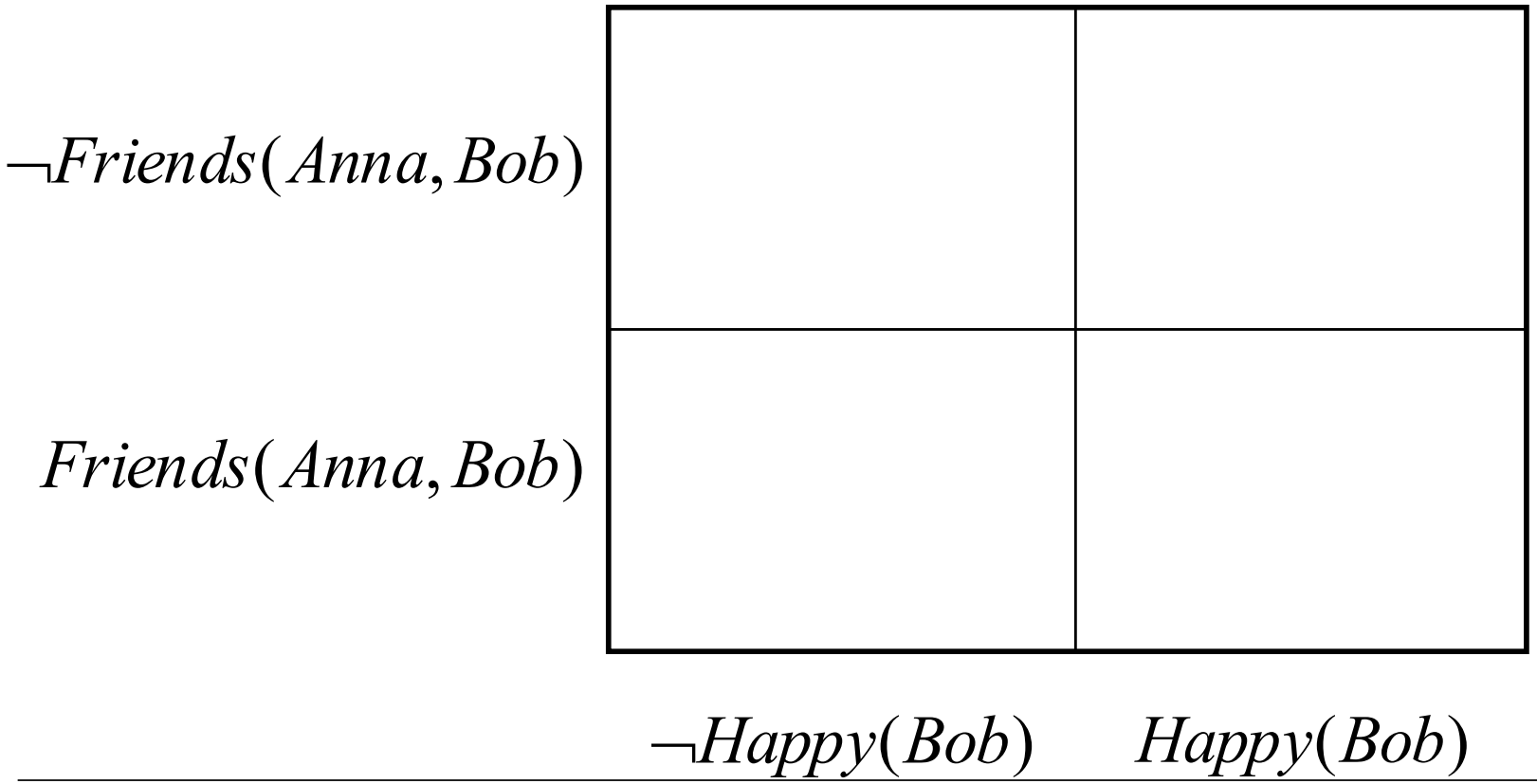
No. of true groundings of formula *i* in *x*

- So the main difference to log-linear models is that we now count how often the feature (the clauses) are true
- **Typed** variables and constants greatly reduce size of ground Markov net
- Functions, existential quantifiers, etc.
- Infinite and continuous domains

A possible worlds view

TECHNISCHE UNIVERSITÄT DARMSTADT

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



A possible worlds view

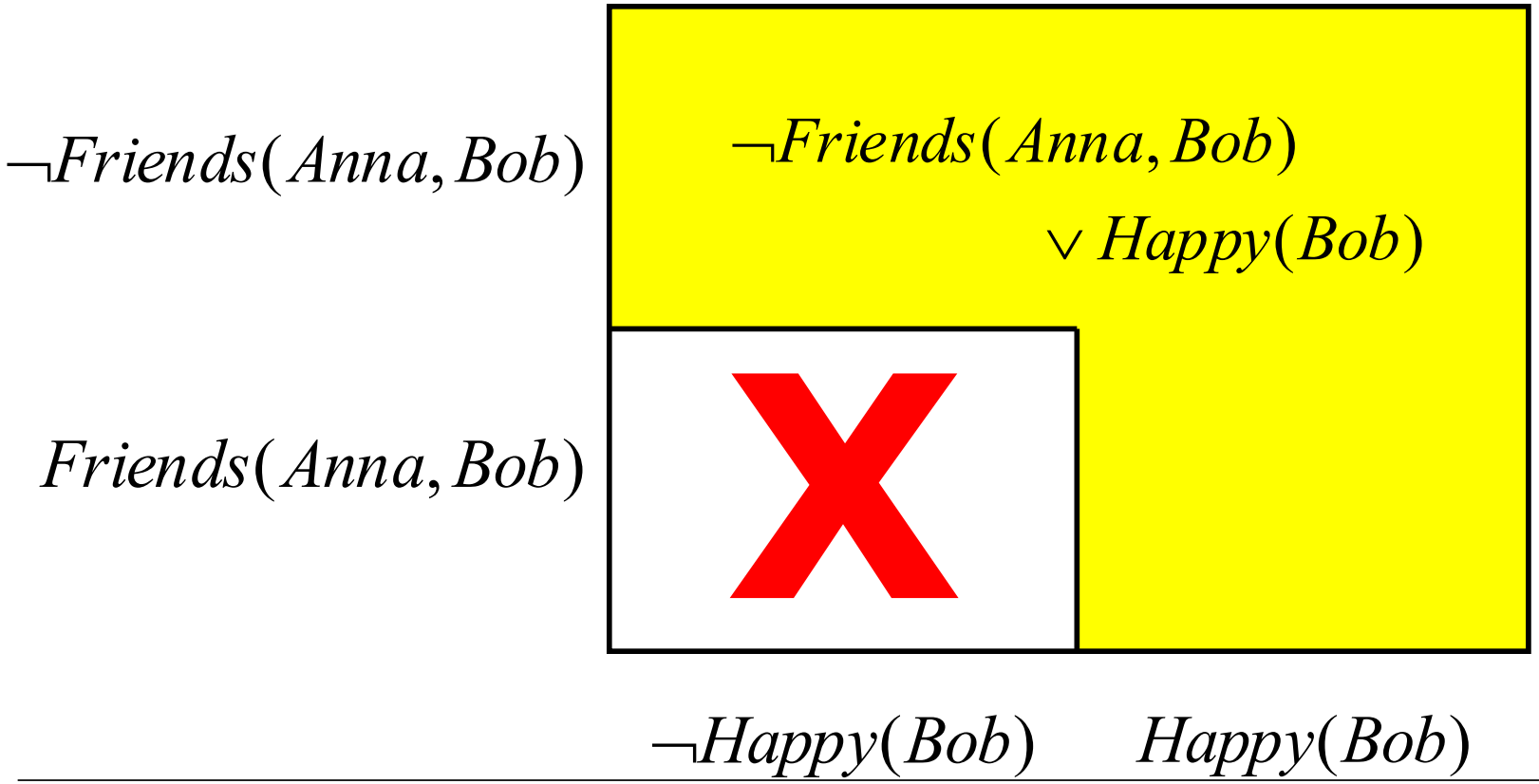
TECHNISCHE UNIVERSITÄT DARMSTADT

Logical formulas such as

IF Friends(Anna,Bob) THEN Happy(Bob)

exclude possible worlds such as

Friends(Anna,Bob), NOT Happy(Bob)

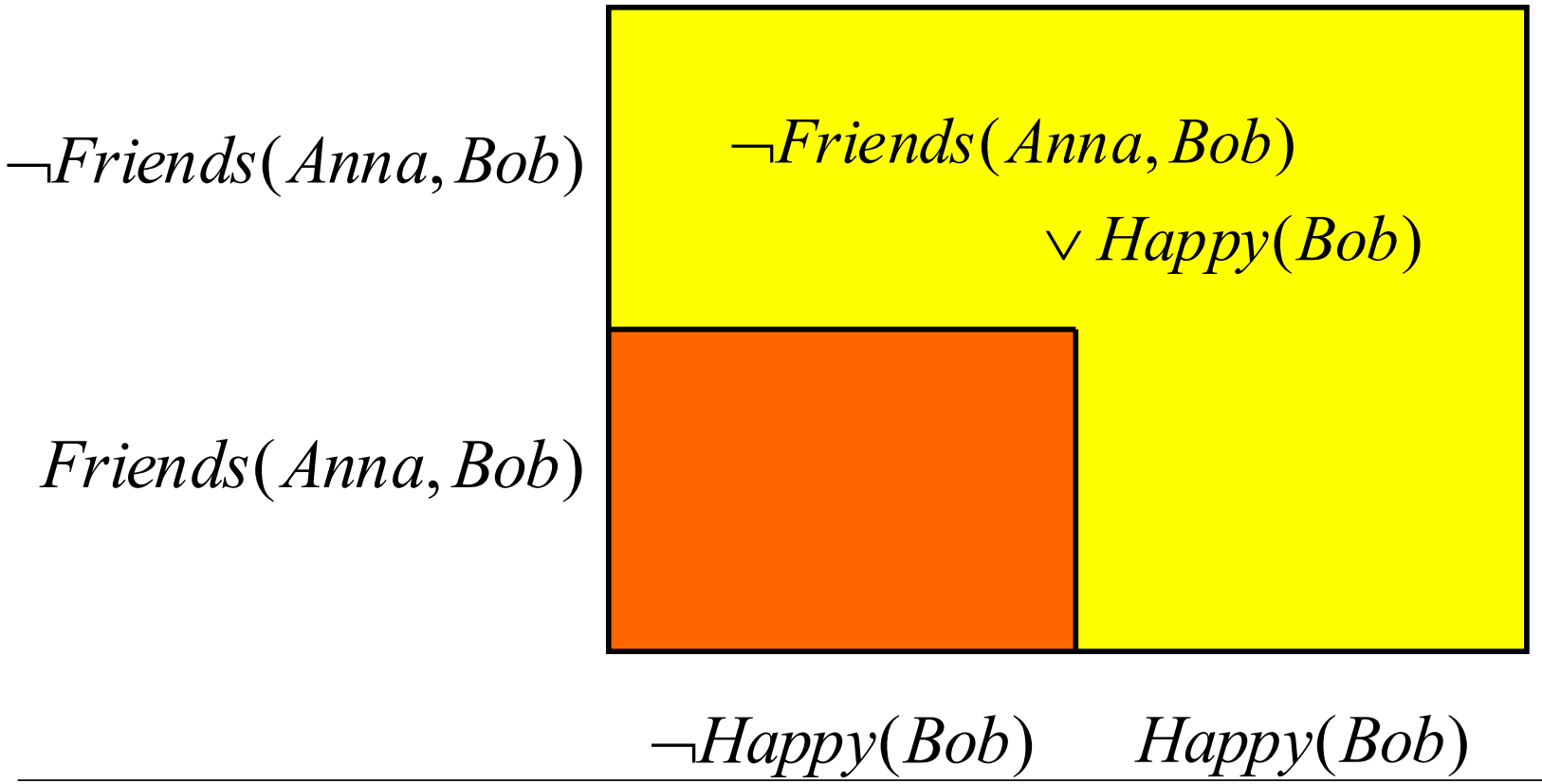


A possible worlds view

Instead of excluding worlds, we want them to become less likely, e.g. $P(\neg Friends(Anna,Bob) \vee Happy(Bob)) = 0.8$

This leaves some probability mass for

Friends(Anna,Bob), NOT Happy(Bob)

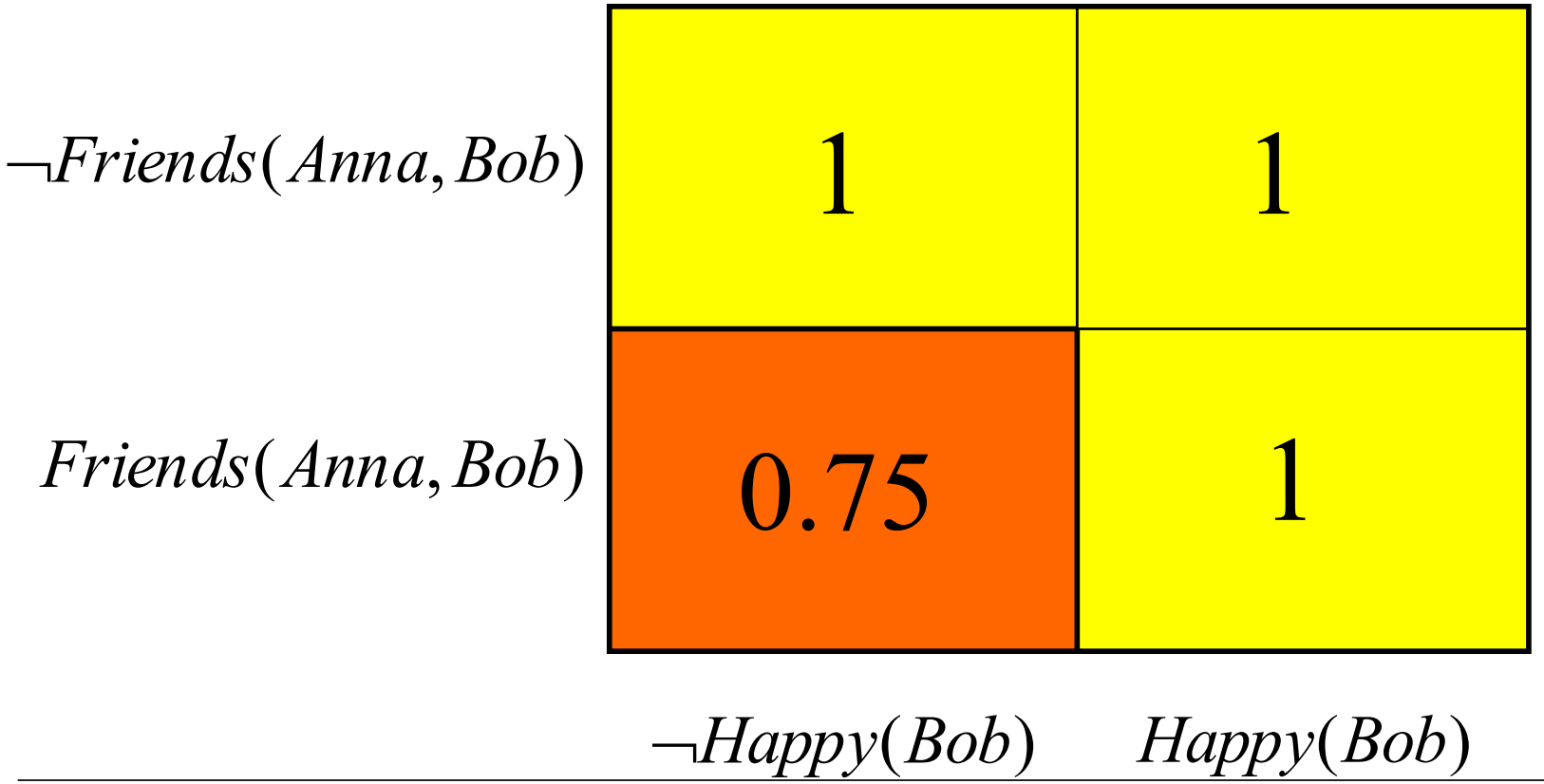


A possible worlds view

TECHNISCHE UNIVERSITÄT DARMSTADT

This can be achieved with the following potentials

$$\Phi(\neg Friends(Anna, Bob) \vee Happy(Bob)) = 1$$
$$\Phi(Friends(Anna, Bob) \wedge \neg Happy(Bob)) = 0.75$$



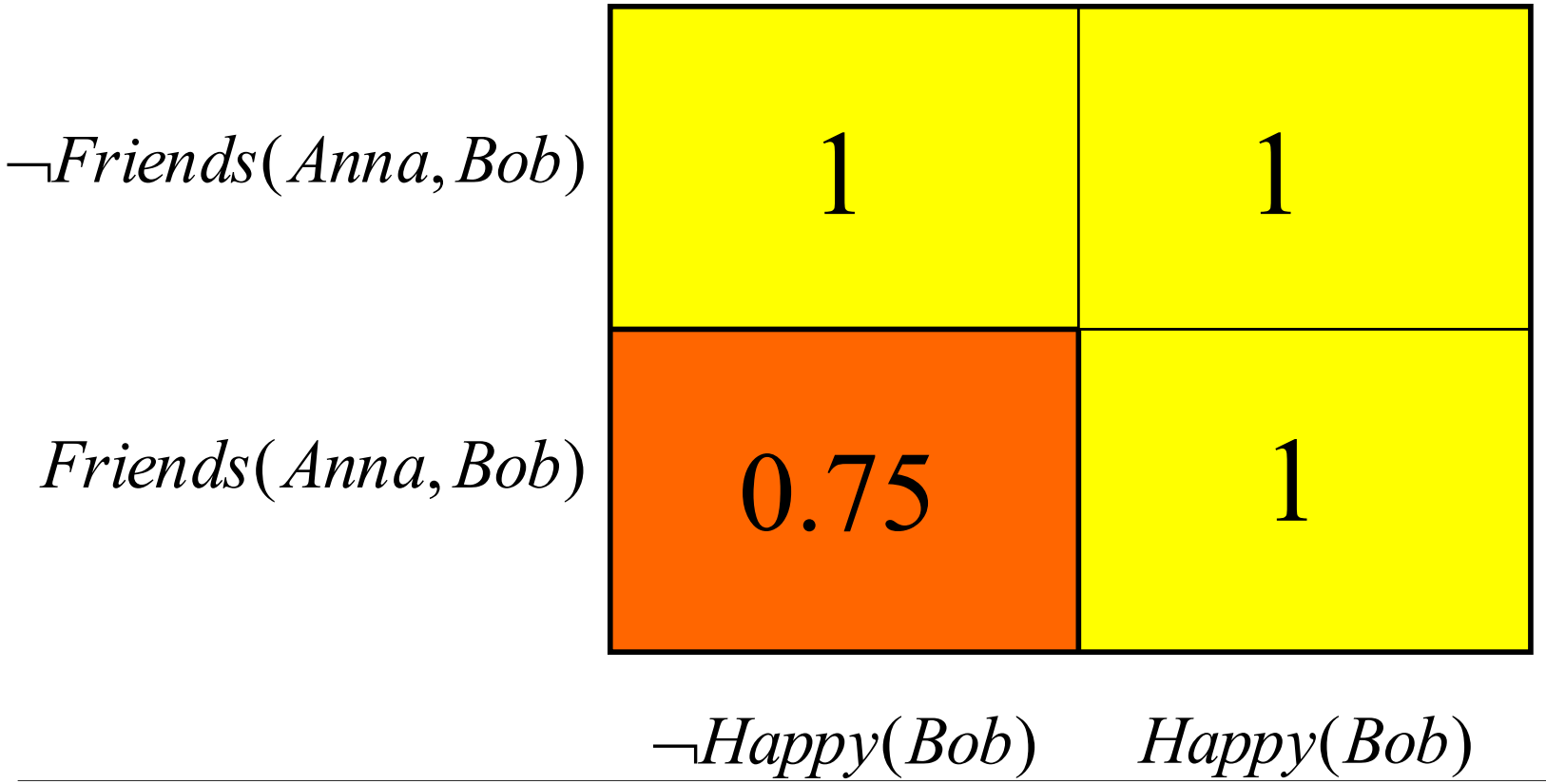
Since then, (1+1+1) /(1+1+1+0.75) = 3 / 3.75 = 0.8

A possible worlds view

TECHNISCHE UNIVERSITÄT DARMSTADT

Or as log-linear model this is:

$$w(\Phi(\neg Friends(Anna, Bob) \vee Happy(Bob)))$$
$$= \log(1/0.75) = 0.29$$



This can also be viewed as building a graphical model

A graphical example : Friends & Smokers

Smoking causes cancer.

Friends have similar smoking habits.

A graphical example : Friends & Smokers

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

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

A graphical example : Friends & Smokers

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

A graphical example : Friends & Smokers

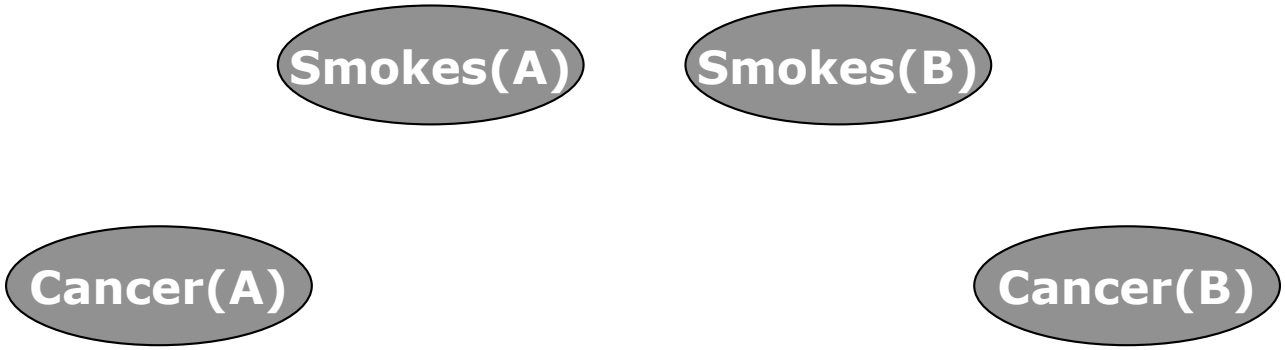
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

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

A graphical example : Friends & Smokers

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

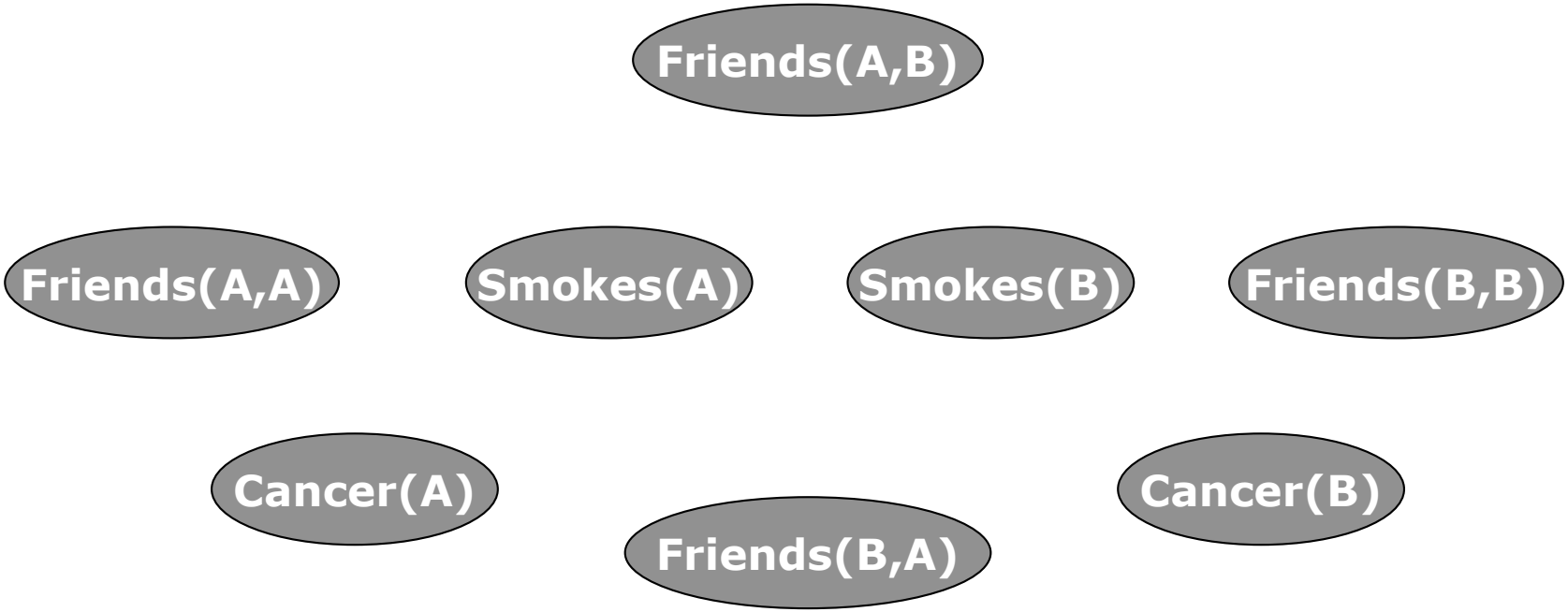
Two constants: Anna (A) and Bob (B)



A graphical example : Friends & Smokers

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

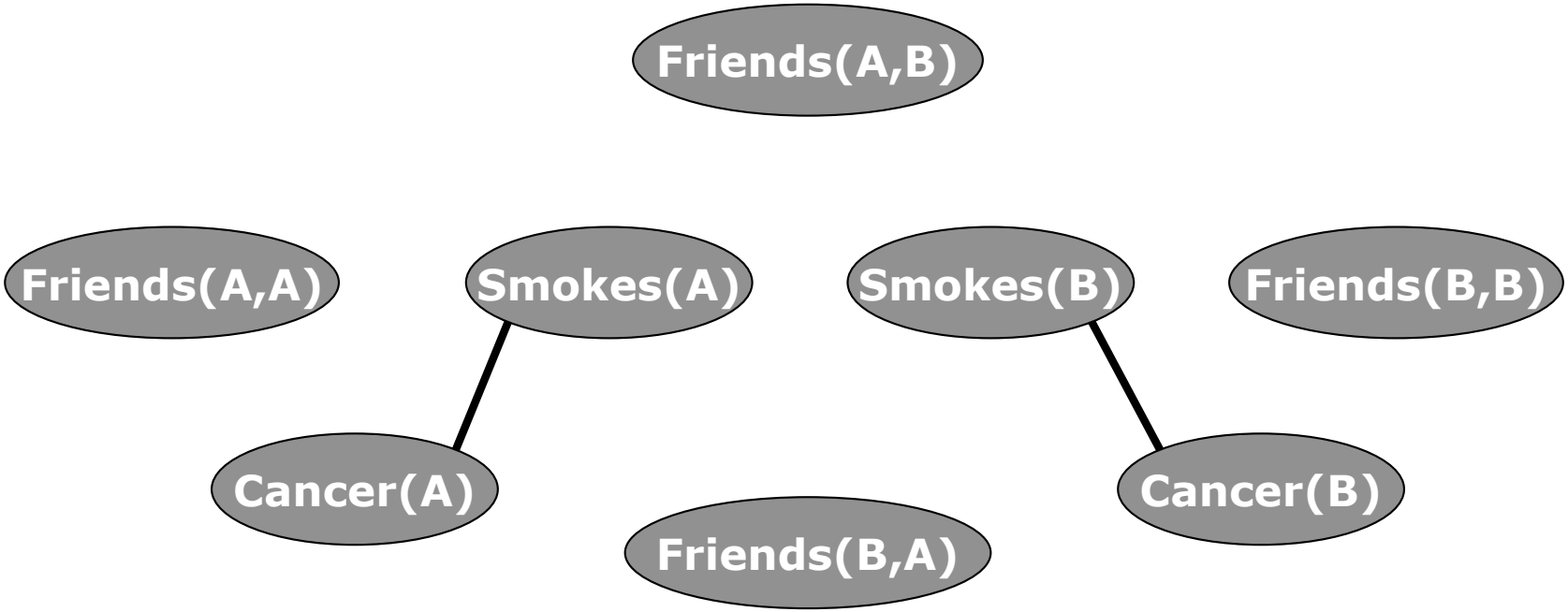
Two constants: Anna (A) and Bob (B)



A graphical example : Friends & Smokers

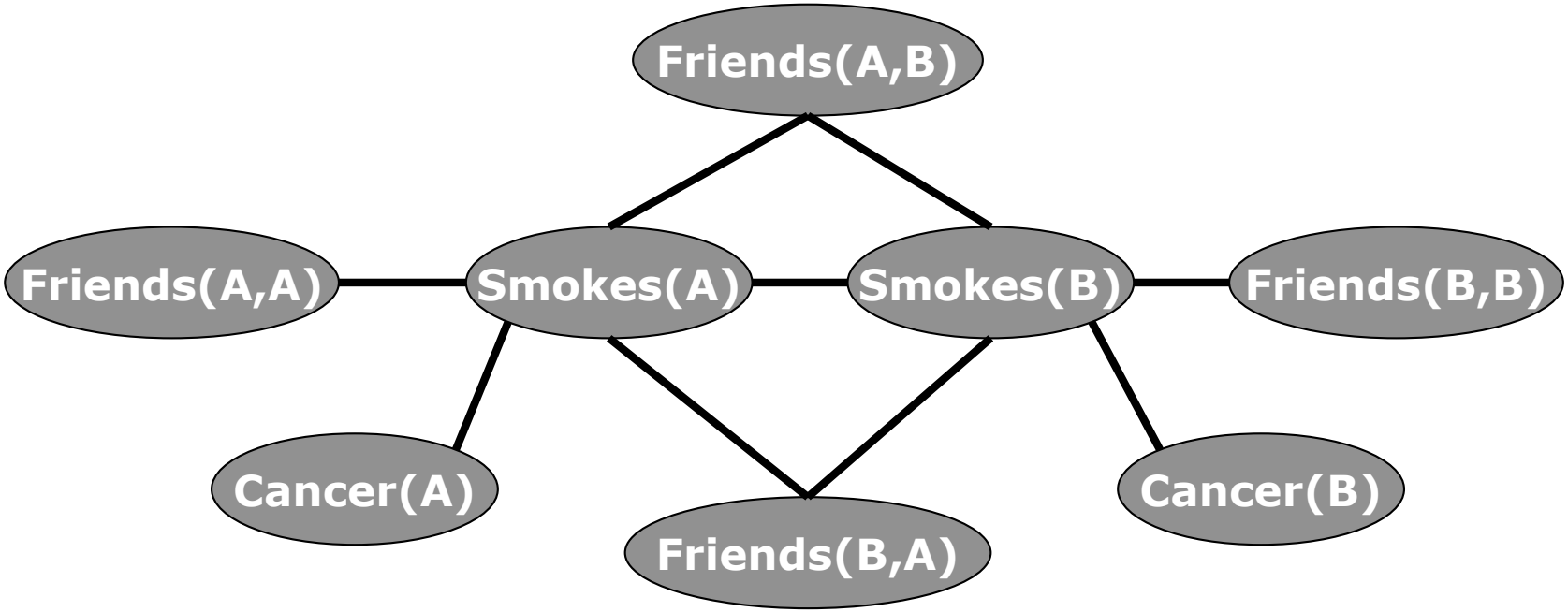
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

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



A graphical example : Friends & Smokers

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

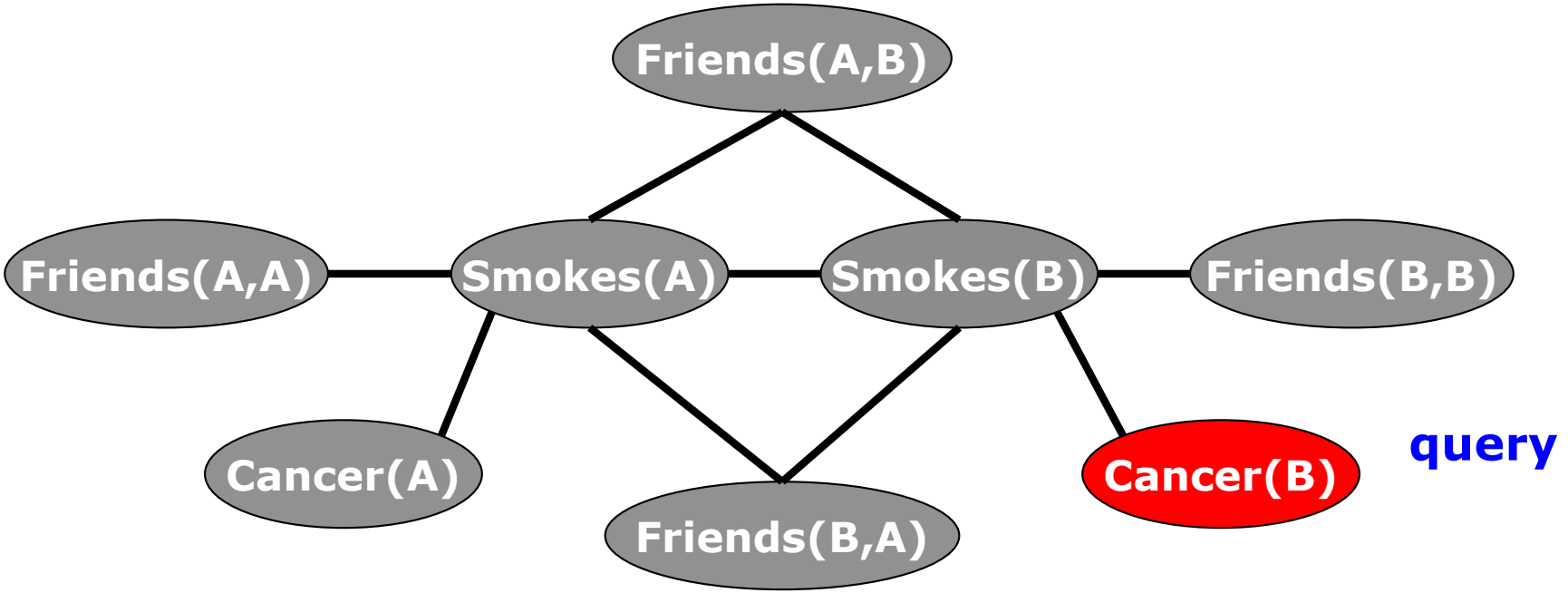


Alternatively, this can be viewed as a set of weighted ground clauses

A graphical example : Friends & Smokers

Can be combined with ideas from graphical models and logic programming to answer queries "efficiently"

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

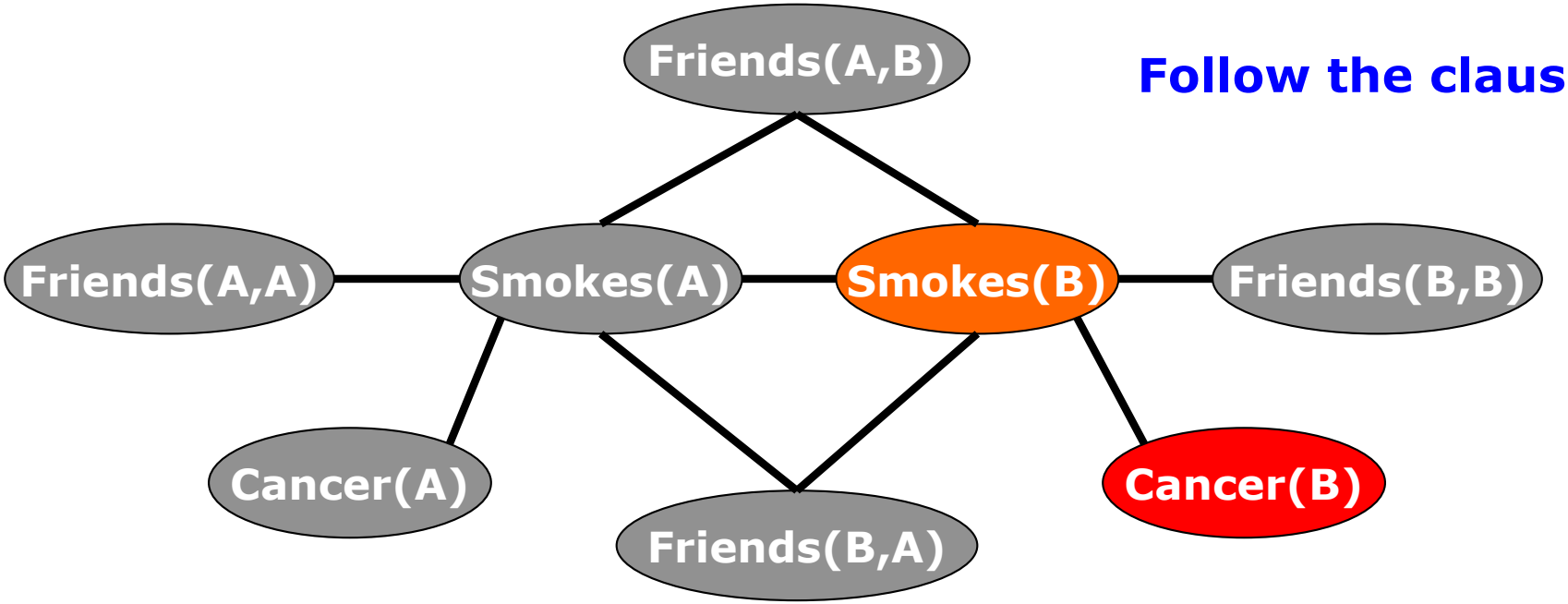


P(Cancer(B) | Smokes(A), Friends(A,B), Friends(B,A))

A graphical example : Friends & Smokers

Can be combined with ideas from graphical models and logic programming to answer queries "efficiently"

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

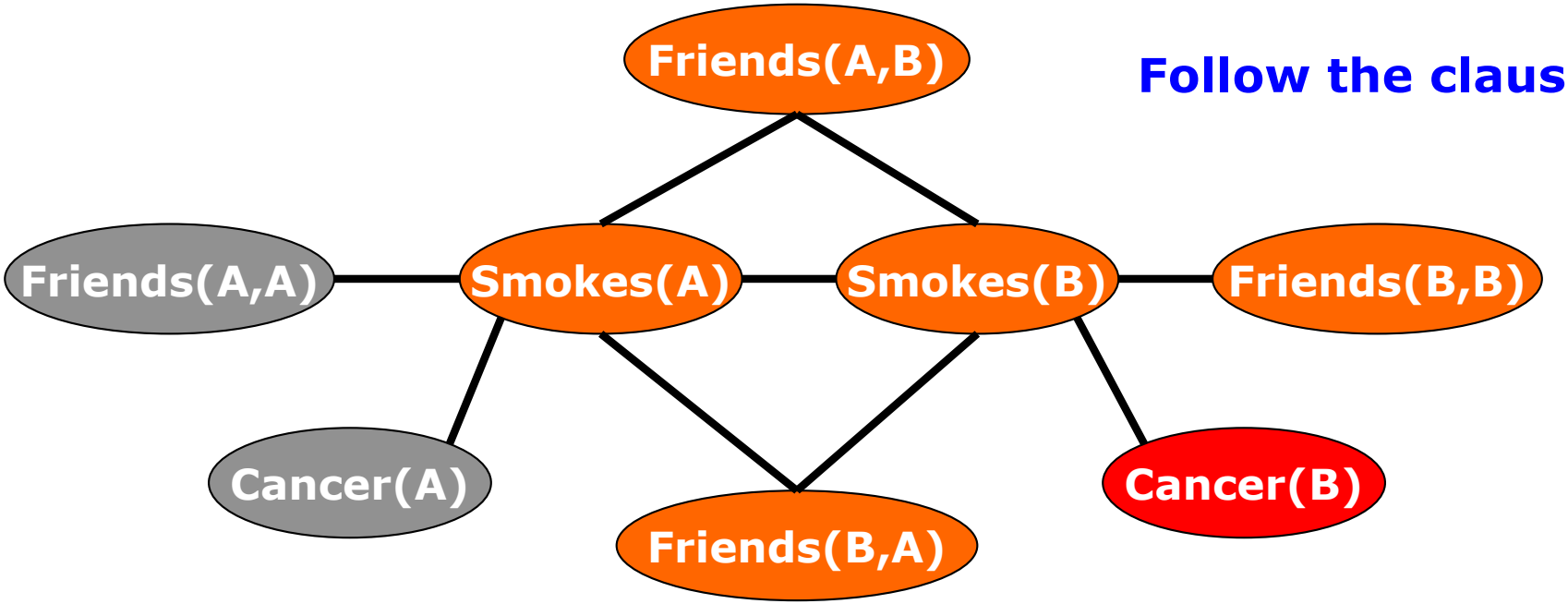


P(Cancer(B) | Smokes(A), Friends(A,B), Friends(B,A))

A graphical example : Friends & Smokers

Can be combined with ideas from graphical models and logic programming to answer queries "efficiently"

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

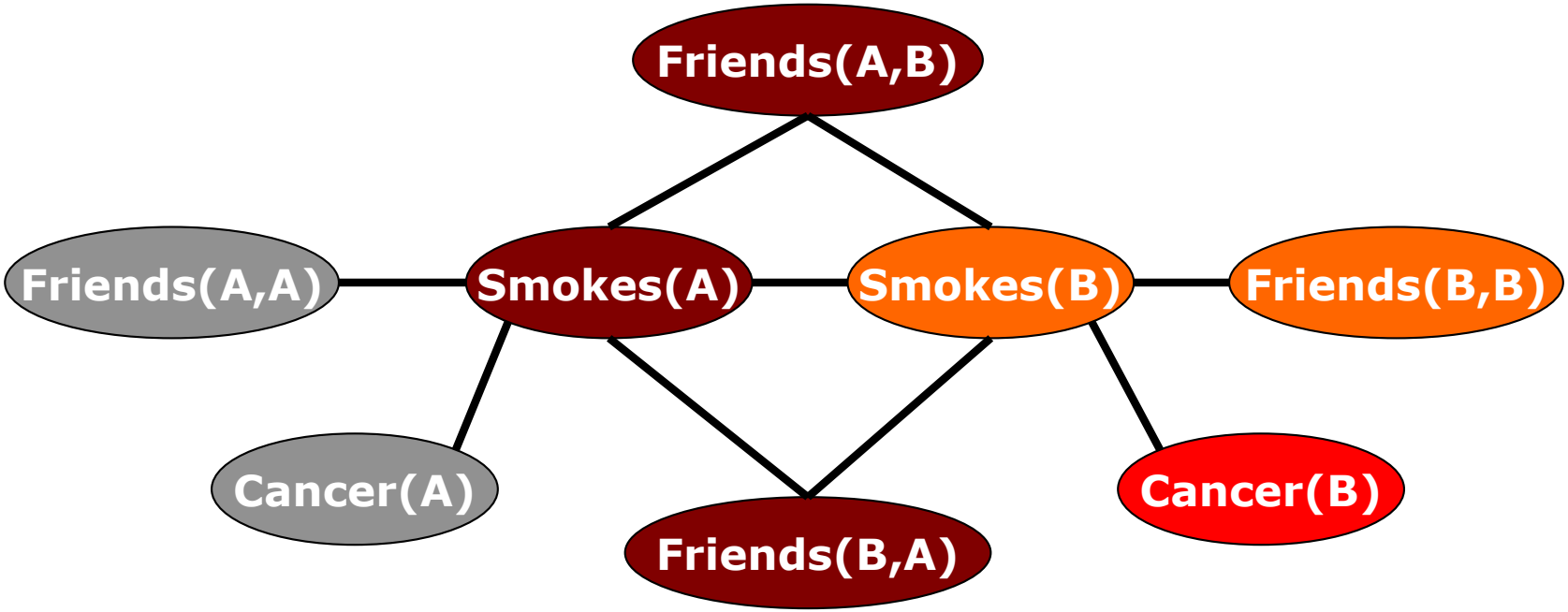


P(Cancer(B) | Smokes(A), Friends(A,B), Friends(B,A))

A graphical example : Friends & Smokers

Can be combined with ideas from graphical models and logic programming to answer queries "efficiently"

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

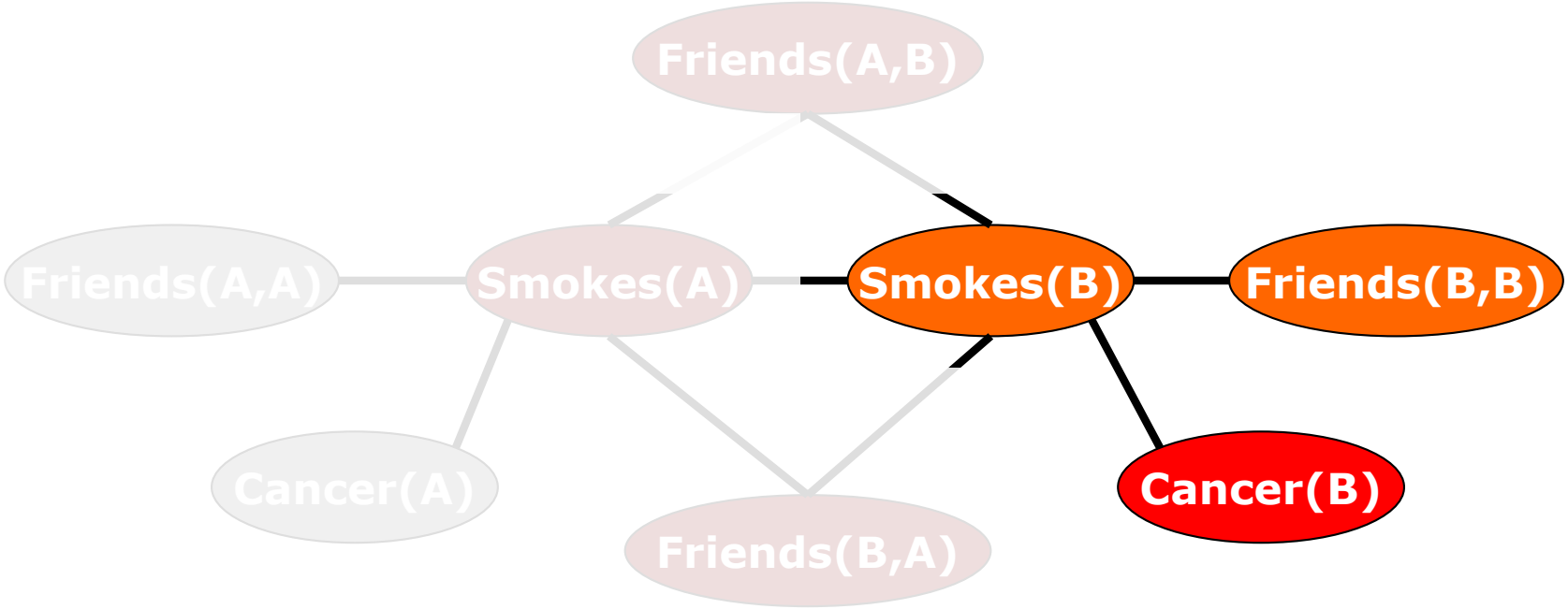


P(Cancer(B) | Smokes(A), Friends(A,B), Friends(B,A))

A graphical example : Friends & Smokers

Can be combined with ideas from graphical models and logic programming to answer queries "efficiently"

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



P(Cancer(B) | Smokes(A), Friends(A,B), Friends(B,A))

We may not have to consider the complete induced Markov networks

Relation of MLNs to Probabilistic Models

- Special cases:
  - Markov networks
  - Markov random fields
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields
- Obtained by making all predicates zero-arity
- Markov logic allows objects to be interdependent (non-i.i.d.)

Relation of MLNs to First-Order Logic

- Infinite weights ⇒ First-order logic (without function symbols)
- Satisfiable KB, positive weights ⇒ Satisfying assignments = Modes of distribution
- In contrast to logic, Markov logic allows contradictions between formulas; the Moon might not be made of cheese!

Applications

TECHNISCHE UNIVERSITÄT DARMSTADT

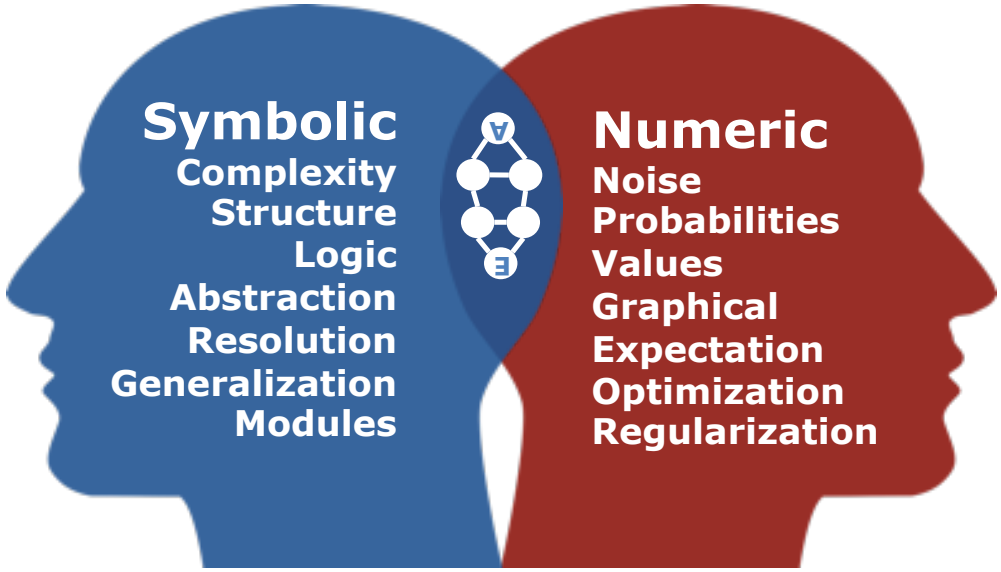
- Natural language processing, Robotics, Collective Classification, Social Networks, Activity Recognition, …

What have we learnt?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Markov logic networks combine first-order logic and Markov networks
  - Syntax: First-order logic + Weights
  - Semantics: Templates for Markov networks
- There are implementations available
- Important real world SRL problems easily formulated as MLNs

- Not touched: continuous RVs, tractable versions, compilation, efficient learning, …



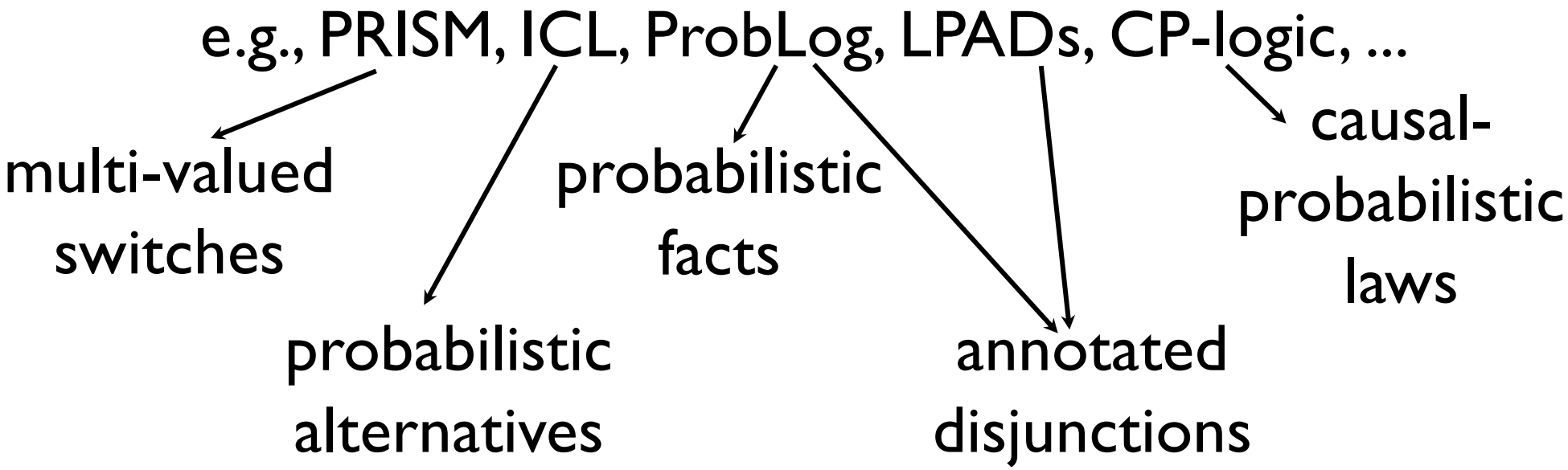
Statistical Relational AI

Relational Probabilistic Models - ProbLog

Kristian Kersting

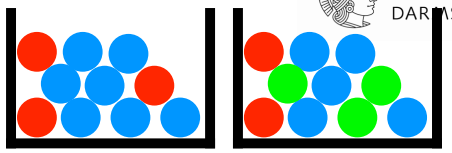
Thanks to Vincent Conitzer, Rina Dechter, Luc De Raedt, Pedro Domingos, Peter Flach, Dieter Fensel, Florian Ficher, Vibhav Gogate, Carlos Guestrin, Daphen Koller, Nir Friedman, Ray Mooney, Sriraam Natarajan, David Poole, Fabrizio Riguzzi, Dan Suciu, Guy van den Broeck, and many others for making their slides publically available

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds



ProbLog by example:

A bit of gambling



h

- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.
```
probabilistic fact: heads is true with prob. 0.4 (and false with 0.6)

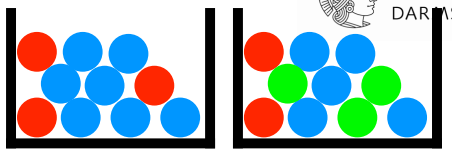
annotated disjunction: first ball is red with probability 0.3 and blue with 0.7

```
0.3 :: col(1,red);   0.7 :: col(1,blue).
0.2 :: col(2,red);   0.3 :: col(2,green);
                     0.5 :: col(2,blue).
```
annotated disjunction:
second ball is red with probability 0.2, green with 0.3, and blue with 0.5

```
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```
logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red);   0.7 :: col(1,blue).
0.2 :: col(2,red);   0.3 :: col(2,green);
                     0.5 :: col(2,blue).
```

probabilistic choices

```
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

consequences

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of `win`?

query **conditional probability**

- Probability of `win` given `col(2,green)`?

evidence

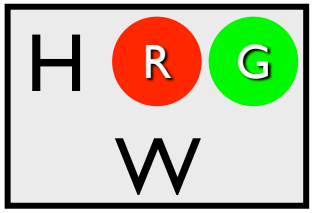
- Most probable world where `win` is true?

MPE inference

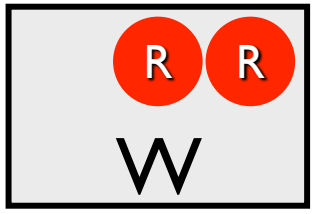
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

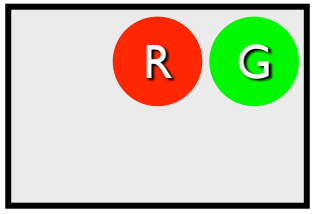
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$

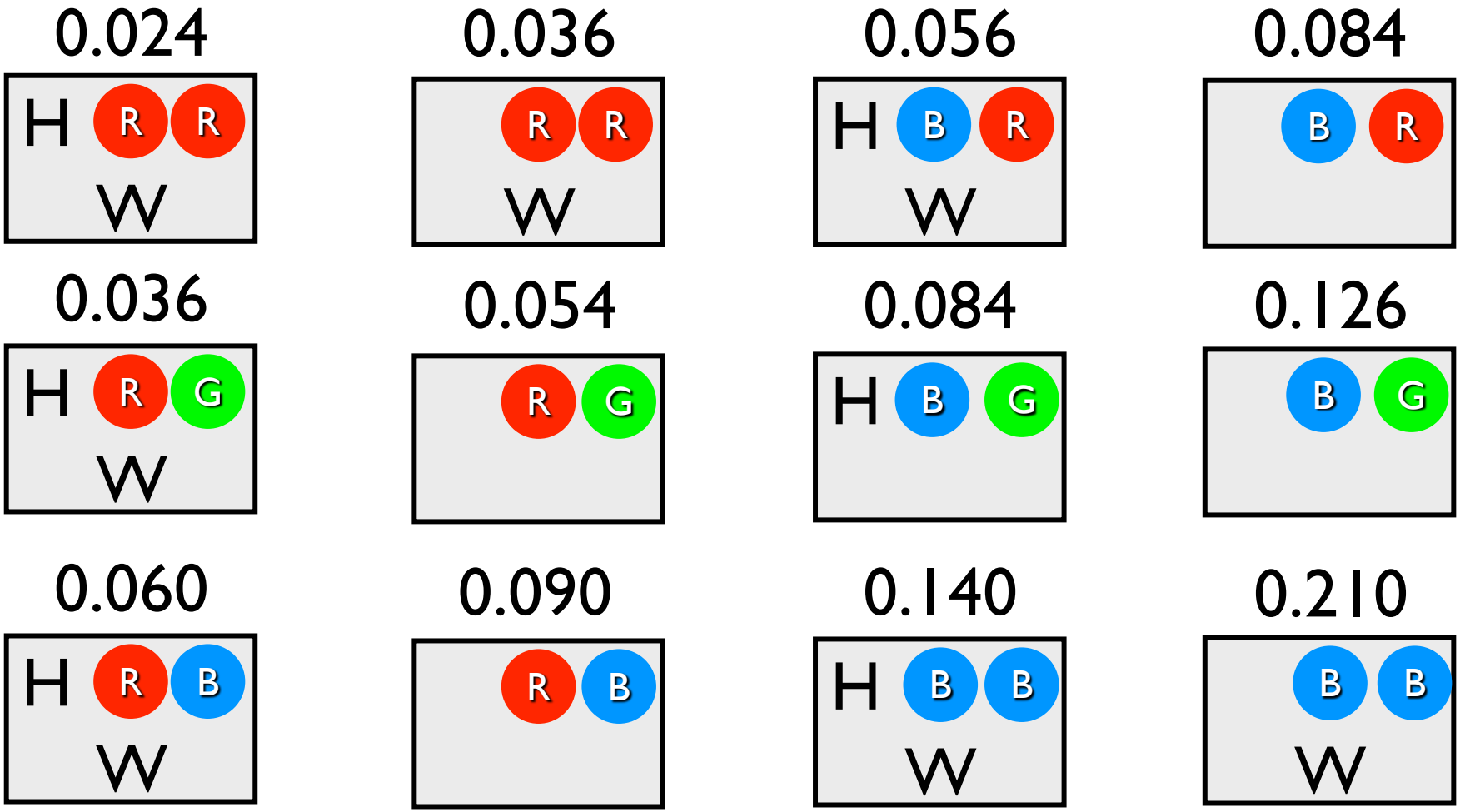


$(1-0.4) \times 0.3 \times 0.3$



All Possible Worlds

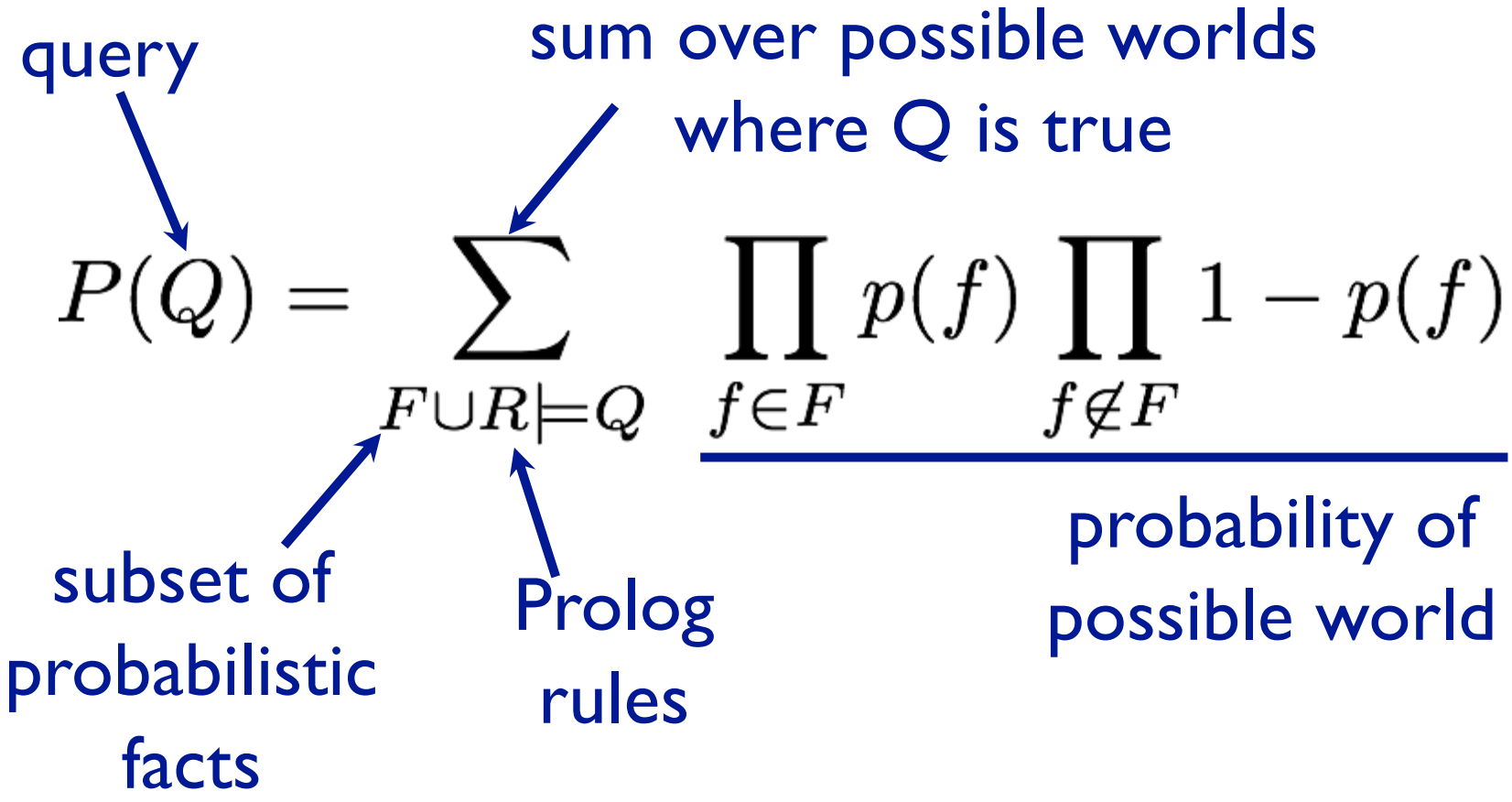
Using the possible worlds we can compute whatever we want (prob. query, MPE, …)



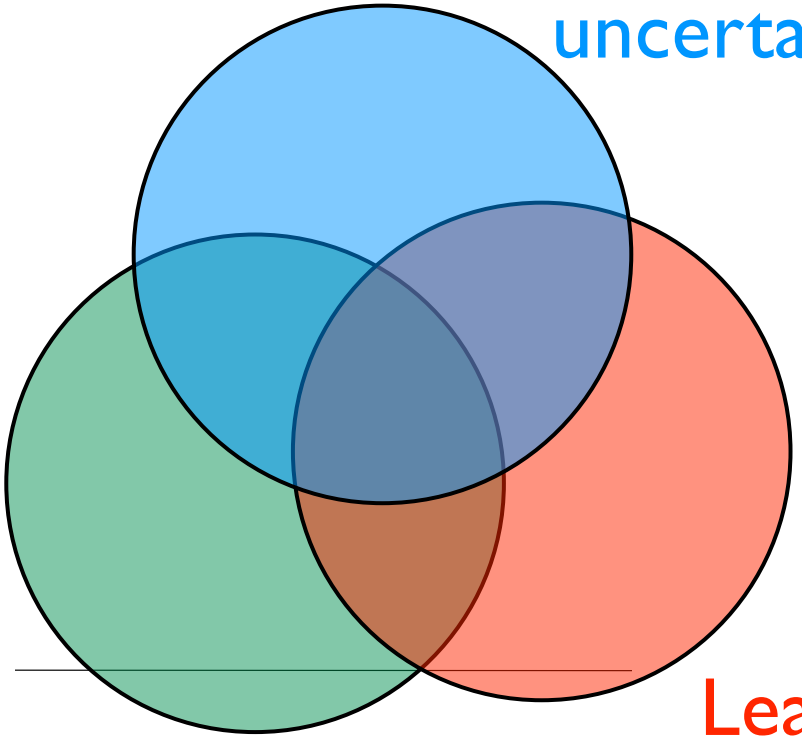
Generally, this follows the Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]



Probabilistic Databases [Suciu et al 2011]



Probabilistic Databases

[Suciu et al 2011]

several possible worlds

bornIn

person	city	P
ann	london	0.87
bob	york	0.95
eve	new york	0.9
tom	paris	0.56

cityIn

city	country	P
london	uk	0.99
york	uk	0.75
paris	usa	0.4

tuples as random variables

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

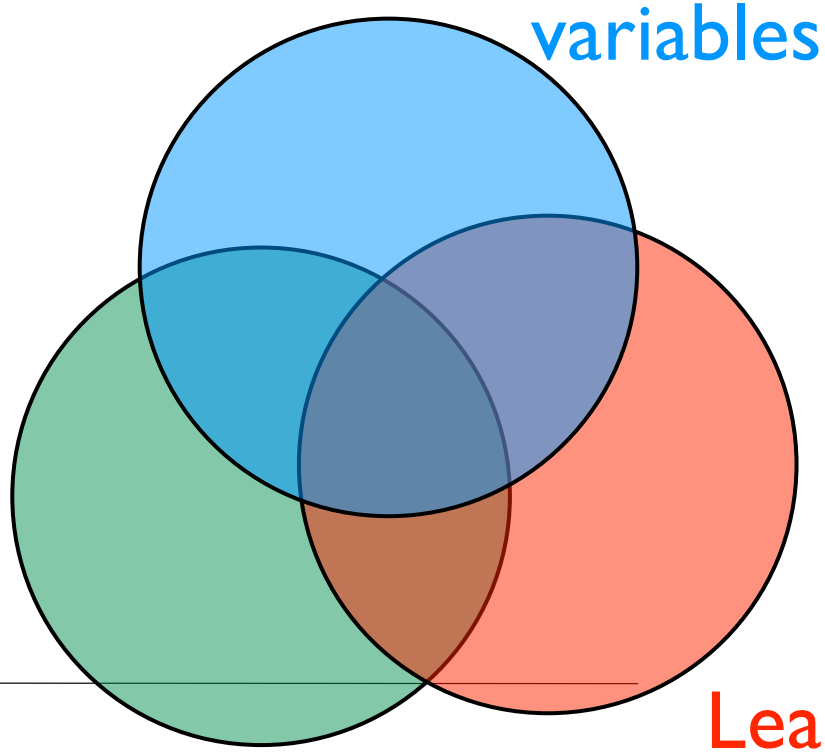
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Learning

Probabilistic Databases

[Suciu et al. 2011]

TECHNISCHE UNIVERSITÄT DARMSTADT

several possible worlds

bornIn

person	city	P
ann	london	0.87
		0.95
		0.9
		0.56

cityIn

city	country	P
london	uk	0.99
york	uk	0.75
paris	usa	0.4

tuples as random variables

probabilistic tables + database queries
→ distribution over possible worlds

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

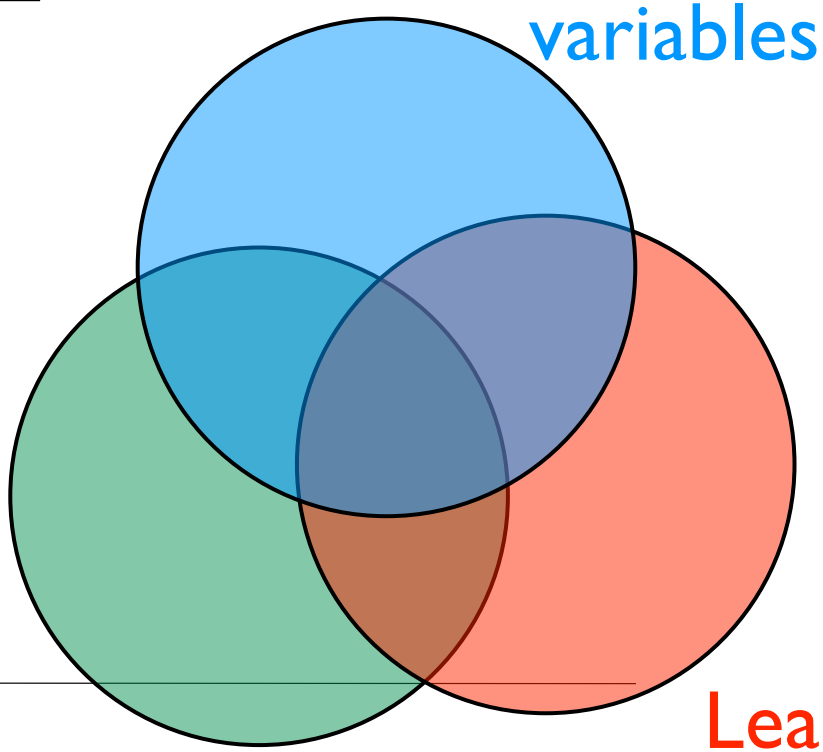
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

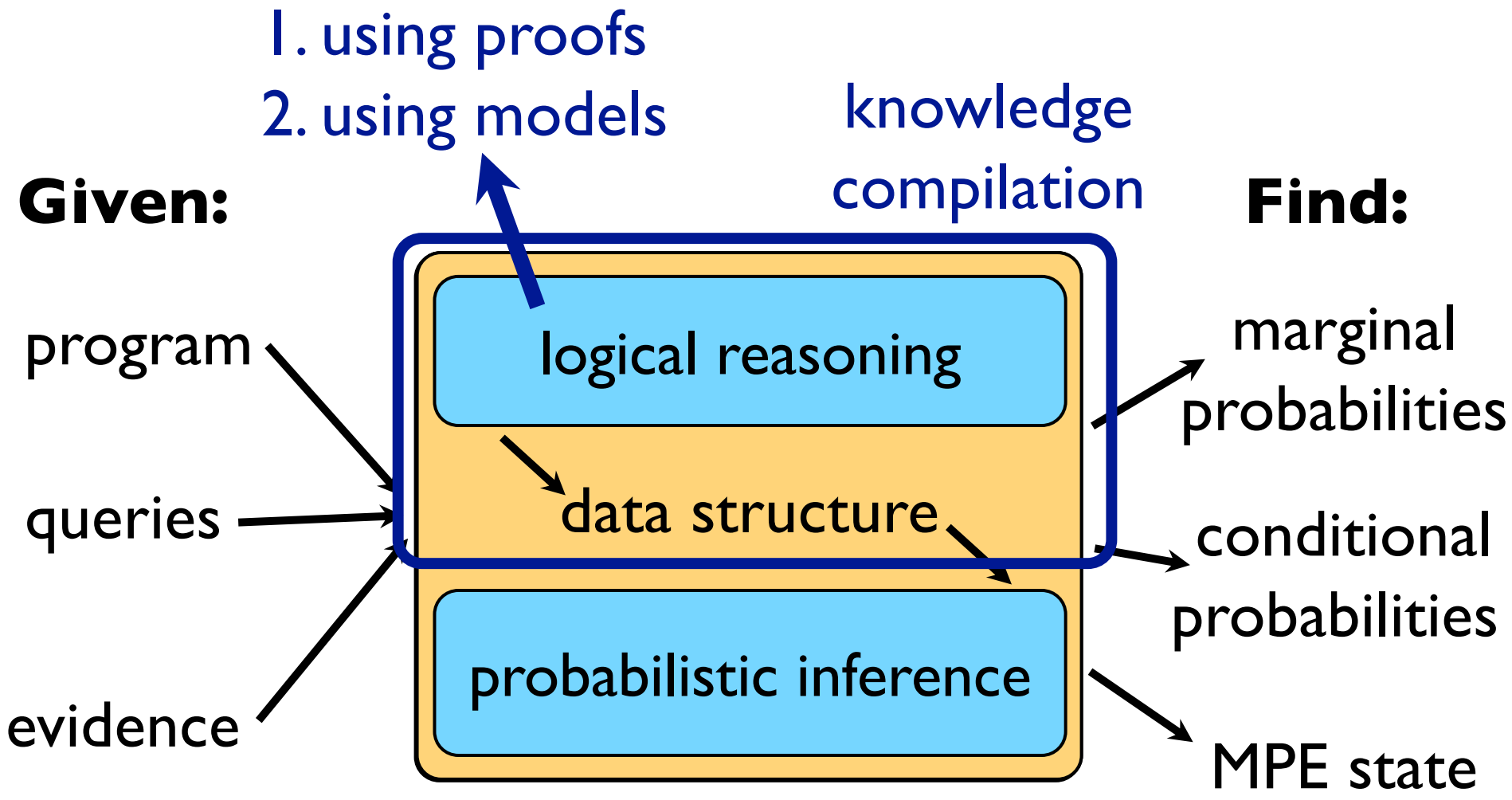
city	country
london	uk
york	uk
paris	usa

relational database



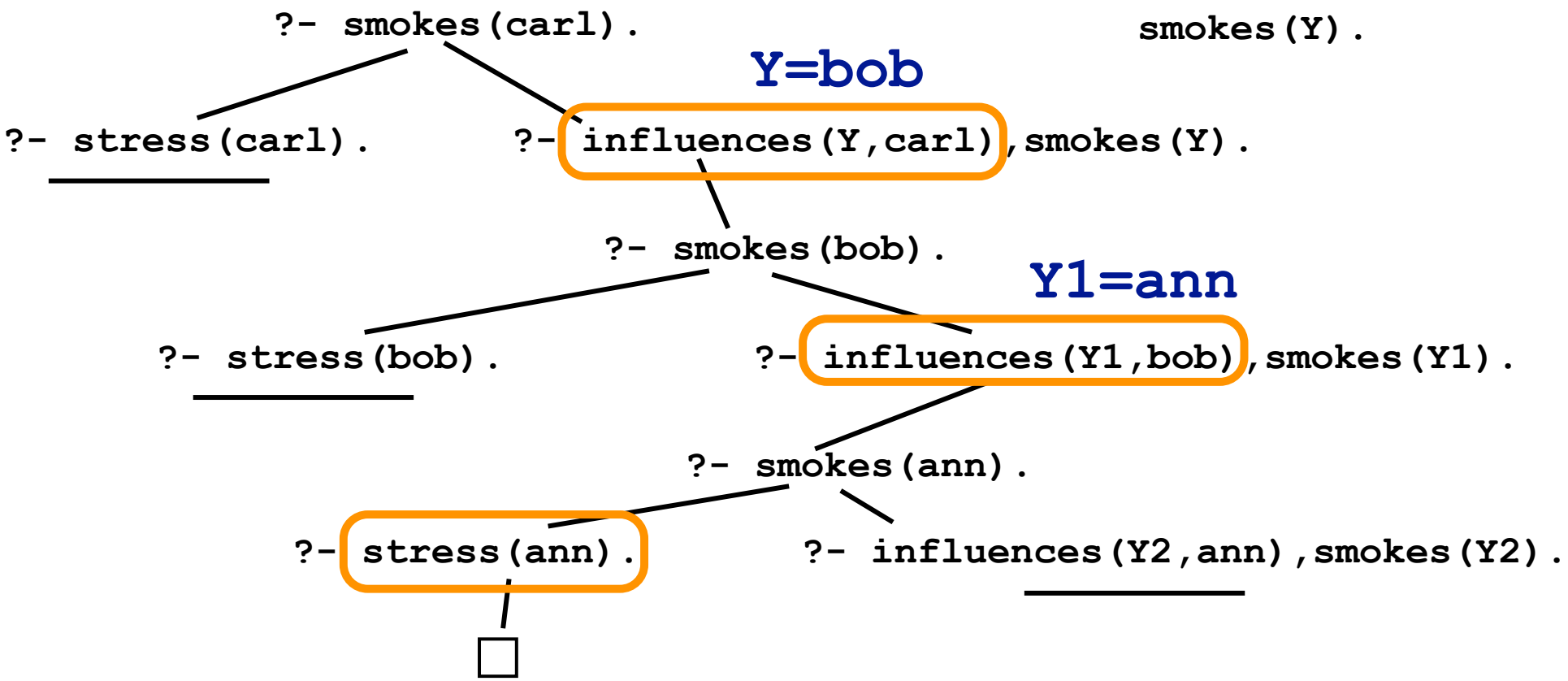
Learning

Answering Questions



Logical Reasoning: Proofs in Prolog
(a reminder)

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

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

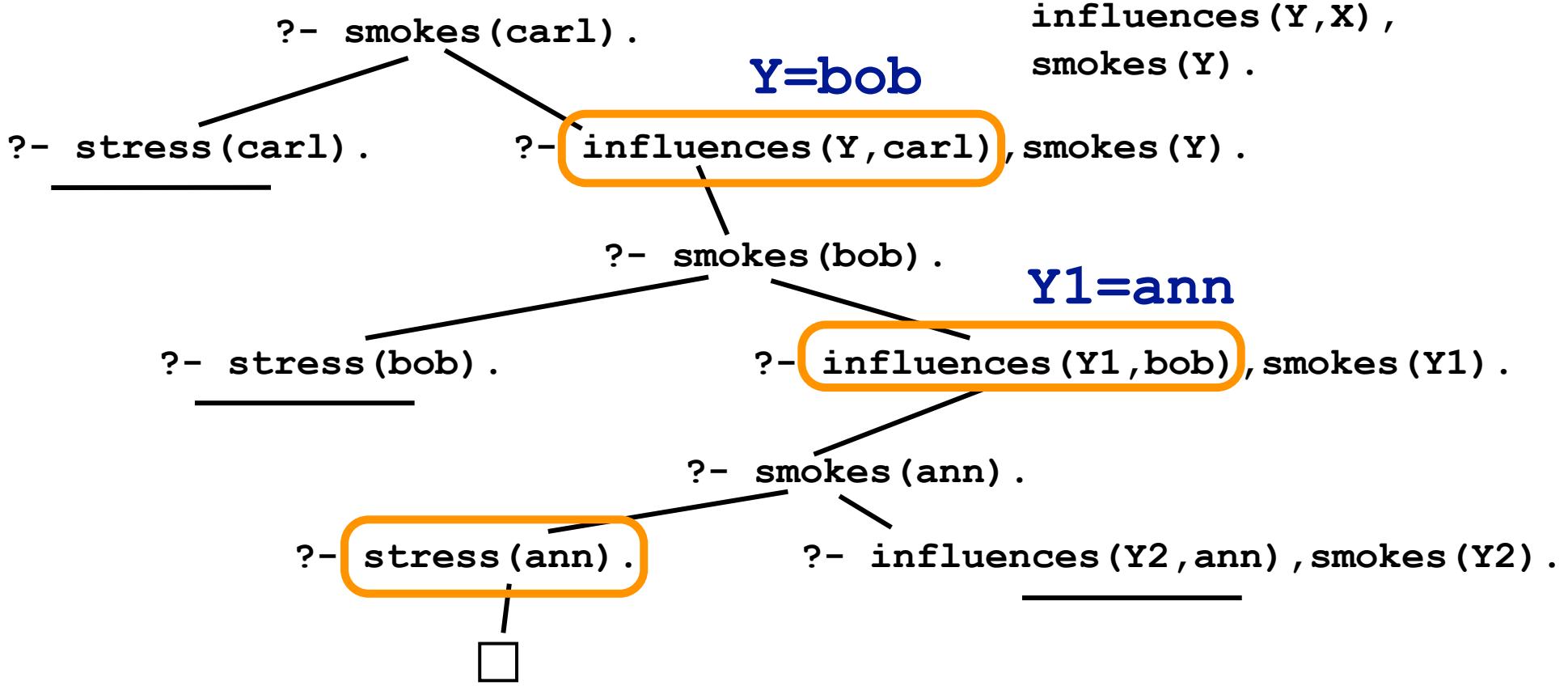


proof = facts used in successful derivation:

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

(Probabilistic) Proofs in ProbLog

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

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

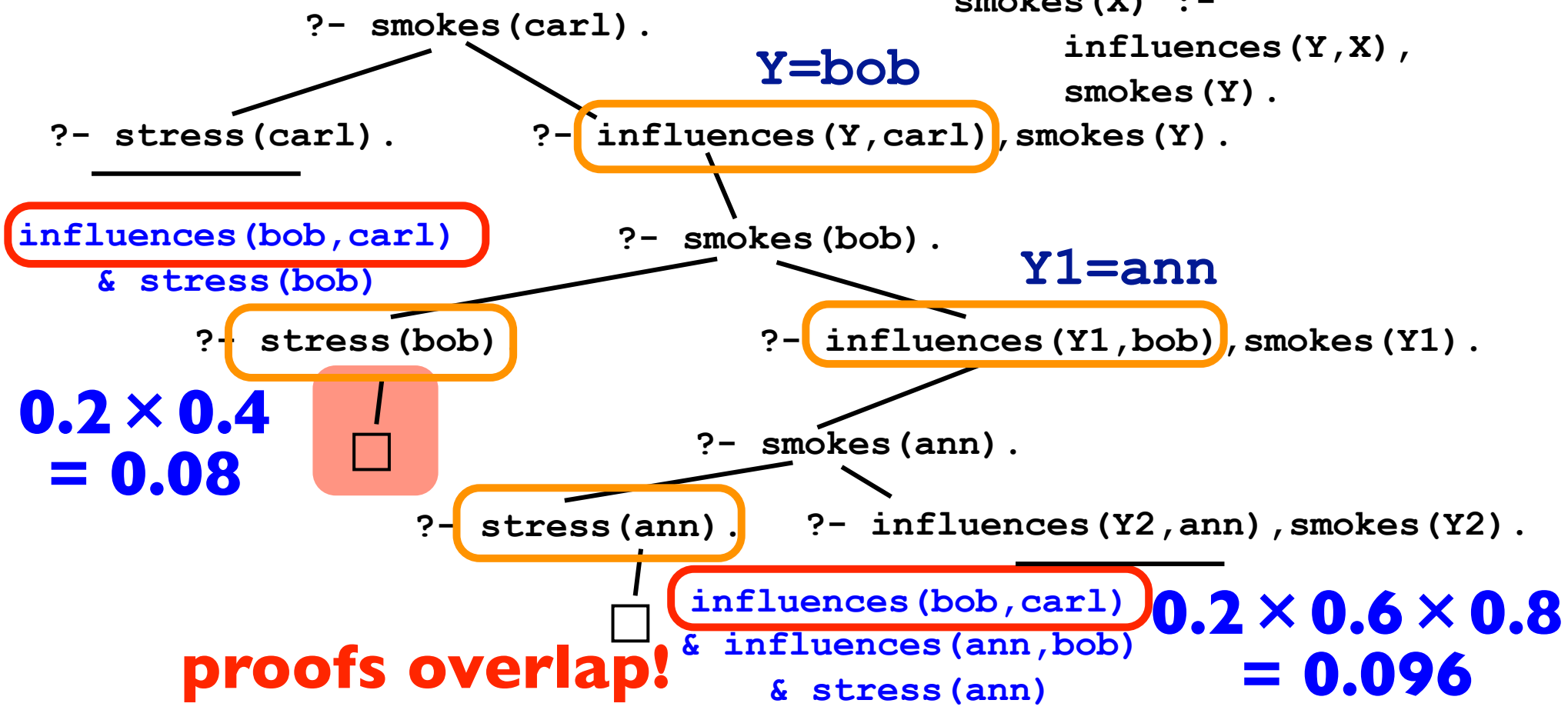


influences(bob,carl)&influences(ann,bob)&stress(ann)

probability of proof = 0.2 × 0.6 × 0.8 = 0.096

(Probabilistic) Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

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



proofs overlap!

cannot sum probabilities

(disjoint-sum-problem)

A prominent solution: knowledge compilation

So we are facing the Disjoint-Sum-Problem

possible worlds

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

`infl(bob,carl) & infl(ann,bob) & st(ann) & \+st(bob)`	0.0576
`infl(bob,carl) & infl(ann,bob) & st(ann) & st(bob)`	0.0384
`infl(bob,carl) & \+infl(ann,bob) & st(ann) & st(bob)`	0.0256
`infl(bob,carl) & infl(ann,bob) & \+st(ann) & st(bob)`	0.0096
`infl(bob,carl) & \+infl(ann,bob) & \+st(ann) & st(bob)`	0.0064
	∑ =
	0.1376

`influences(bob,carl) & stress(bob)`

...

sum of proof probabilities: 0.096+0.08 = 0.1760

Another Approach
(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```

Find relevant ground program for queries & evidence **win**

Weighted CNF

use weighted model counting / satisfiability

```
win :- heads(1).
win :- heads(2), heads(3).
```

win ↔ h(1) ∨ (h(2) ∧ h(3))

(¬win ∨ h(1) ∨ h(2))
∧ (¬win ∨ h(1) ∨ h(3))
∧ (win ∨ ¬h(1))
∧ (win ∨ ¬h(2) ∨ ¬h(3))

use standard solver for weighted CNF

h(1) → 0.4 h(2) → 0.7 h(3) → 0.5
¬h(1) → 0.6 ¬h(2) → 0.3 ¬h(3) → 0.5

[Fierens et al, TPLP 14]

But how do we convert from ProbLog to CNF?

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

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

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

- Convert to propositional logic formula using Clark's completion (a kind of CWA for negation as failure)

may require loop-breaking

$$sm(c) \leftrightarrow (i(b,c) \wedge sm(b))$$
$$\wedge\ sm(b) \leftrightarrow (st(b) \vee (i(a,b) \wedge sm(a)))$$
$$\wedge\ sm(a) \leftrightarrow st(a)$$

- Rewrite in CNF (as usual)

Now, run Weighted Model Counting

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

for p::f,

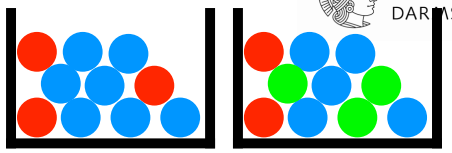
w(f) = p

w(not f) = 1−p

Indeed, there are other Probabilistic Programming Languages outside LP

- IBAL [Pfeffer 01]
- Figaro [Pfeffer 09]
- Church [Goodman et al 08]
- BLOG [Milch et al 05]
- Venture [Mansingha et al.]
- Anglican and Probabilistic-C [Wood et al].
- and many more appearing recently

For instance, Church: A bit of gambling revisited



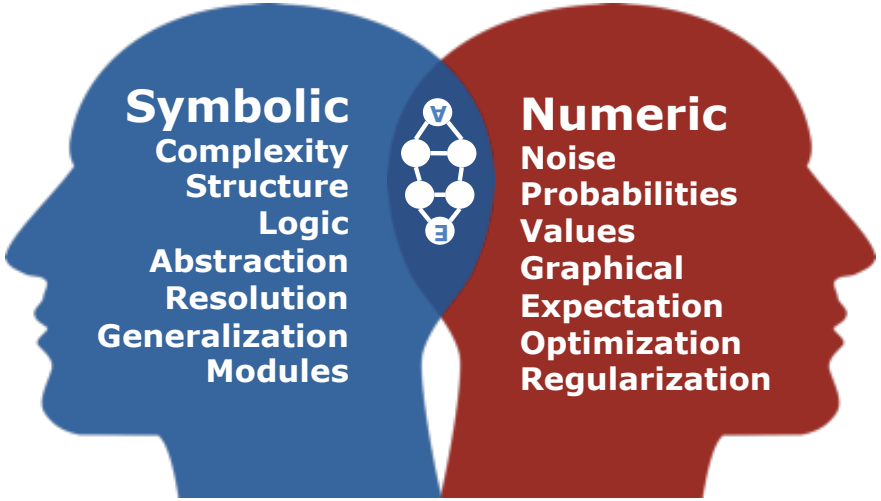
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                  (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))
```

Statistical Relational AI

Statistical Relational Learning

TECHNISCHE UNIVERSITÄT DARMSTADT



Kristian Kersting

Thanks to Vincent Conitzer, Rina Dechter, Luc De Raedt, Pedro Domingos, Peter Flach, Dieter Fensel, Florian Ficher, Vibhav Gogate, Carlos Guestrin, Daphen Koller, Nir Friedman, Ray Mooney, Sriraam Natarajan, David Poole, Fabrizio Riguzzi, Dan Suciu, Guy van den Broeck, and many others for making their slides publically available

Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

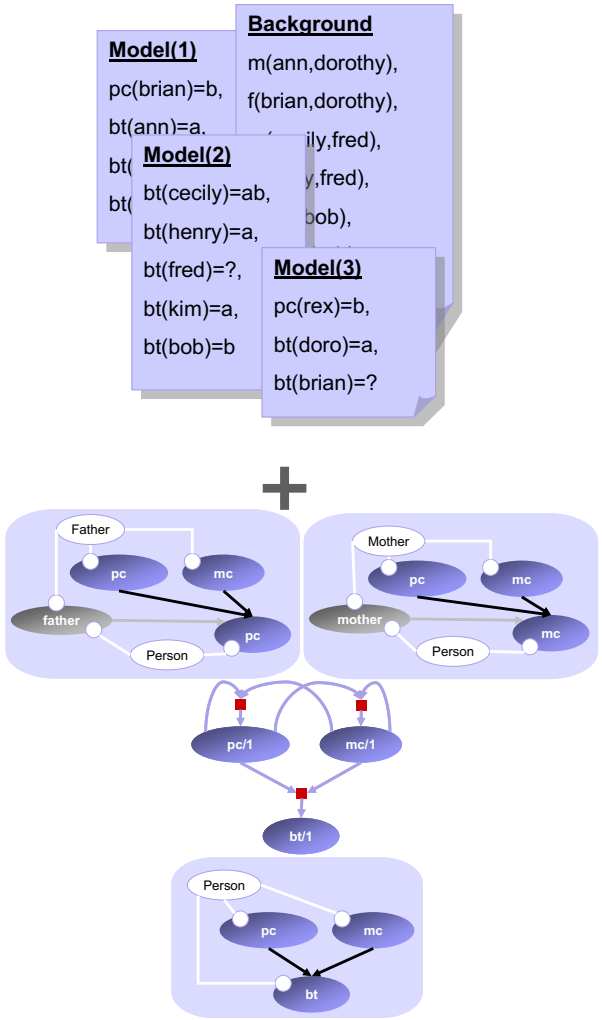
- Parameter Learning – Where do the numbers come from
- Structure Learning – neither logic program nor models are fixed
- Evidence
  - Partial assignments of values to variables
    {*burglary = false, earthquake = true, alarm = ?, johncalls = ?, marycalls = true*}

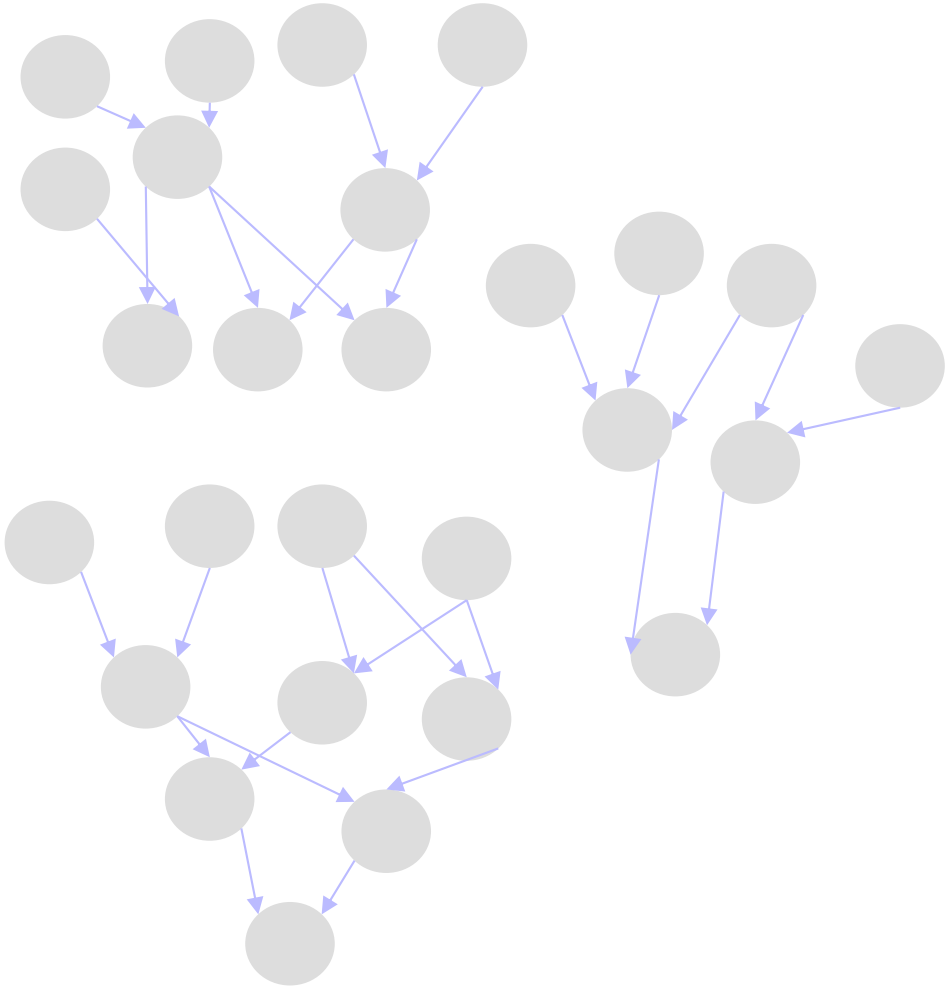
Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

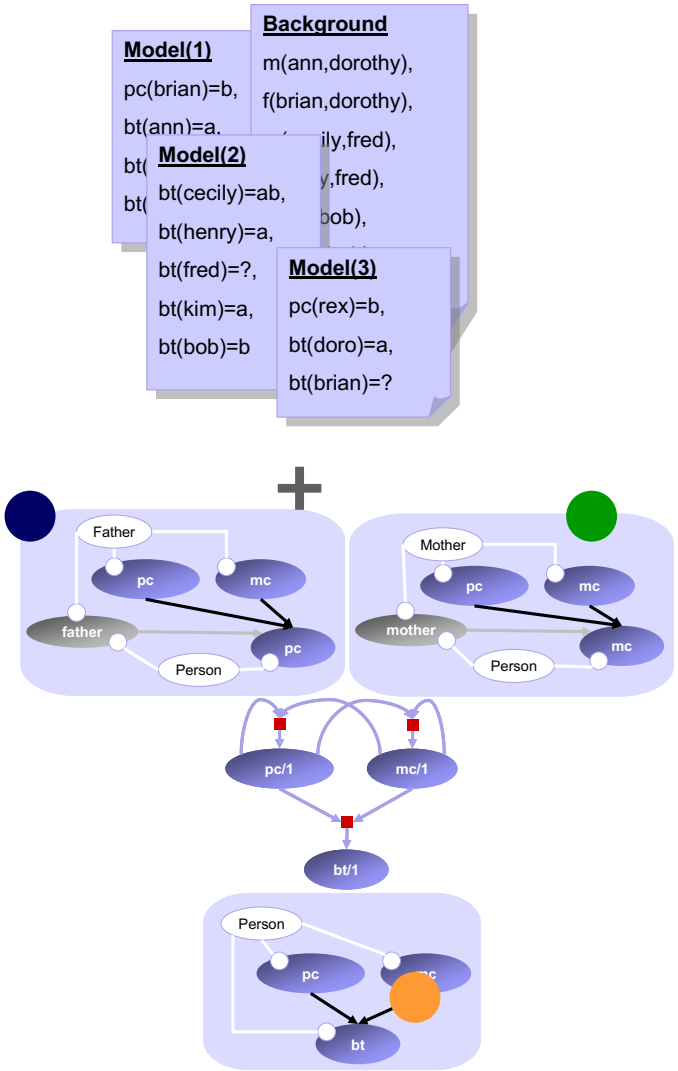
- Parameter Learning – Where do the numbers come from
- Structure Learning – neither logic program nor models are fixed
- Evidence
  - Partial assignments of values to variables
    {*burglary = false, earthquake = true, alarm = ?, johncalls = ?, marycalls = true*}

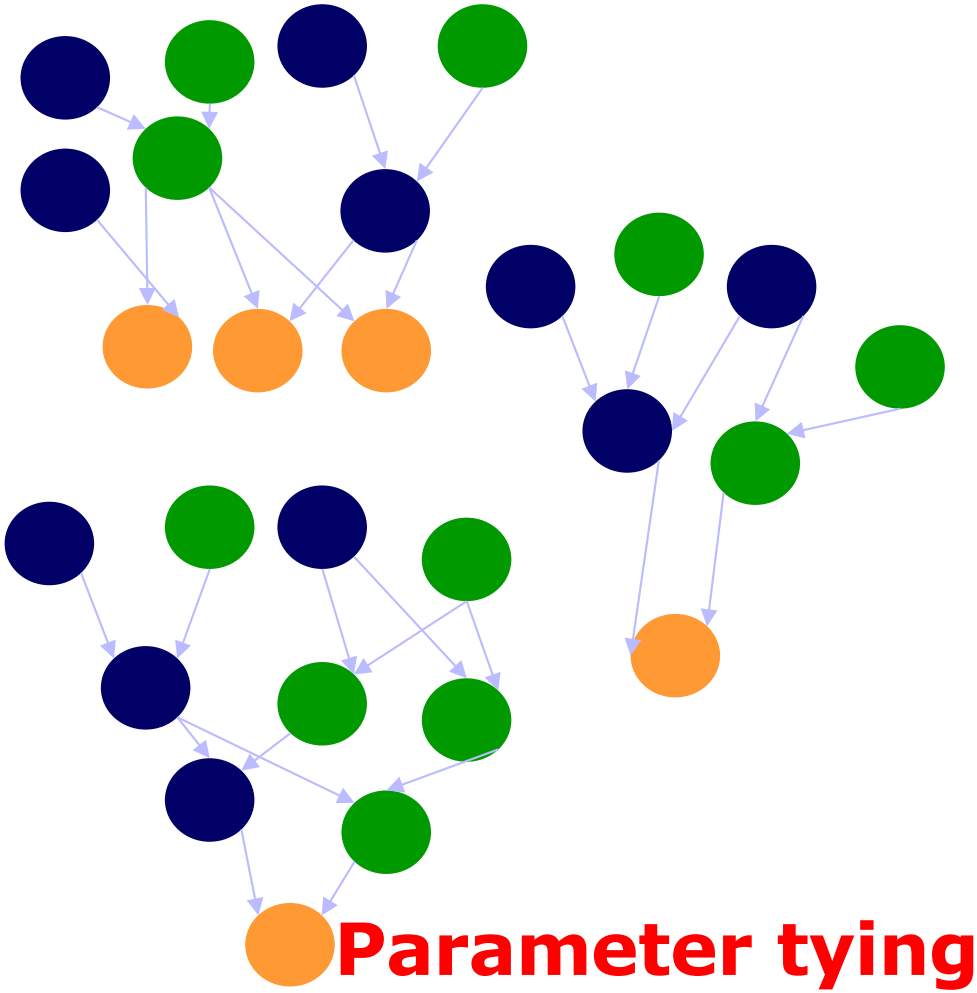
Relational Parameter Estimation





Relational Parameter Estimation

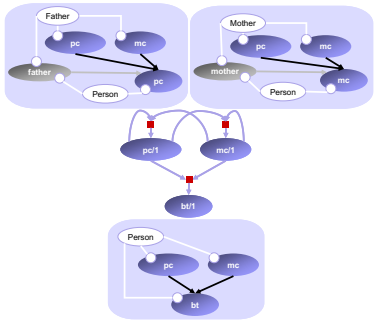




So, apply „standard“ EM

TECHNISCHE UNIVERSITÄT

Logic Program L



Initial Parameters q0

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)=?

iterate until convergence

Expectation

Current Model (M,qk)

Inference

Expected counts of a clause

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

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

Maximization

Update parameters (ML, MAP)

MLN Weight Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

- It is #P-complete to count the number of true groundings
- Generative learning: Pseudo-likelihood
- Discriminative learning: Cond. likelihood

Slide based on Domingos' talk

Vanilla SRL Approach [De Raedt, Kersting ALT04]

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

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

=0.882

...

- Traverses the hypotheses space a la ILP
- Replaces ILP's 0-1 covers relation by a "smooth", probabilistic one [0,1]

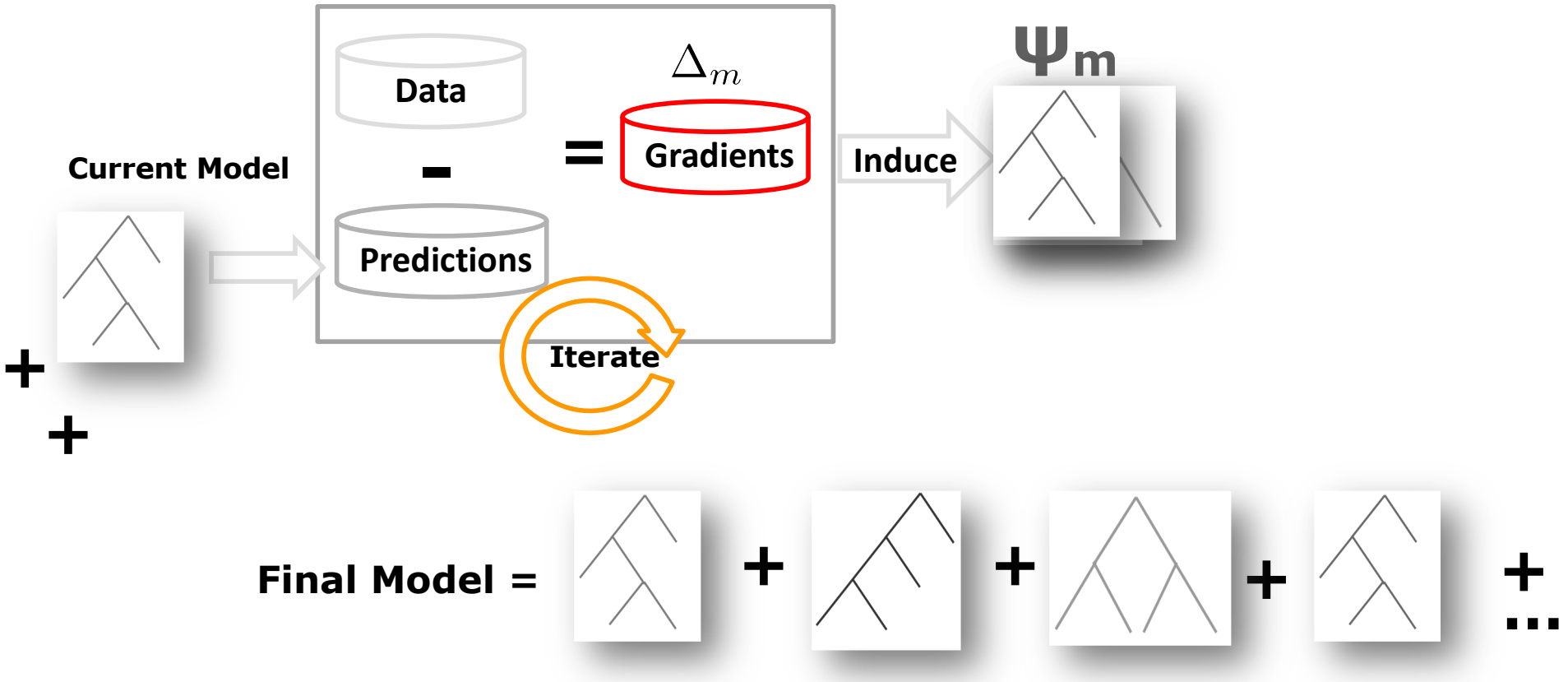
$$\text{cover}(e, H, B) = P(e|H, B)$$

$$\text{cover}(E, H, B) = \prod_{e \in E} \text{cover}(e, H, B)$$

Functional Gradient Boosting

TECHNISCHE UNIVERSITÄT DARMSTADT

Learn multiple weak models rather than a single complex model



Friedman et al 2001, Dietterich et al. 2004, Natarajan et al. MLJ 2012

Functional Gradients for SRL Models (here for (relational) dependency networsks

- Probability of an example

$$P(x_i = true|\mathbf{Pa}(x_i)) = \frac{e^{\psi(x_i;\mathbf{Pa}(x_i))}}{e^{\psi(x_i;\mathbf{Pa}(x_i))} + 1}$$

x	Δ
target(x1)	0.7
target(x2)	-0.2
target(x3)	-0.9

- Functional gradient
  - **Maximize**

$$LL(\mathbf{X} = \mathbf{x}) = \sum_{x_i \in \mathbf{x}} \log P(x_i|\mathbf{Pa}(x_i))$$

  - Gradient of log-likelihood w.r.t **ψ**

$$\Delta(x_i) = \frac{\partial \log P(\mathbf{X} = \mathbf{x})}{\partial \psi(x_i; \mathbf{Pa}(x_i))} = I(x_i = true; \mathbf{Pa}(x_i)) - P(x_i = true; \mathbf{Pa}(x_i))$$

  - Sum all gradients to get final **ψ**

$$\psi_m = \psi_0 + \Delta_1 + ... + \Delta_m$$

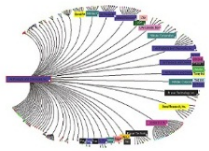
Extended to multiple SRL models & in presence of hidden data

Impressive Results on Standard Domains

Predicting the advisor for a student

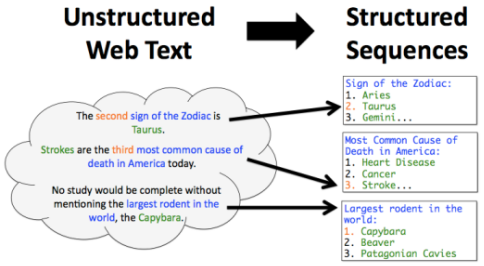
Algo	Likelihood	AUC-ROC	AUC-PR	Time
Boosting	0.810	0.961	0.930	9s
RPT	0.805	0.894	0.863	1s
MLN	0.730	0.535	0.621	93 hrs

Movie Recommendation



Citation Analysis

Learning from Demonstrations



Information Extraction

Scale of Learning Structure
- **150 k facts describing the citations**
- **115k drug-disease interactions**
- **11 M facts on NLP tasks**

Natarajan et al. MLJ'12, Khot et al. ICDM '11, Natarajan et al. IJCAI '11, Natarajan et al. IAAI '13
Weiss et al. IAAI '12 AI Magazine '12, Natarajan et al. IJMLC '13, Khot et al. MLJ' 14

What have we learnt?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Early learning methods extended standard graphical model learning
- Parameter tying exploited when learning relational models
- Structure learning is a harder task due to searching multiple levels of abstraction
- Local search/boosting methods are popular for learning multiple SRL models