DTAI

MACHINE LEARNING

Capita Selecta AI:

Probabilistic Programming

Wannes Meert

7/12/2016

Slides courtesy of Angelika Kimmig, Luc De Raedt and Guy Van den Broeck

KU LEUVEN

Today

- Assignment
- Lifted Inference
- Probabilistic Databases

Assignment

Topics

- Probabilistic Inference Using Weighted Model Counting
  - PGM to CNF
  - SRL to CNF
  - Weighted Model Counting
  - Knowledge Compilation
- Build an Inference Engine
- Comparison with Approximate Inference

ProbLog

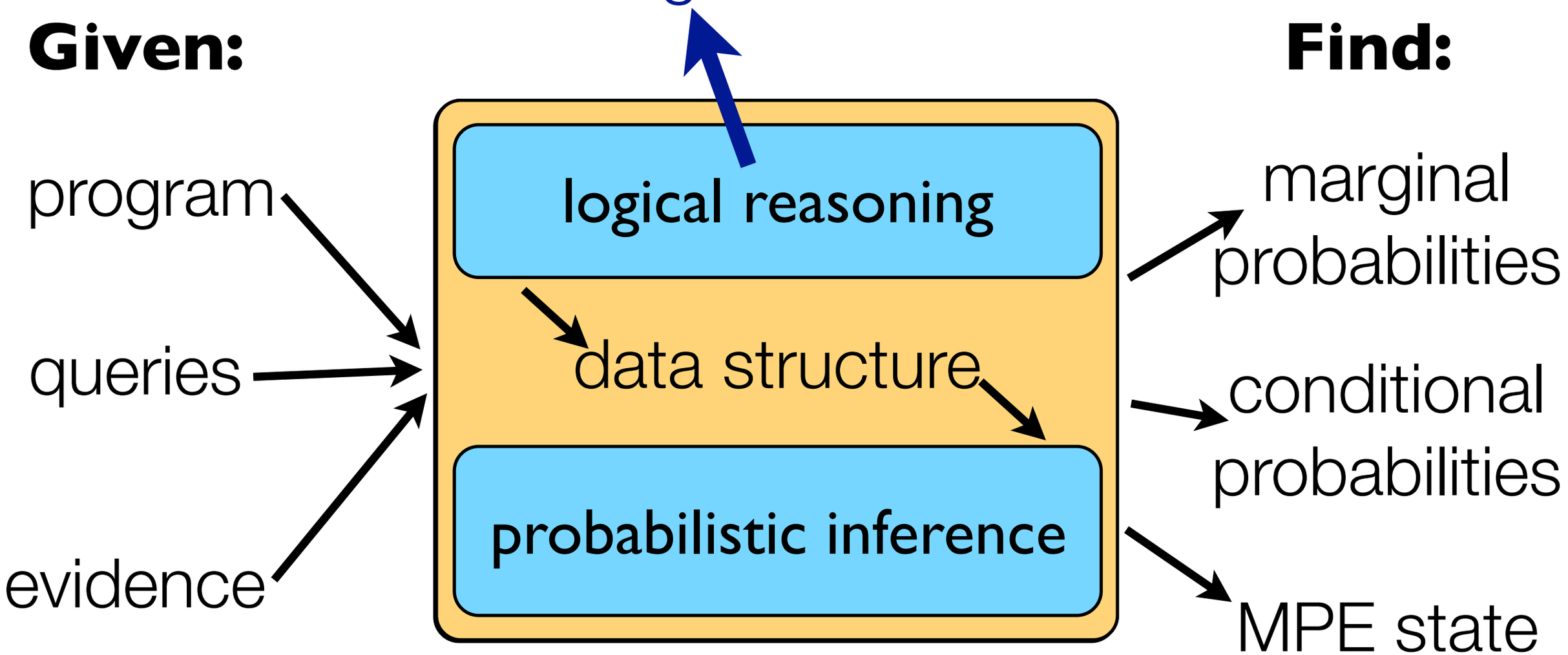
Short recap

Answering Questions

1. using proofs
2. using models

Given:

Find:



Weighted Model Counting

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

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

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

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

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

interpretations (truth value assignments) of propositional variables

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

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

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

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

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

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

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

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

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

for p::f,

w(f) = p

w(not f) = 1–p

Weight

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

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

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

interpretations (truth value assignments) of propositional variables

possible worlds

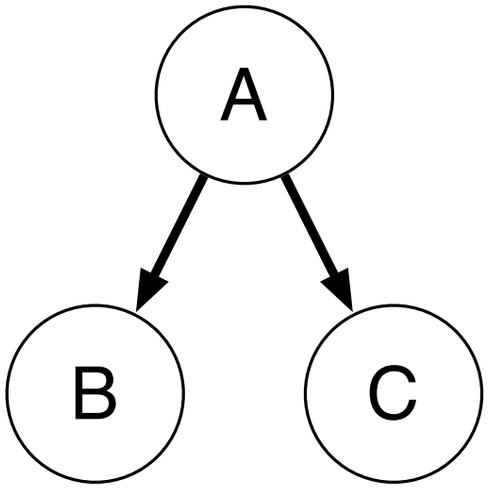
weight of literal

for p::f,

w(f) = p

w(not f) = 1–p

Bayesian net to WMC



A	Pr
a_1	0.1
a_2	0.9

A	B	Pr
a_1	b_1	0.1
a_1	b_2	0.9
a_2	b_1	0.2
a_2	b_2	0.8

A	C	Pr
a_1	c_1	0.1
a_1	c_2	0.2
a_1	c_3	0.7
a_2	c_1	0.01
a_2	c_2	0.09
a_2	c_3	0.9

Pr(A|B=1,C=1) ~ Pr(A=1,B=1,C=1) + Pr(A=2,B=1,C=1)
= 0.1·0.1·0.1 + 0.9·0.2·0.01
2 models

Variable A: $\lambda_{a_1} \vee \lambda_{a_2}$ $\quad \neg\lambda_{a_1} \vee \neg\lambda_{a_2}$

Variable B: $\lambda_{b_1} \vee \lambda_{b_2}$ $\quad \neg\lambda_{b_1} \vee \neg\lambda_{b_2}$

Variable C: $\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}$ $\quad \neg\lambda_{c_1} \vee \neg\lambda_{c_2}$ $\quad \neg\lambda_{c_1} \vee \neg\lambda_{c_3}$ $\quad \neg\lambda_{c_2} \vee \neg\lambda_{c_3}$

CPT 1: $\lambda_{a_1} \Leftrightarrow \theta_{a_1}$ $\quad \lambda_{a_2} \Leftrightarrow \theta_{a_2}$

CPT 2: $\lambda_{a_1} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1|a_1}$ $\quad \lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1|a_2}$

$\lambda_{a_1} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2|a_1}$ $\quad \lambda_{a_2} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2|a_2}$

CPT 3: $\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_1}$ $\quad \lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2|a_1}$ $\quad \lambda_{a_2} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3|a_1}$ $\quad \lambda_{a_2} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3|a_2}$

$W(\theta_{a_1}) = 0.1$ $\quad W(\theta_{a_2}) = 0.9$

$W(\theta_{b_1|a_1}) = 0.1$ $\quad W(\theta_{b_2|a_1}) = 0.9$

$W(\theta_{b_1|a_2}) = 0.2$ $\quad W(\theta_{b_2|a_2}) = 0.8$

$W(\theta_{c_1|a_1}) = 0.1$ $\quad W(\theta_{c_2|a_1}) = 0.2$ $\quad W(\theta_{c_3|a_1}) = 0.7$

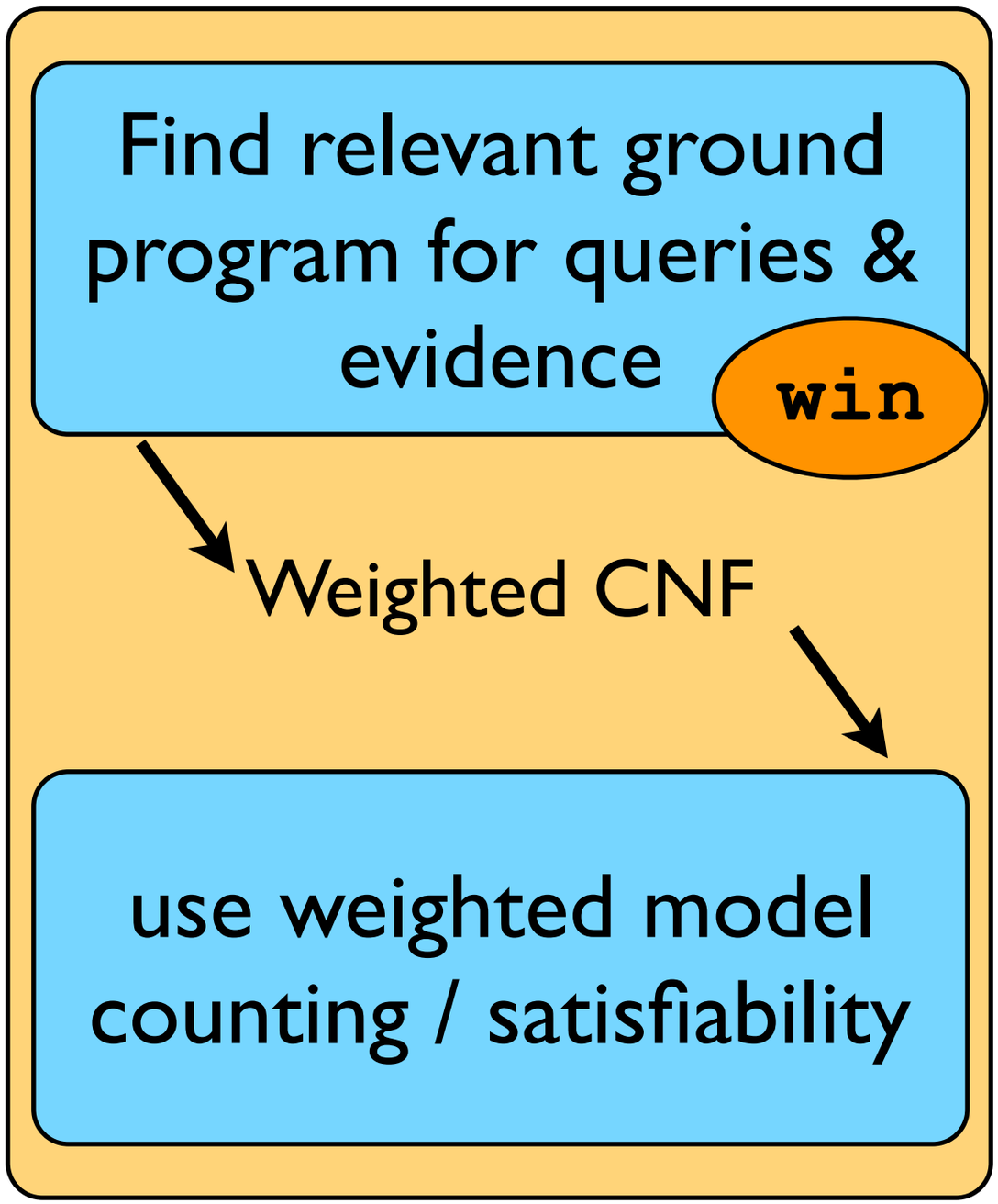
$W(\theta_{c_1|a_2}) = 0.01$ $\quad W(\theta_{c_2|a_2}) = 0.09$ $\quad W(\theta_{c_3|a_2}) = 0.9$

M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. Artificial Intelligence, 172(6):772–799, 2008.

KU LEUVEN

ProbLog inference

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




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

$$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$$

$$(\neg\text{win} \vee h(1) \vee h(2)) \wedge (\neg\text{win} \vee h(1) \vee h(3)) \wedge (\text{win} \vee \neg h(1)) \wedge (\text{win} \vee \neg h(2) \vee \neg h(3))$$

use standard tool

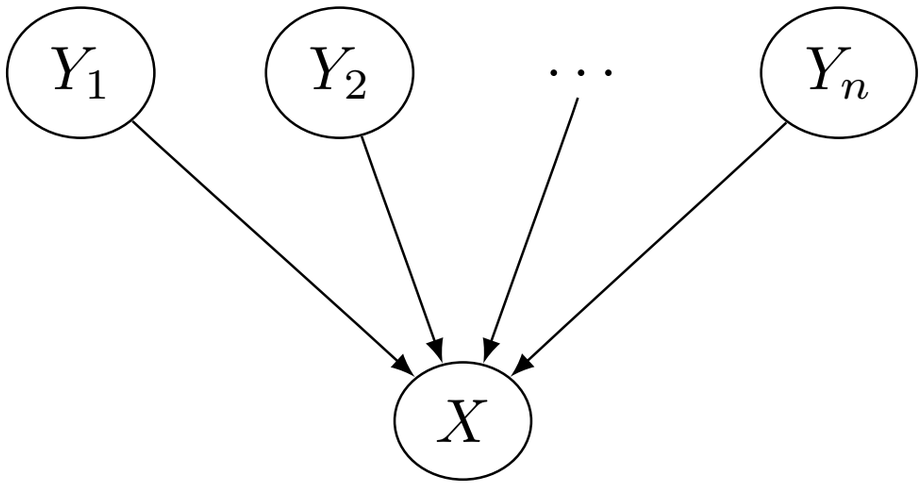
$h(1) \rightarrow 0.4$ $h(2) \rightarrow 0.7$ $h(3) \rightarrow 0.5$

$\neg h(1) \rightarrow 0.6$ $\neg h(2) \rightarrow 0.3$ $\neg h(3) \rightarrow 0.5$

[Fierens et al, TPLP 14]

Example: Encoding

Bayesian Network



Probabilistic Logic

```
0.4::y(1).
0.3::y(2).
…
0.5::y(n).
0.8::x :- y(N).
```

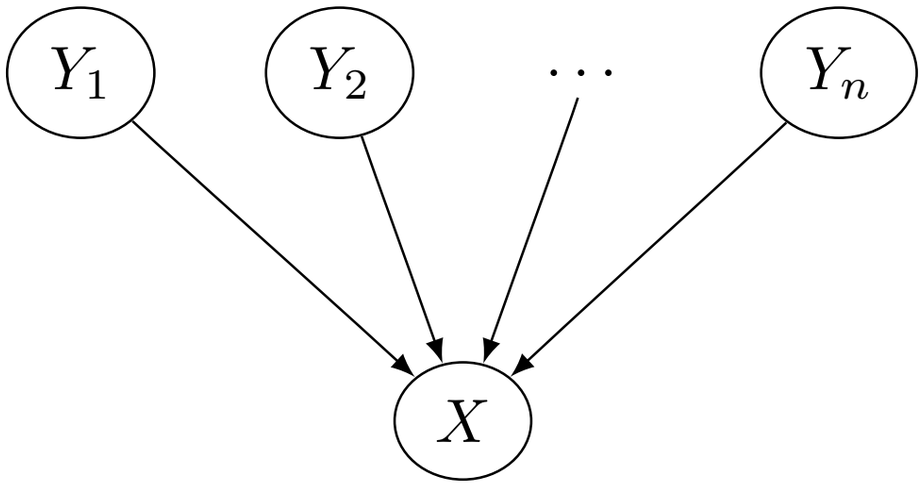
Propositional Logic

```
y(1) ⇔ p(1,1)
y(2) ⇔ p(1,2)
…
y(n) ⇔ p(1,n)
x ⇔ (y(1) ∧ p(2,1)) ∨ …
      ∨ (y(n) ∧ p(2,n))
```

```
w(p(1,1)) = 0.4
w(p(1,2)) = 0.3
…
w(p(1,n)) = 0.5
w(p(2,1)) = 0.8
…
w(p(2,n)) = 0.8
```

Example: Encoding

Bayesian Network



Probabilistic Logic

```
0.4::y(1).
0.3::y(2).
…
0.5::y(n).
0.8::x :- y(N).
```

Propositional Logic

```
y(1) ⇔ p(1,1)
y(2) ⇔ p(1,2)
…
y(n) ⇔ p(1,n)
x ⇔ (y(1) ∧ p(2,1)) ∨ …
      ∨ (y(n) ∧ p(2,n))
```

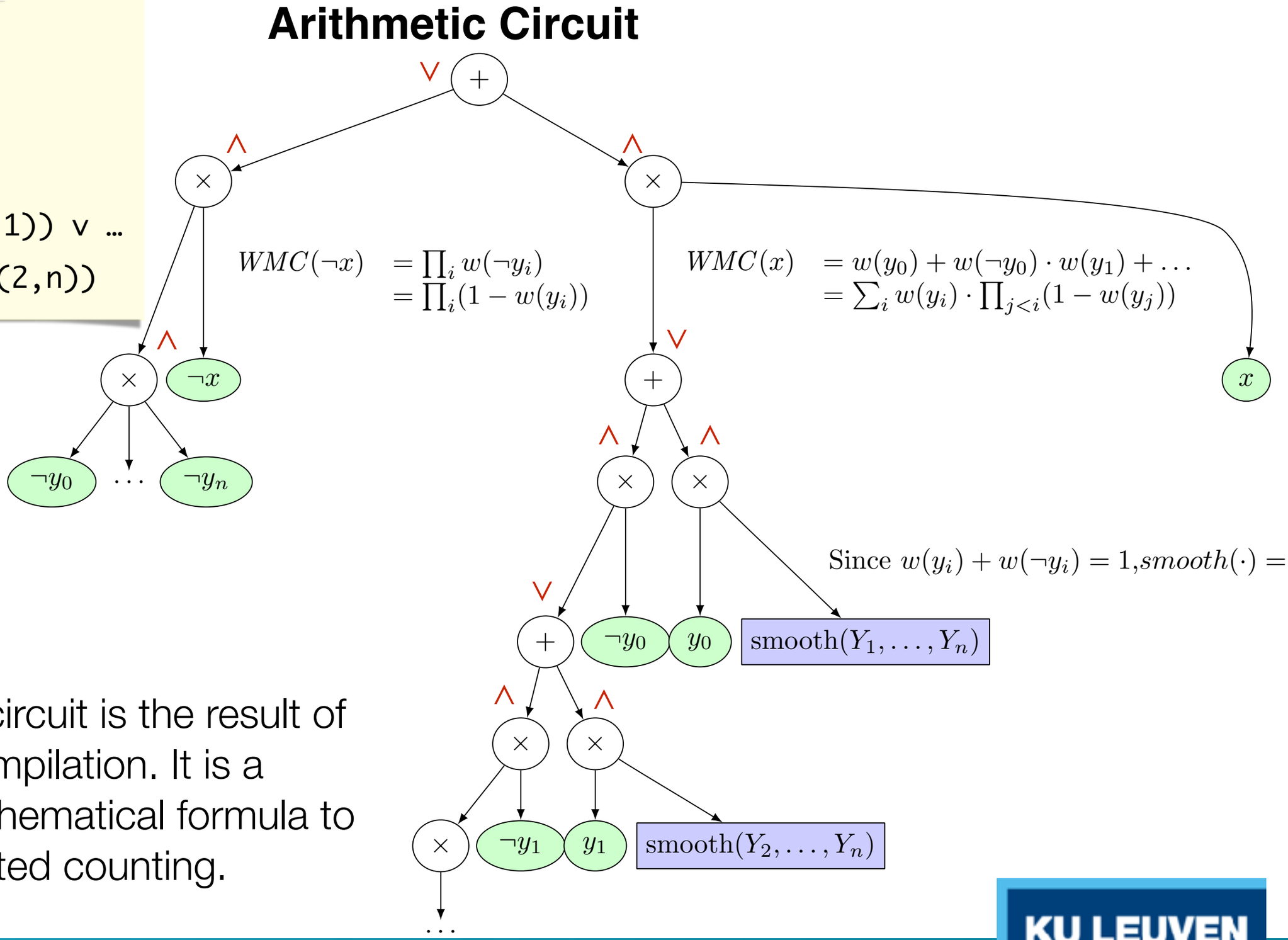
```
w(p(1,1)) = 0.4
w(p(1,2)) = 0.3
…
w(p(1,n)) = 0.5
w(p(2,1)) = 0.8
…
w(p(2,n)) = 0.8
```

Compact representation of full CPT

KU LEUVEN

Example: Weighted Model Counting

```
y(1) ⇔ p(1,1)
y(2) ⇔ p(1,2)

…
y(n) ⇔ p(1,n)
x ⇔ (y(1) ∧ p(2,1)) ∨ …
      ∨ (y(n) ∧ p(2,n))
```

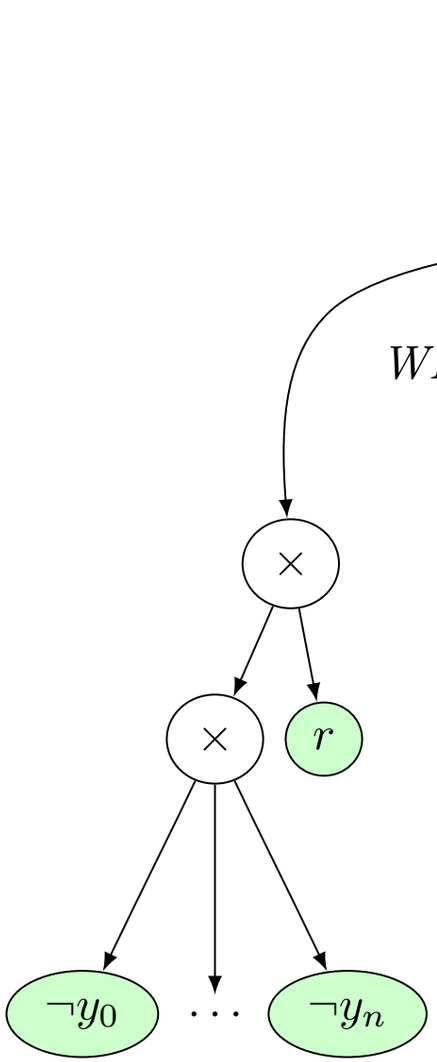


An arithmetic circuit is the result of knowledge compilation. It is a (compact) mathematical formula to perform weighted counting.

Example: Alternative Encoding

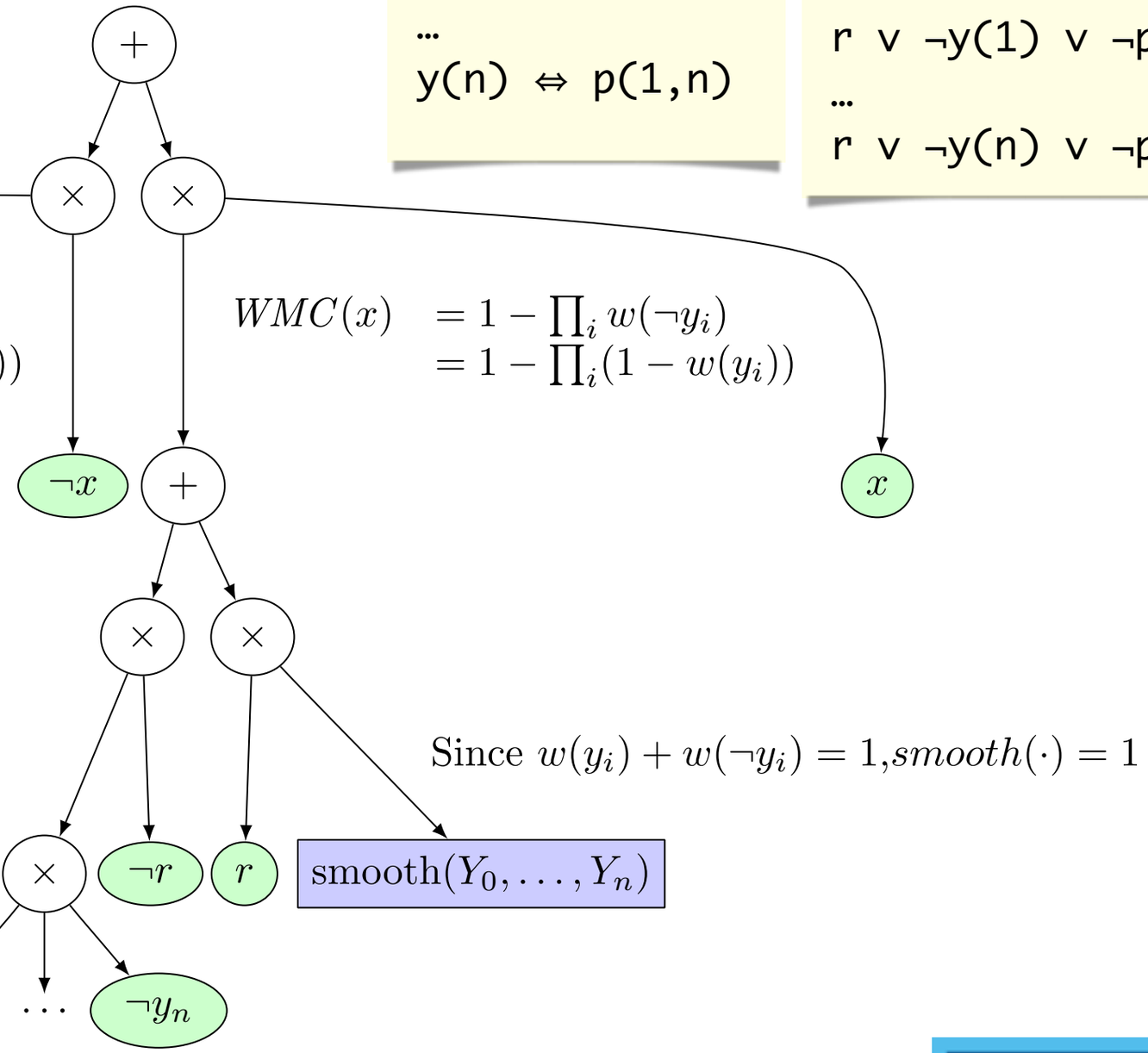
```
y(1) ⇔ p(1,1)
y(2) ⇔ p(1,2)
…
y(n) ⇔ p(1,n)
```

```
x ⇔ z
z v ¬y(1) v ¬p(2,1)
…
z v ¬y(n) v ¬p(2,n)
r v z
r v ¬y(1) v ¬p(2,1)
…
r v ¬y(n) v ¬p(2,n)
```



$$WMC(\neg x) = \prod_i w(\neg y_i) = \prod_i (1 - w(y_i))$$

$$WMC(x) = 1 - \prod_i w(\neg y_i) = 1 - \prod_i (1 - w(y_i))$$



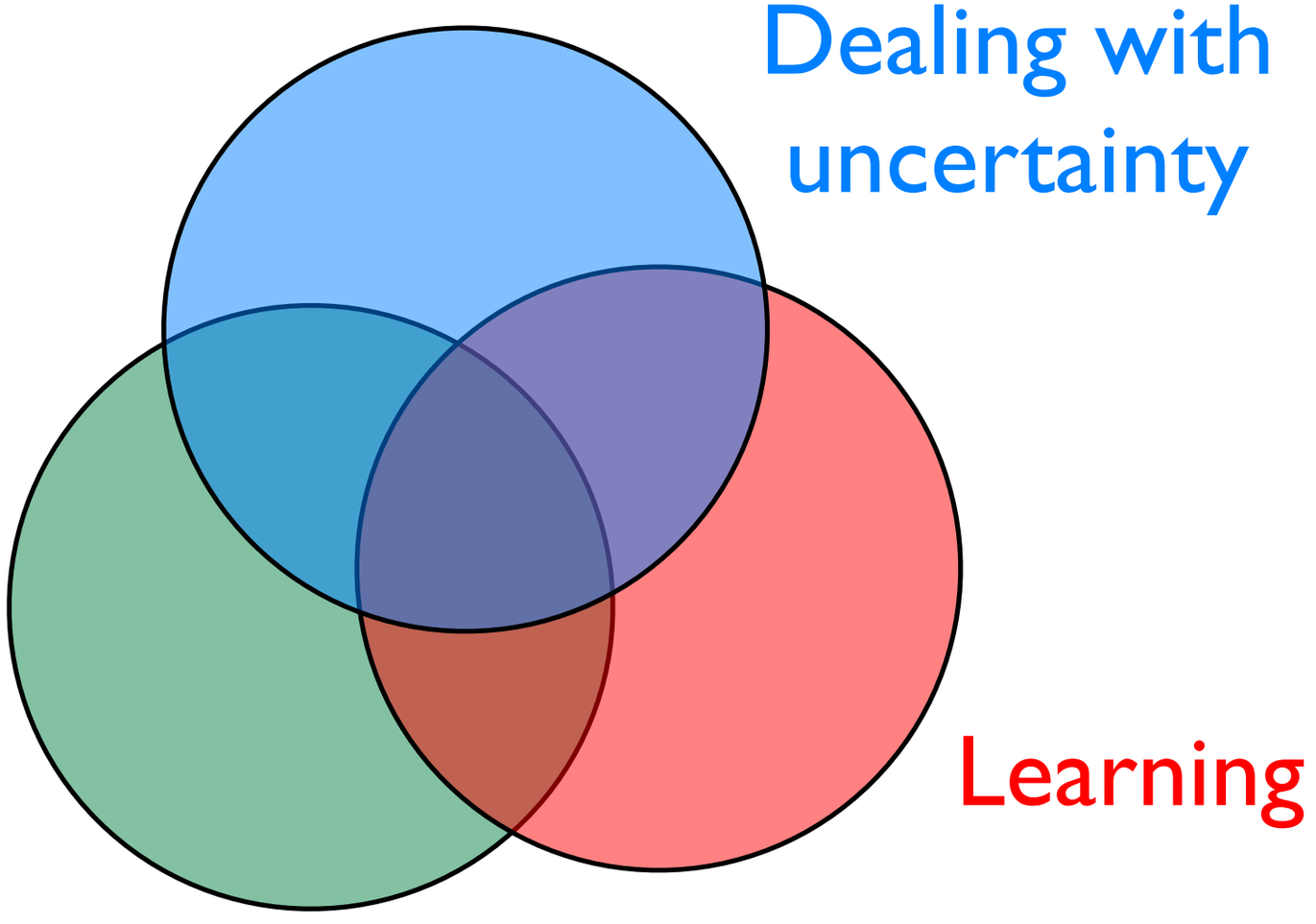
Since $w(y_i) + w(\neg y_i) = 1, smooth(\cdot) = 1$

G. Van den Broeck, W. Meert, and A. Darwiche. Skolemization for weighted first-order model counting.KR , 2014.
W. Meert, J. Vlasselaer, and G. Van den Broeck. A relaxed tseitin transformation for weighted model counting. StarAI, 2016.

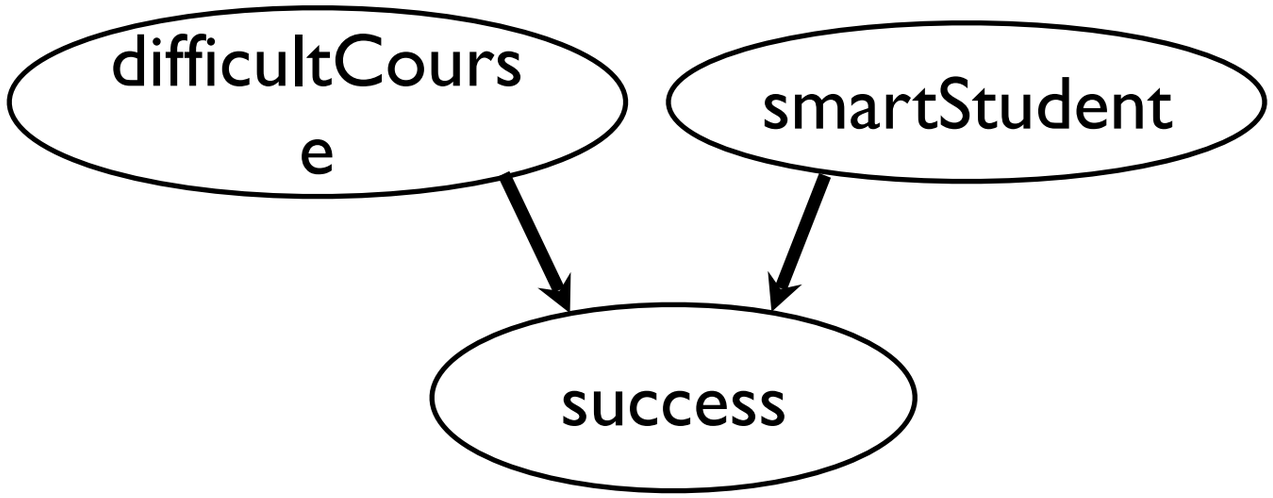
KU LEUVEN

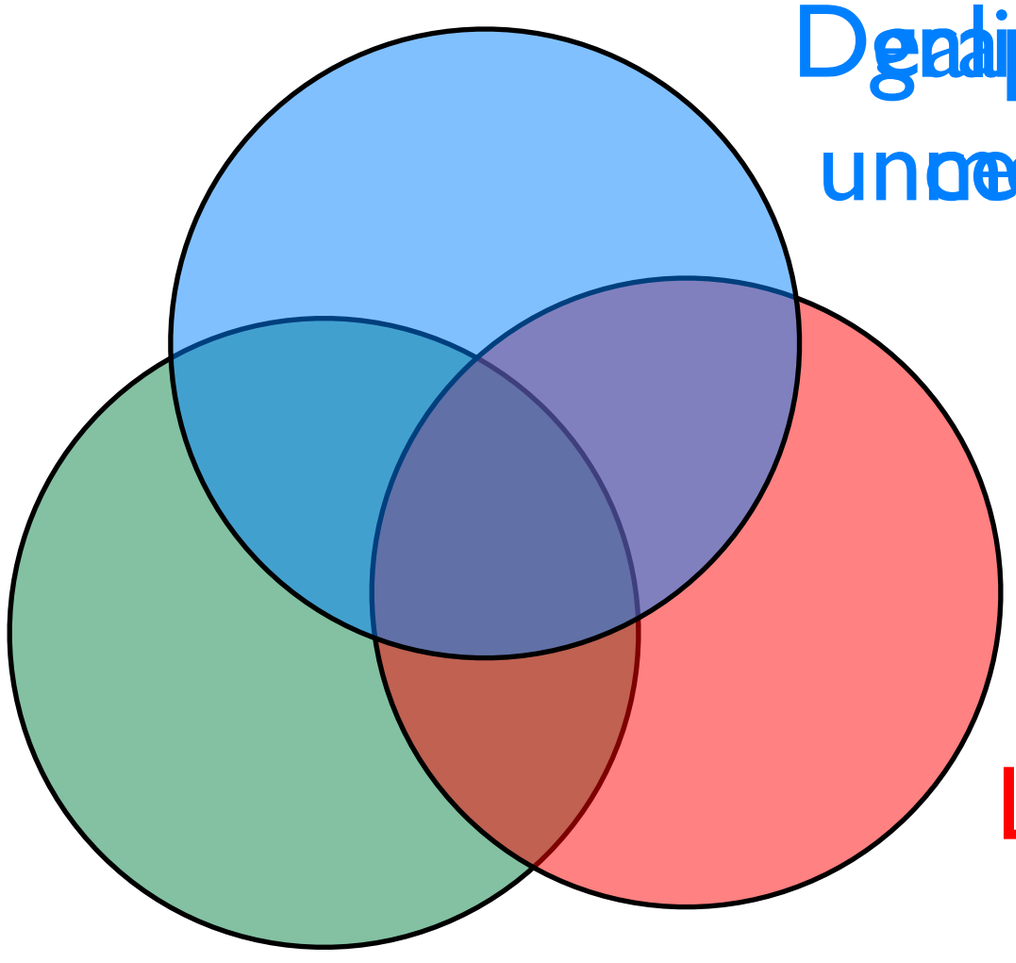
Lifted Graphical Models

Lifted graphical models



Lifted graphical models





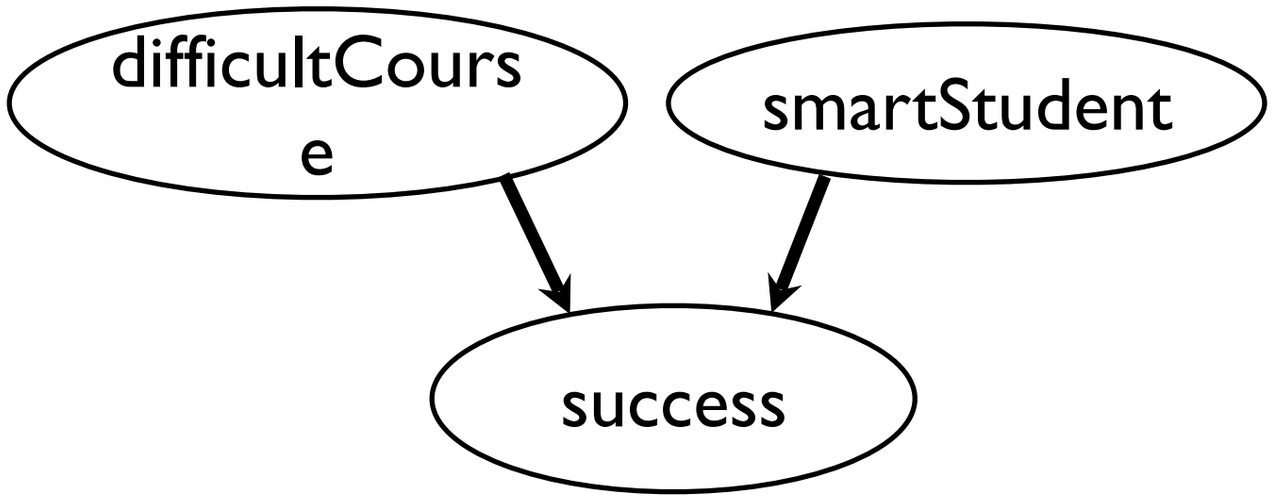
Dealing with uncertainty

Reasoning with relational data

Learning

Lifted graphical models

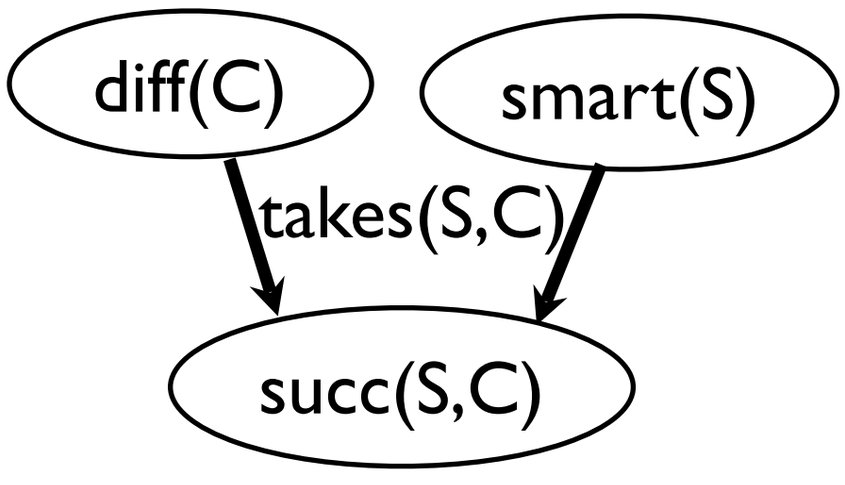
fixed set of random variables



Dealing with uncertainty

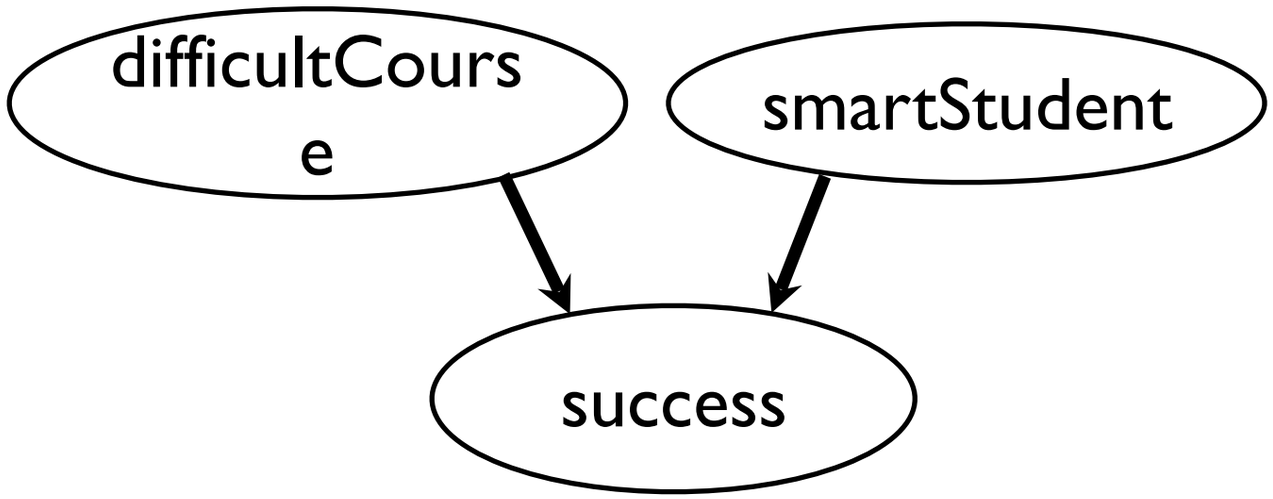
Reasoning about relational structure

Learning



Lifted graphical models

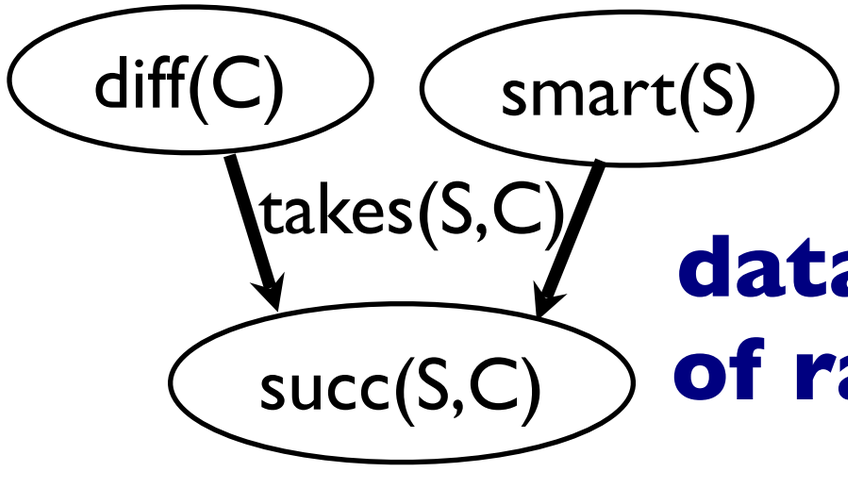
fixed set of random variables



Dealing with uncertainty

Reasoning with relational models

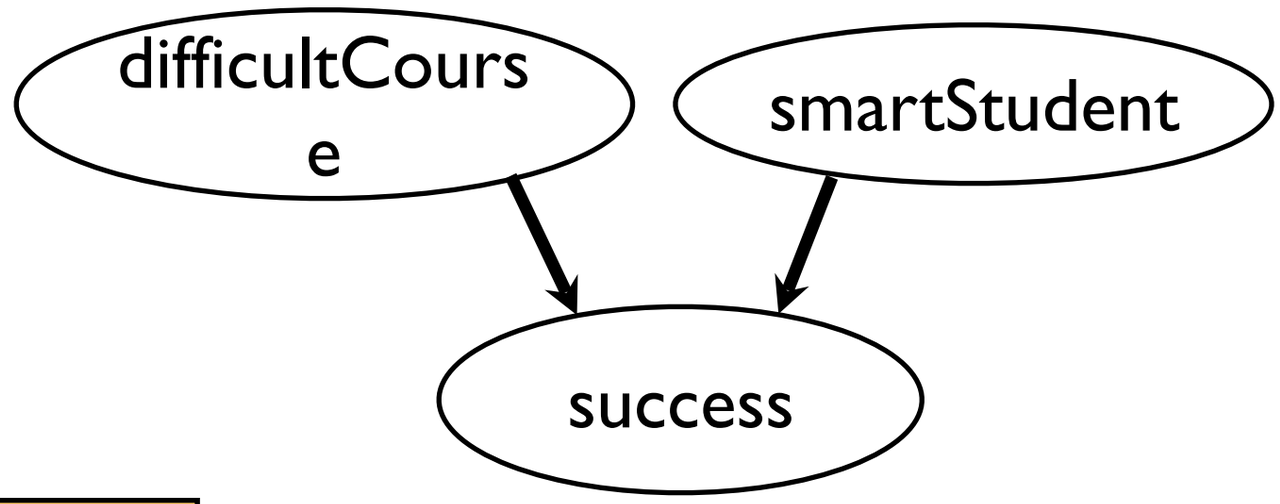
Learning



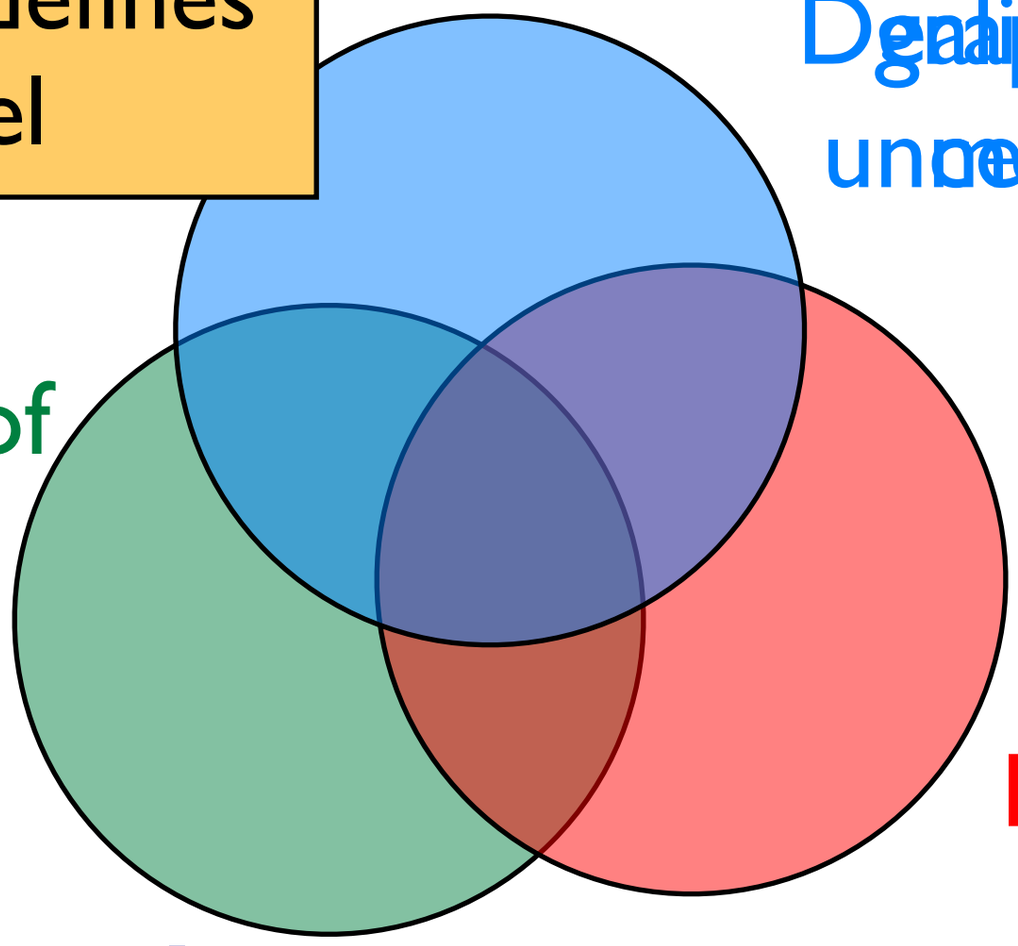
data-dependent set of random variables

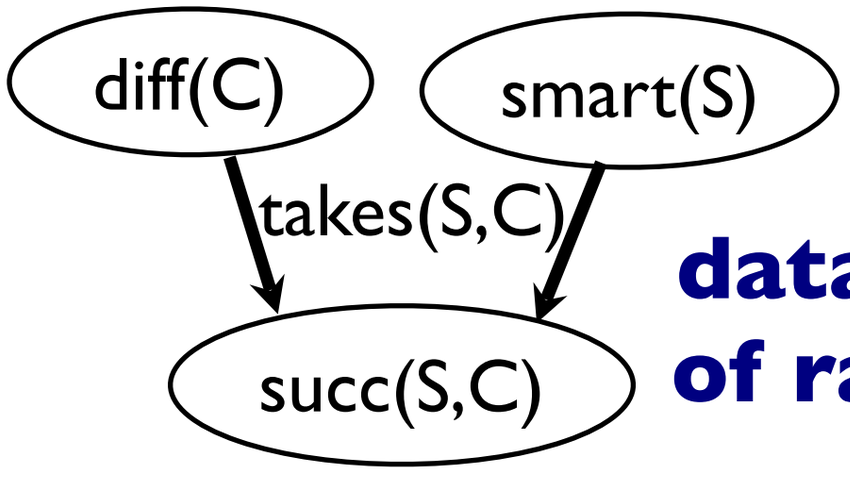
Lifted graphical models

fixed set of random variables



relational language defines graphical model

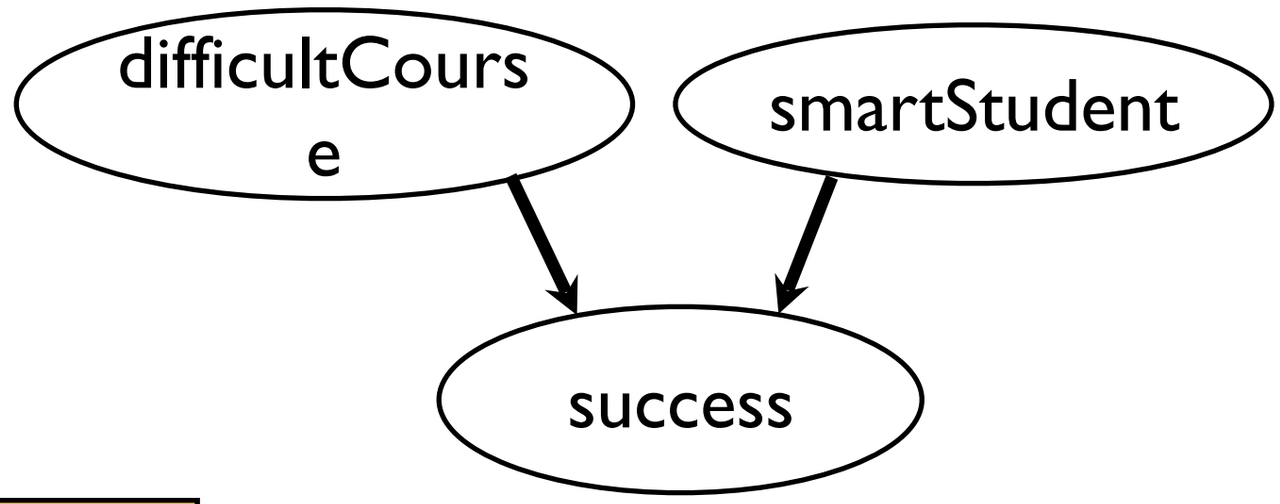




data-dependent set of random variables

Lifted graphical models

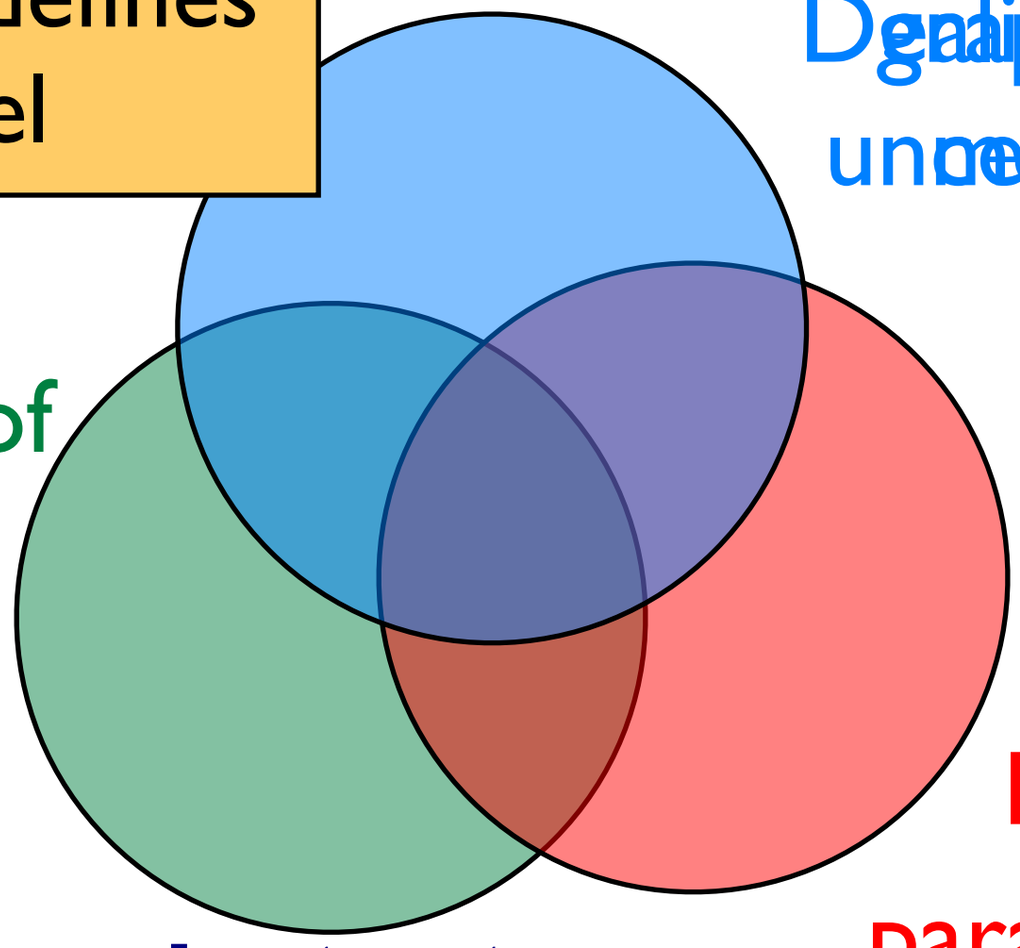
fixed set of random variables

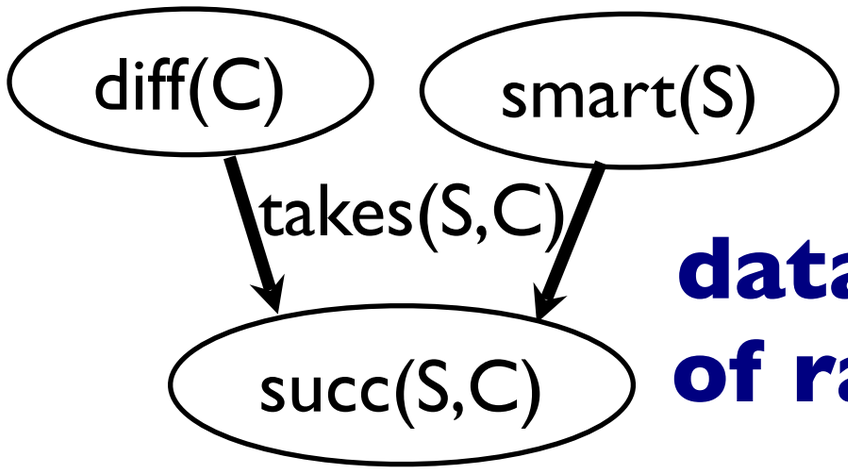


relational language defines graphical model

Dealing with uncertainty

Reasoning with relational structure





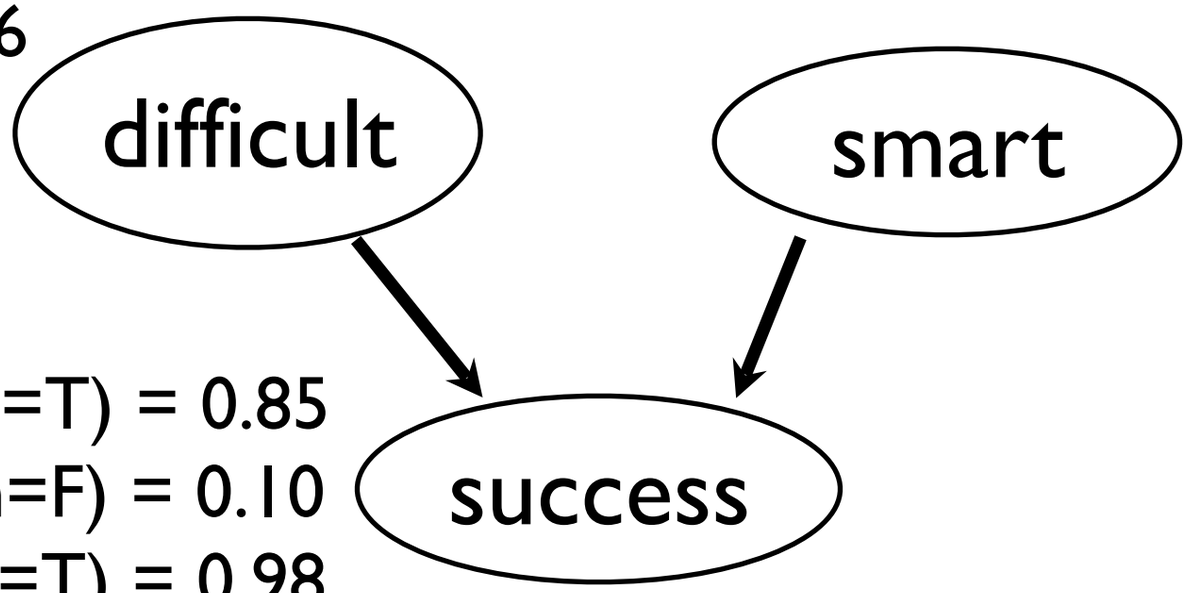
data-dependent set of random variables

Learning parameters & structure

Bayesian Network

P(difficult=T) = 0.6

P(smart=T) = 0.7



P(success=T|d=T,sm=T) = 0.85
P(success=T|d=T,sm=F) = 0.10
P(success=T|d=F,sm=T) = 0.98
P(success=T|d=F,sm=F) = 0.45

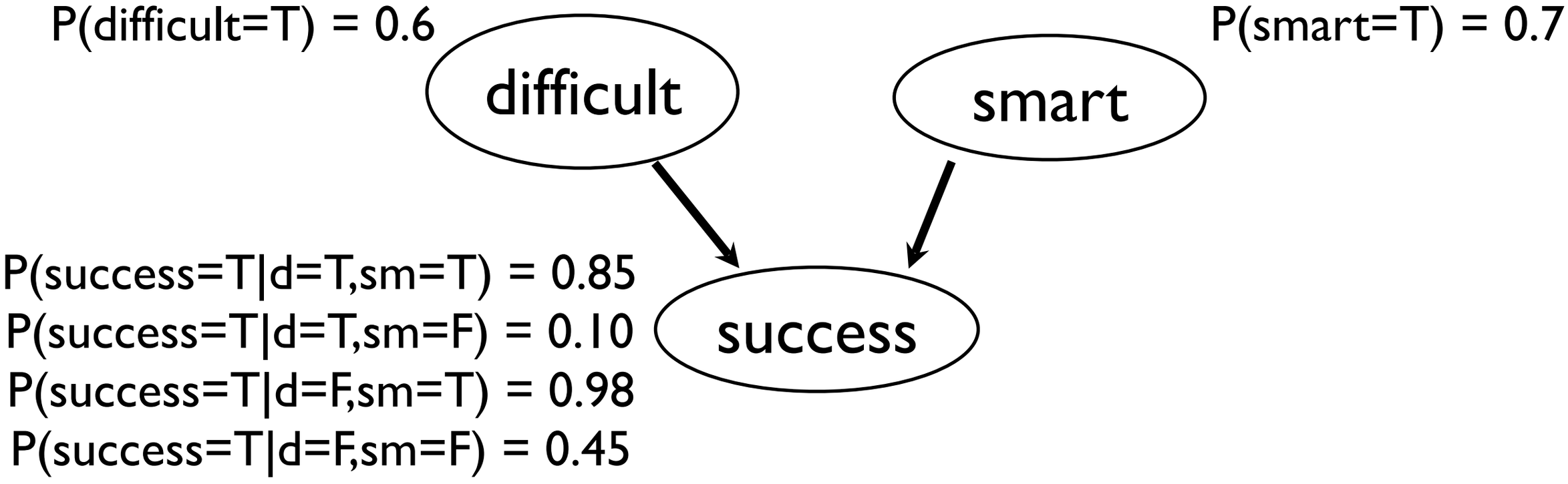
joint distribution

P(difficult)×P(smart)×P(success|difficult,smart)

directed acyclic graph

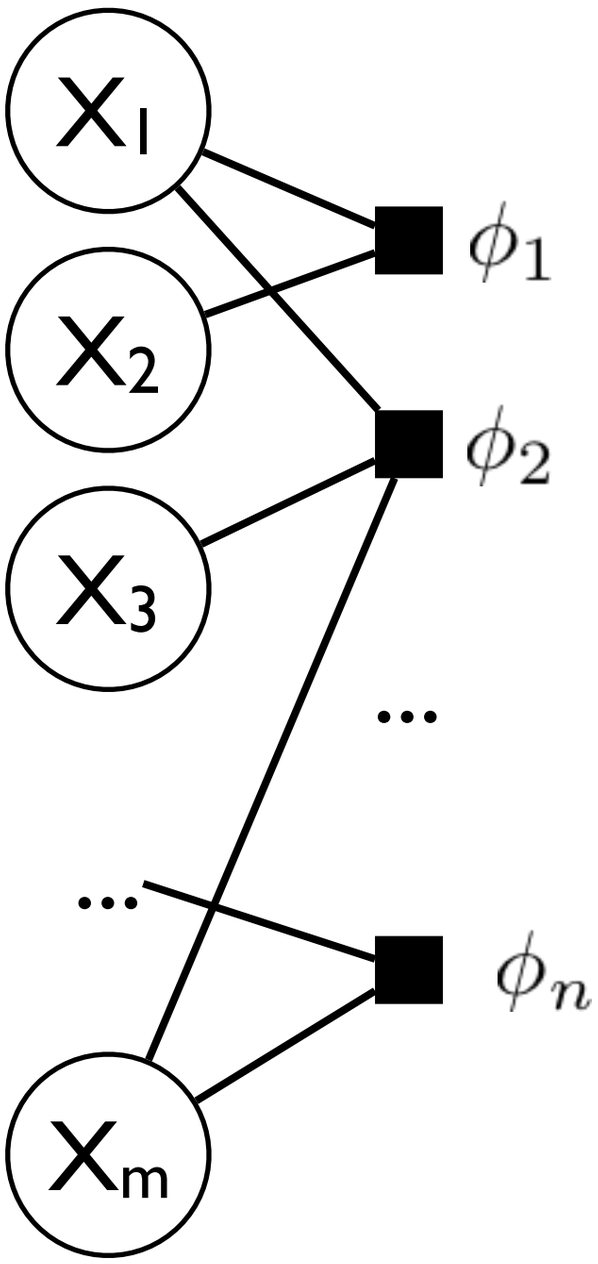
$$P(X_1, \ldots, X_n) = \prod_{i=1,\ldots,n} P(X_i|\mathrm{par}(X_i))$$

Bayesian Network in ProbLog?




```
0.6::difficult.
0.7::smart.
0.85::success :- difficult, smart.
0.10::success :- difficult, \+smart.
0.98::success :- \+difficult, smart.
0.45::success :- \+difficult, \+smart.
```

Factor Graph

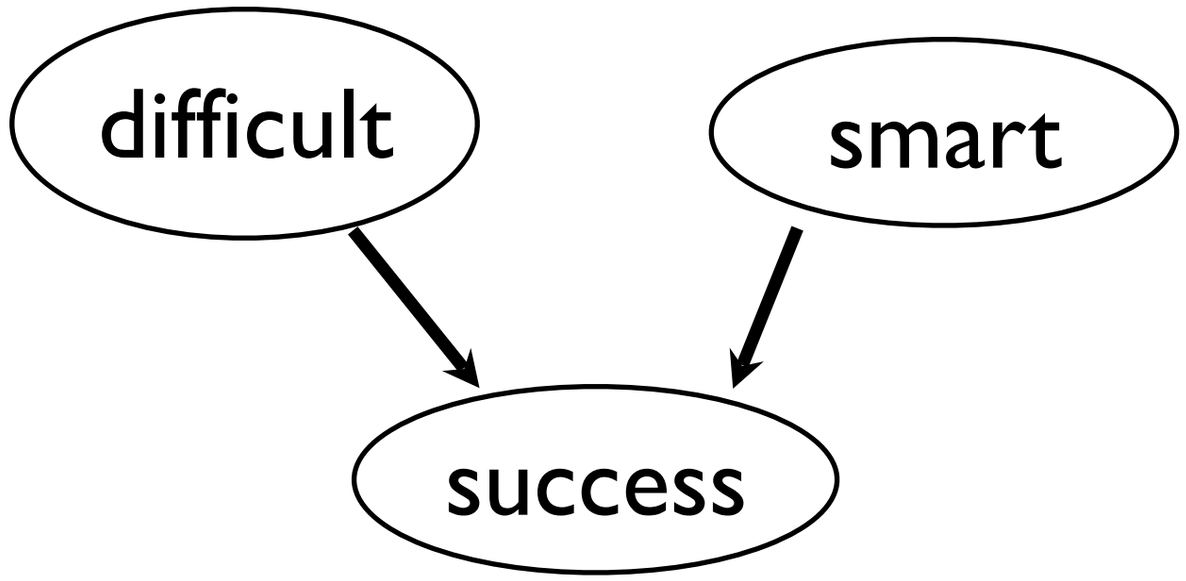


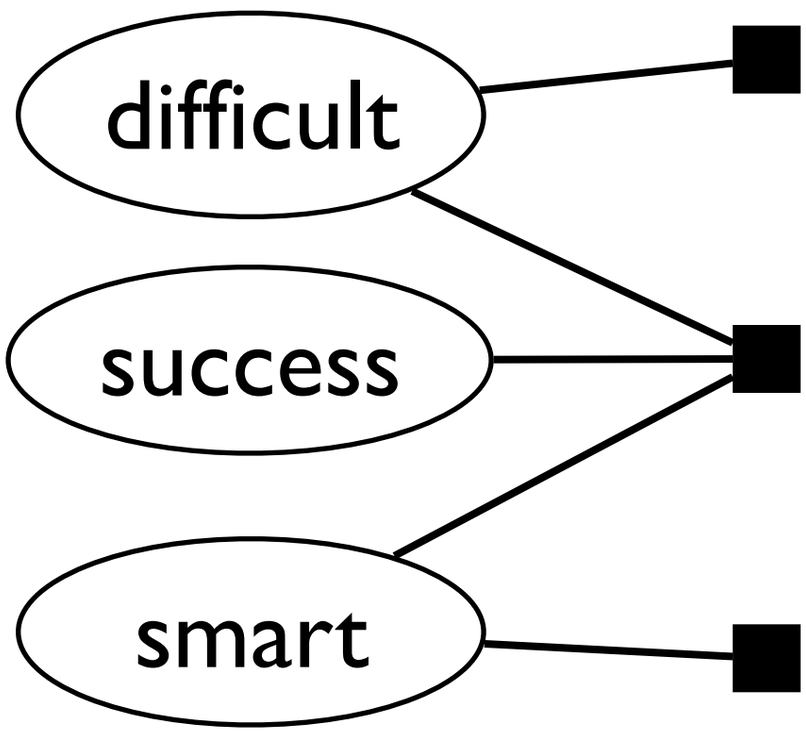
- bi-partite undirected graph
- variables $X_1, \ldots, X_m$
- factors $\phi_1, \ldots, \phi_n$ map interpretations of subsets of variables to non-negative reals
- joint distribution

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{i=1..n} \phi_i(\mathbf{X}_{\phi_i} = \mathbf{x}_{\phi_i})$$

$$Z = \sum_{\mathbf{x}'} \prod_{i=1..n} \phi_i(\mathbf{X}_{\phi_i} = \mathbf{x'}_{\phi_i})$$

BN as Factor Graph?





$$\phi_1(d) = P(d)$$

$$\phi_2(su, d, sm) = P(su|d, sm)$$

$$\phi_3(sm) = P(sm)$$

Z=1

More students, more courses...

gets cumbersome quickly....
relational language to write template?

s(P)

su(P,C)

d(C)

d3

su13

s1

su12

su14

d4

su32

su22

su34

s3

Lots of proposals in the literature, e.g.

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

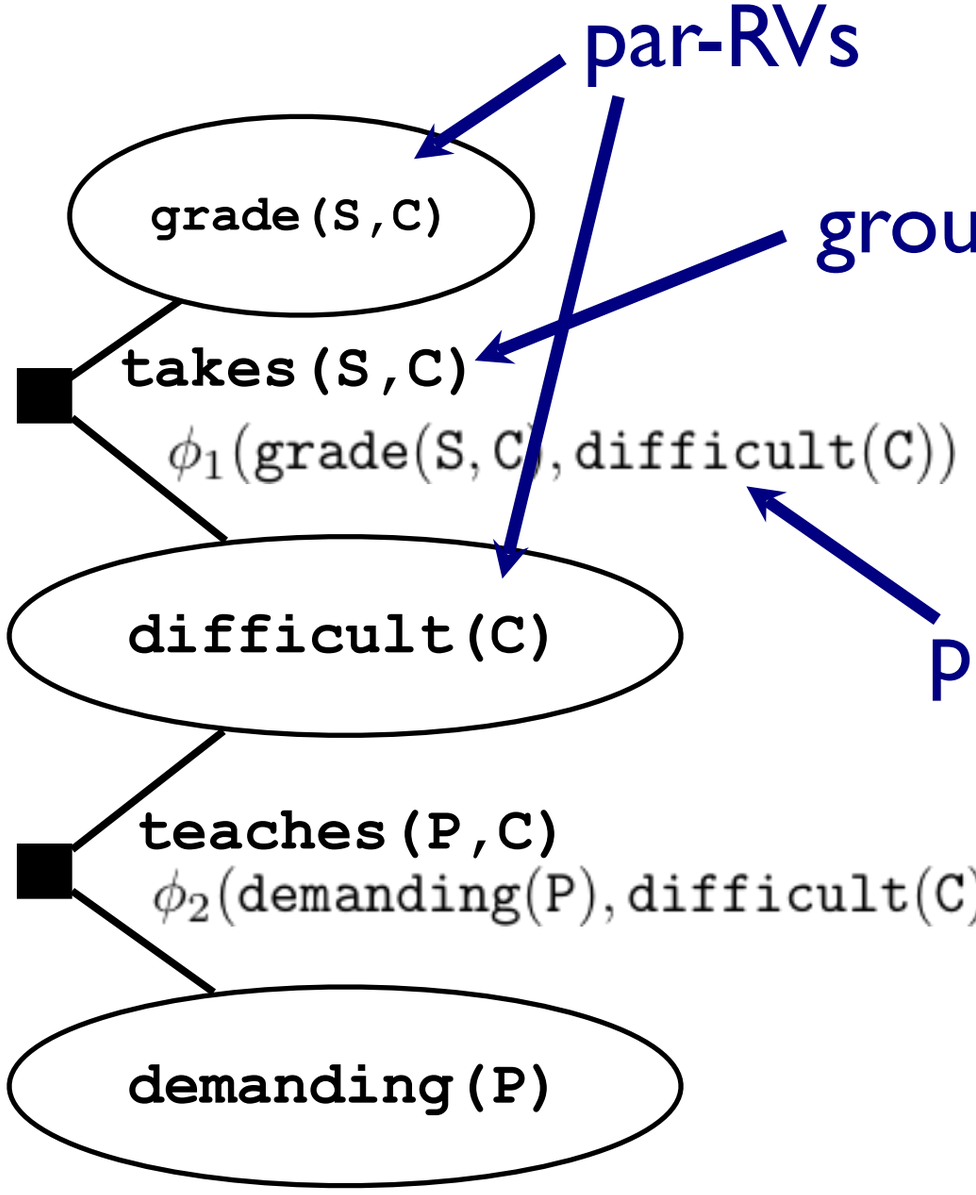
Lots of proposals in the literature, e.g.

- relational Markov networks (RMNs) [Taskar et al 2002]
- Markov logic networks (MLNs) [Richardson & Domingos 2006]
- probabilistic soft logic (PSL) [Broecheler et al 2010]
- FACTORIE [McCallum et al 2009]
- Bay
- rela
- logi
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- probabilistic programming languages such as ProbLog, PRISM, Church, ...
- and many more ...

common principle:
parameterized factor graph
(par-factor graph)

Par-Factor Graph

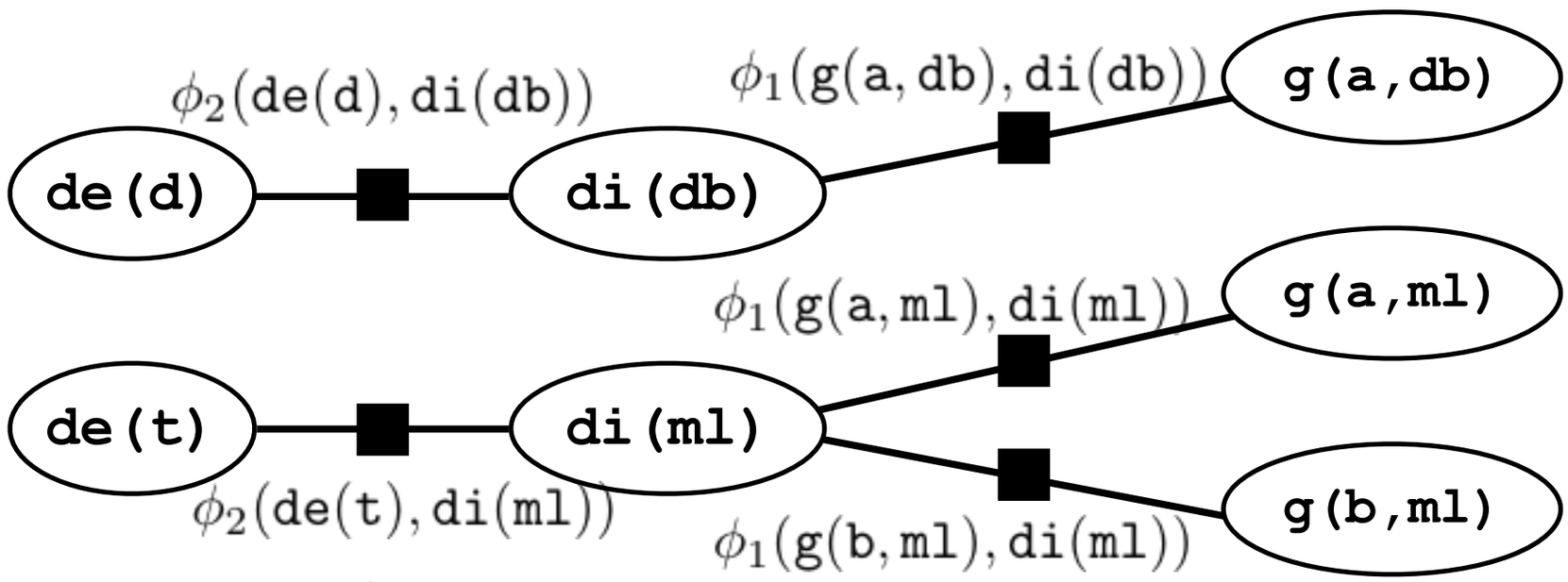
[Poole 03]



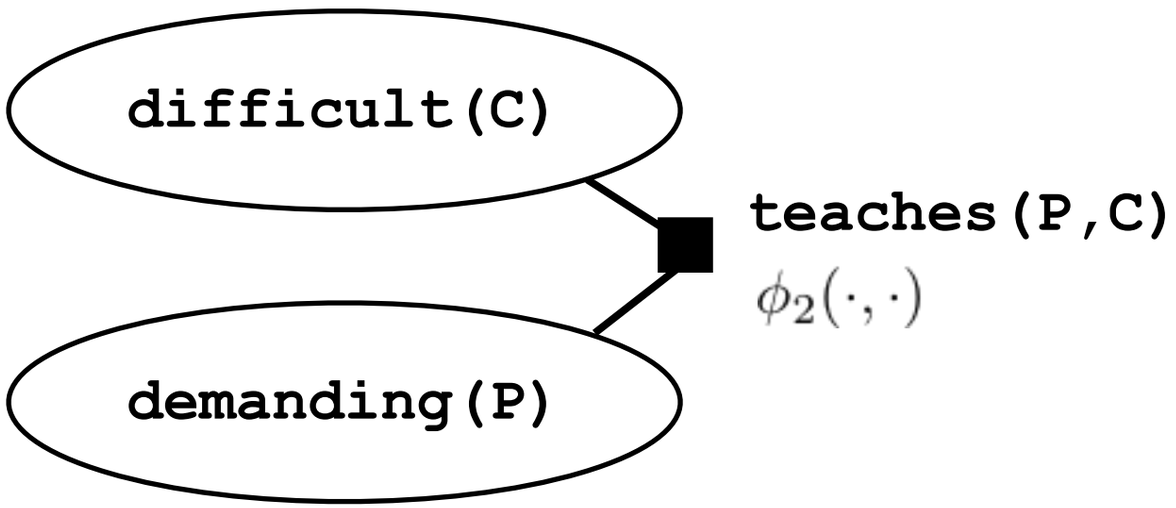
parameterized factor (par-factor)

$$P(.) = \frac{1}{Z} \cdot \prod_{\mathtt{takes(s,c)}} \phi_1(\mathtt{grade(s,c)}, \mathtt{difficult(c)}) \cdot \prod_{\mathtt{teaches(p,c)}} \phi_2(\mathtt{demanding(p)}, \mathtt{difficult(c)})$$

takes(ann,ml).
takes(bob,ml).
takes(ann,db).
teaches(dan,db).
teaches(tom,ml).



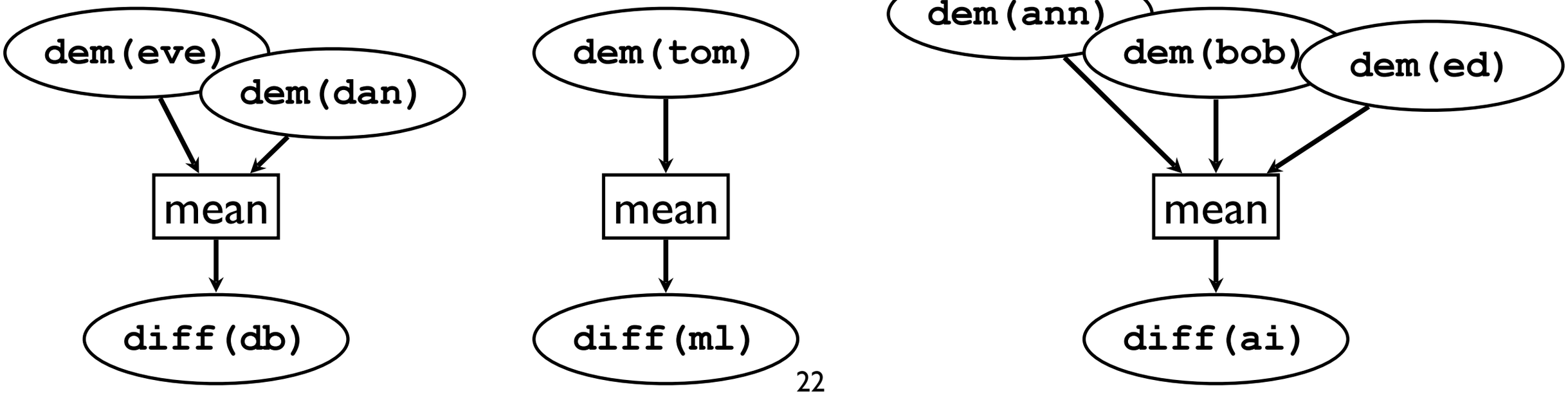
What if multiple professors teach the same course?



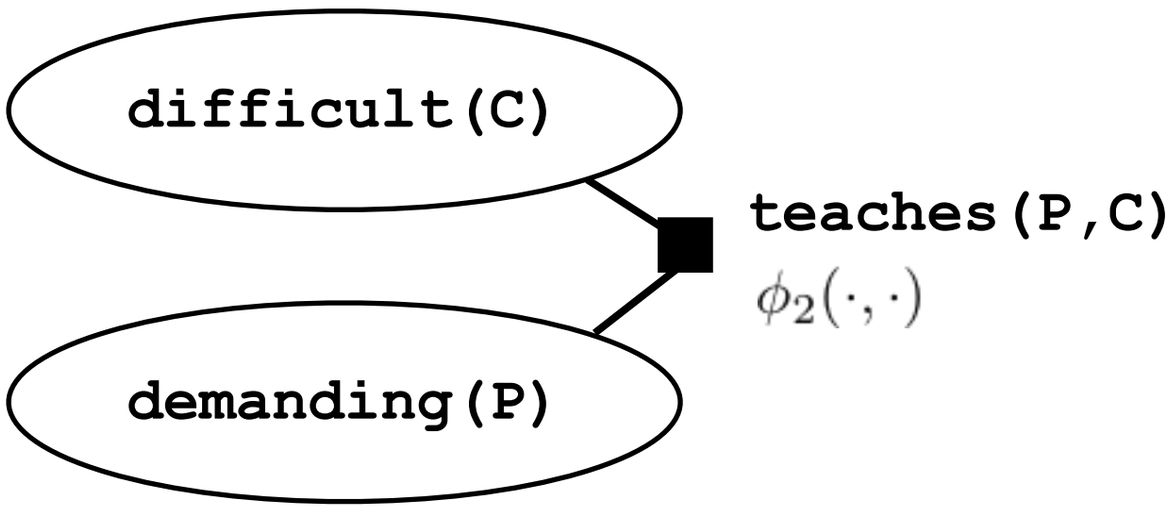

```
teaches(eve,db)
teaches(dan,db).
teaches(tom,ml).
teaches(bob,ai).
teaches(ann,ai).
teaches(ed,ai).
```

Option 1: aggregate values of RVs, then compute potential

$$\phi_2(\texttt{demanding(P)}, \texttt{difficult(C)}) = P(\texttt{difficult(C)}|\text{mean}\{\texttt{demanding(P)}\})$$



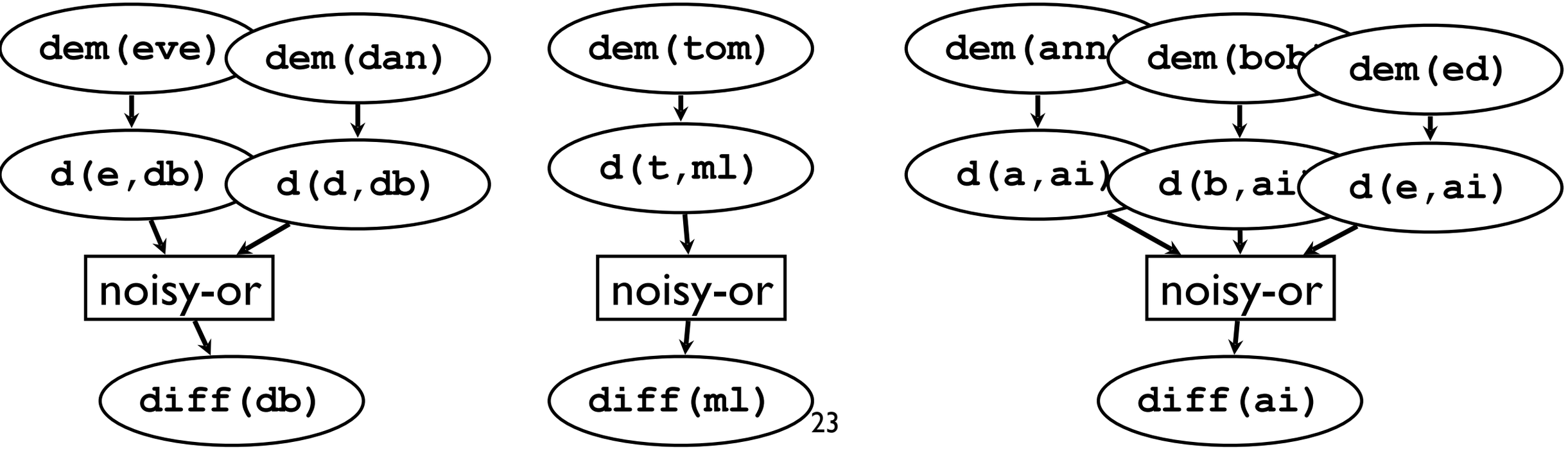
What if multiple professors teach the same course?

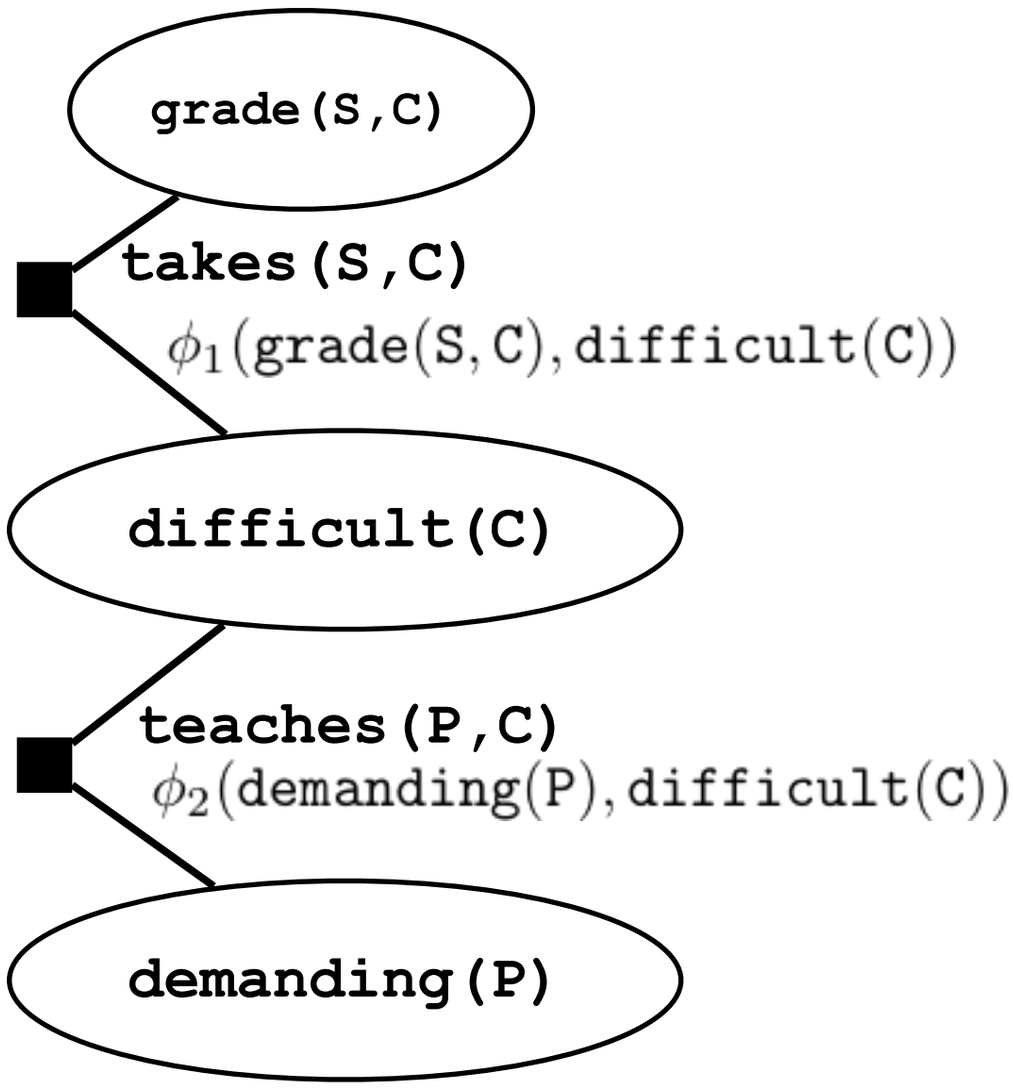



```
teaches(eve,db)
teaches(dan,db).
teaches(tom,ml).
teaches(bob,ai).
teaches(ann,ai).
teaches(ed,ai).
```

Option 2: compute potential for each RV, then combine

$$\phi_2(\texttt{demanding(P)}, \texttt{difficult(C)}) = 1 - \prod_{\texttt{P}} (1 - P(\texttt{difficult(C)} | \texttt{demanding(P)}))$$



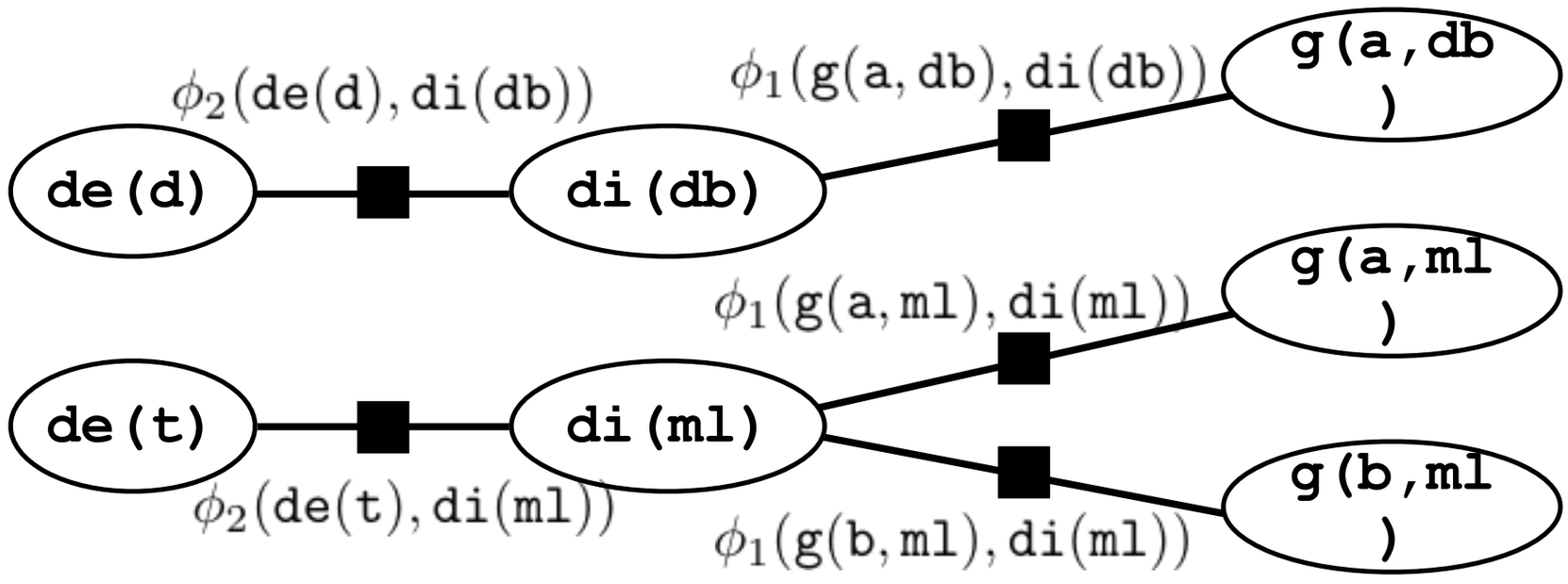



```
takes(ann,ml).
takes(bob,ml).
takes(ann,db).
teaches(dan,db).
teaches(tom,ml).
```

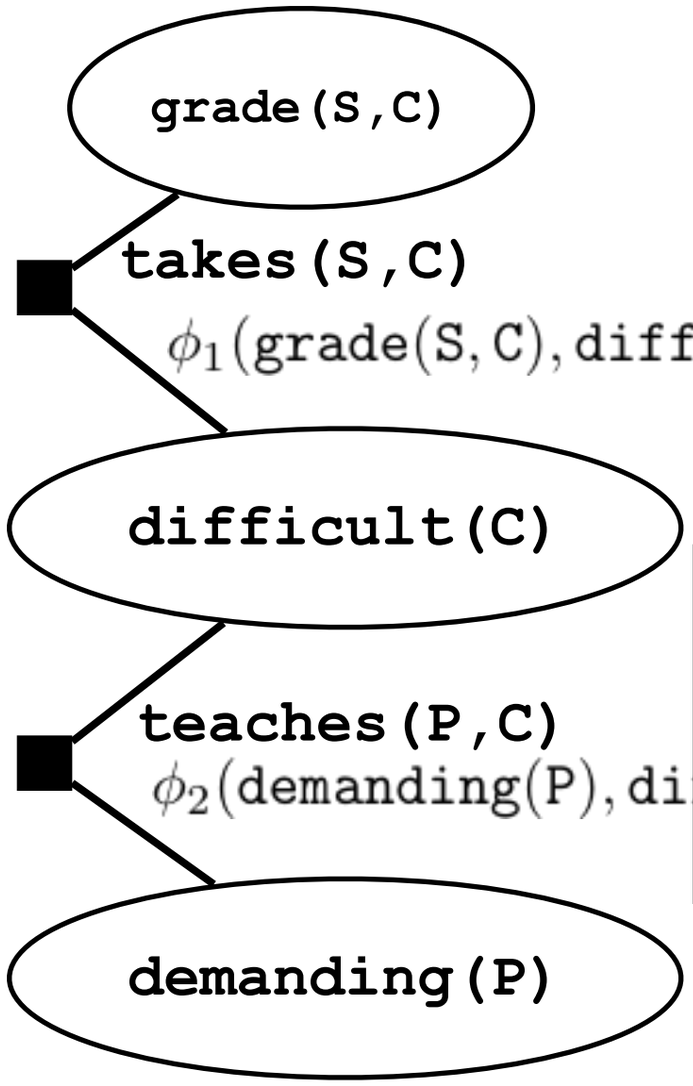
Inference by Grounding

demanding(tom)?

1. construct factor graph
2. run any propositional inference technique



Inference by Grounding



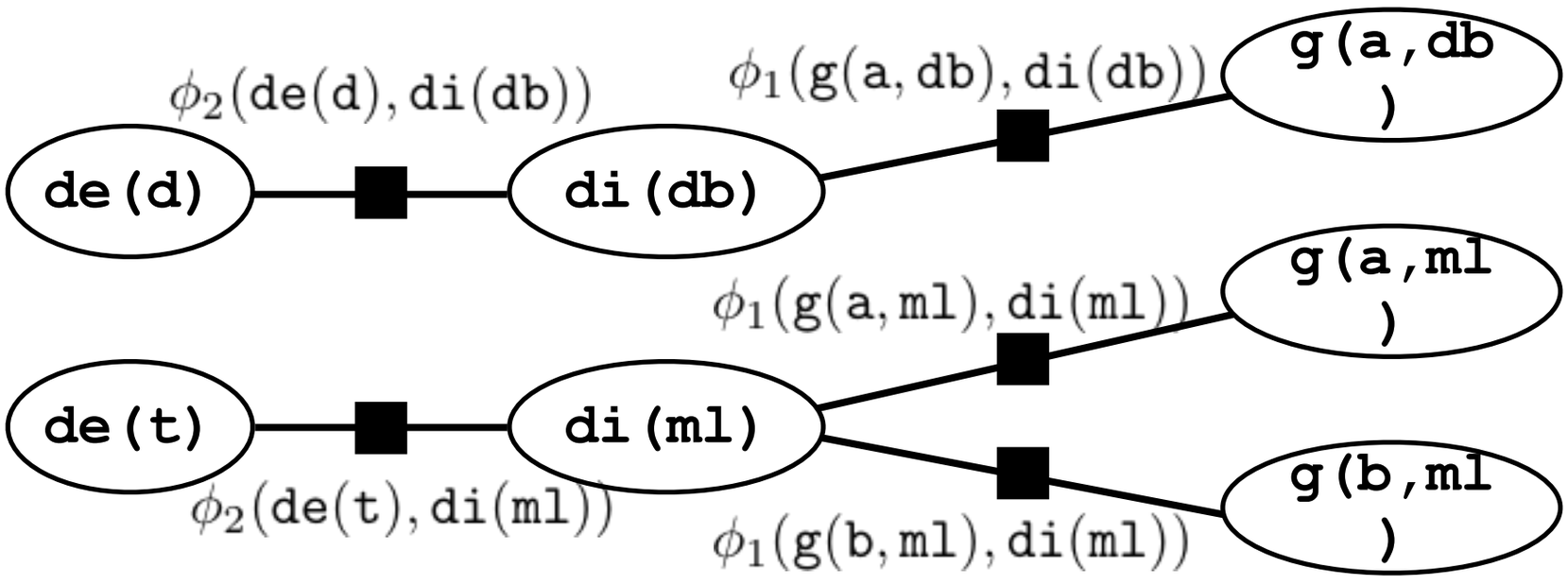
Problem: often huge factor graphs with many repetitions or symmetries (even if restricted to relevant parts)

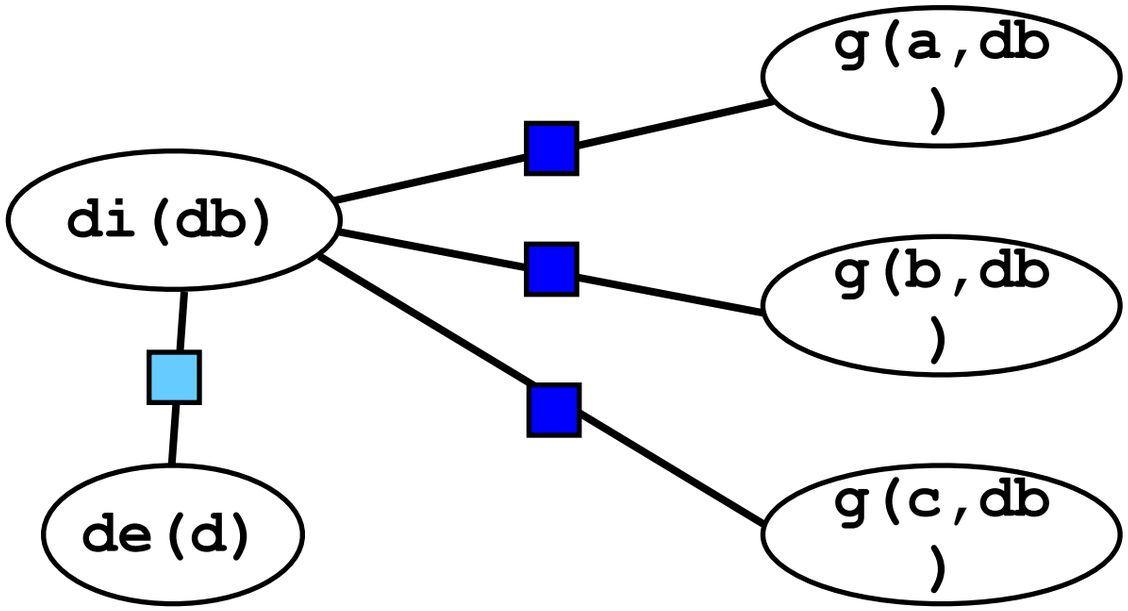
demanding(tom)?

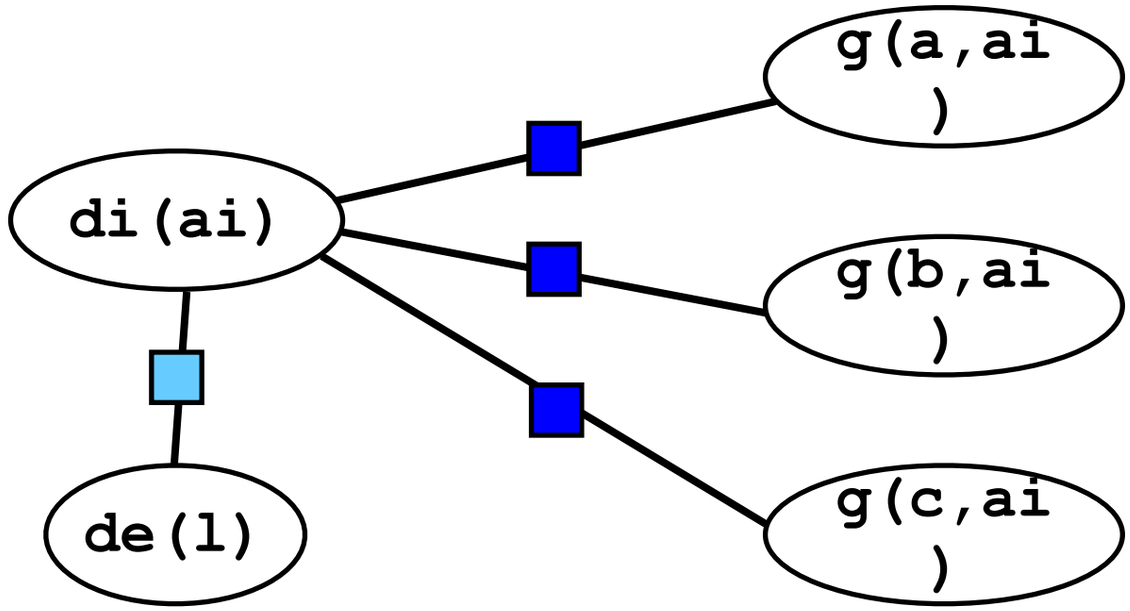
ctor graph

2. Run any propositional inference technique

```
takes(ann,ml).
takes(bob,ml).
takes(ann,db).
teaches(dan,db).
teaches(tom,ml).
```

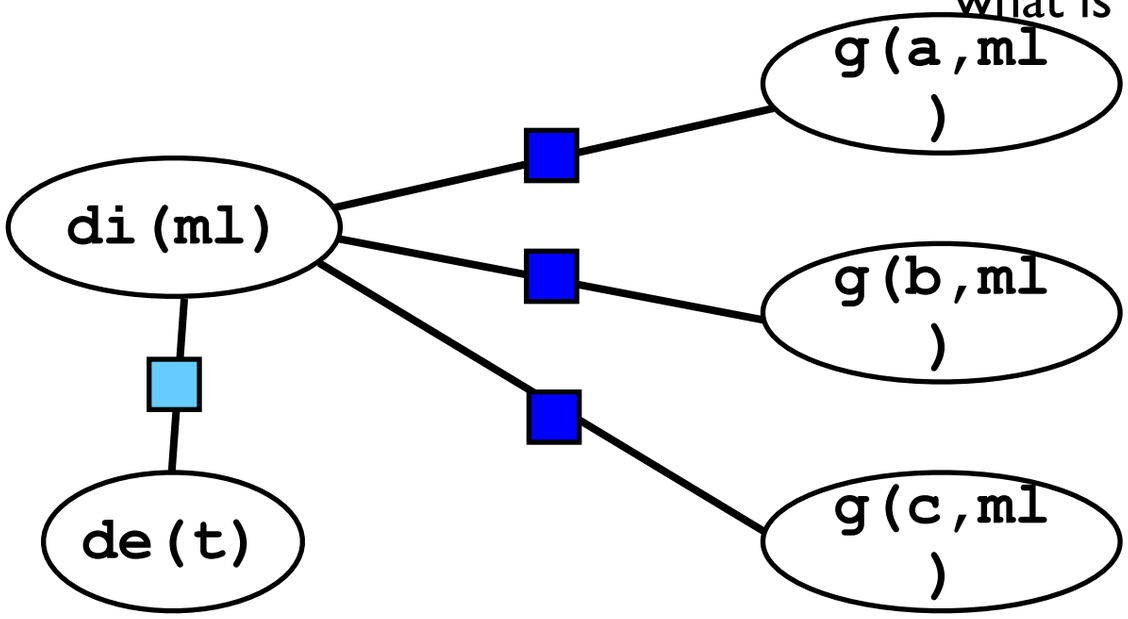


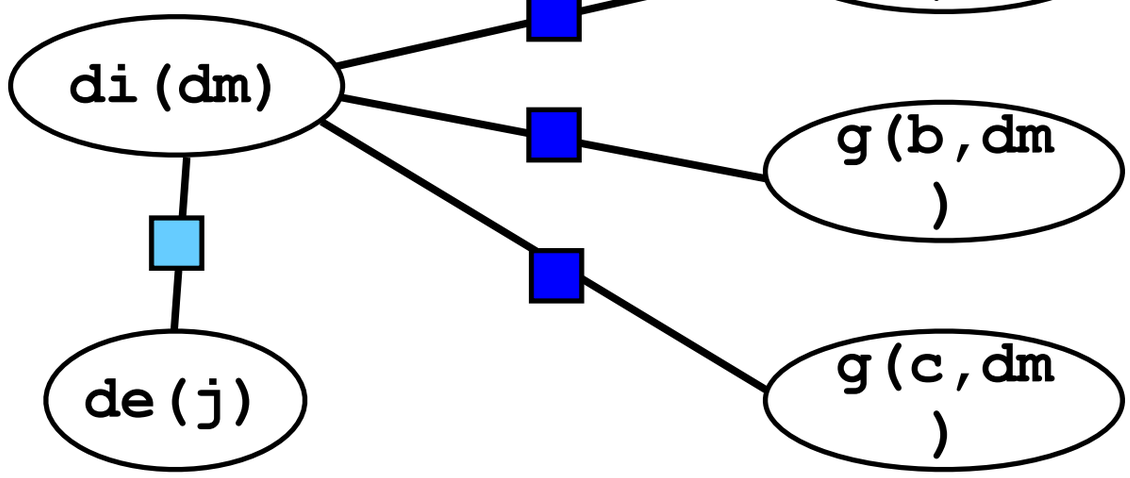


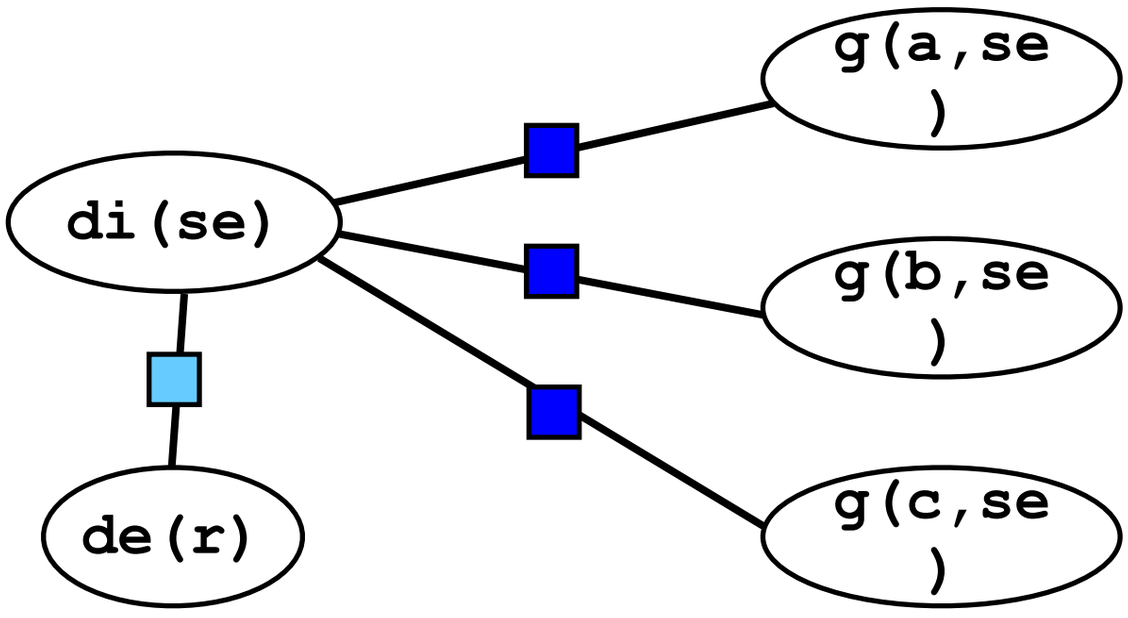


what is the probability that some professor is demanding?

same computation for each professor!





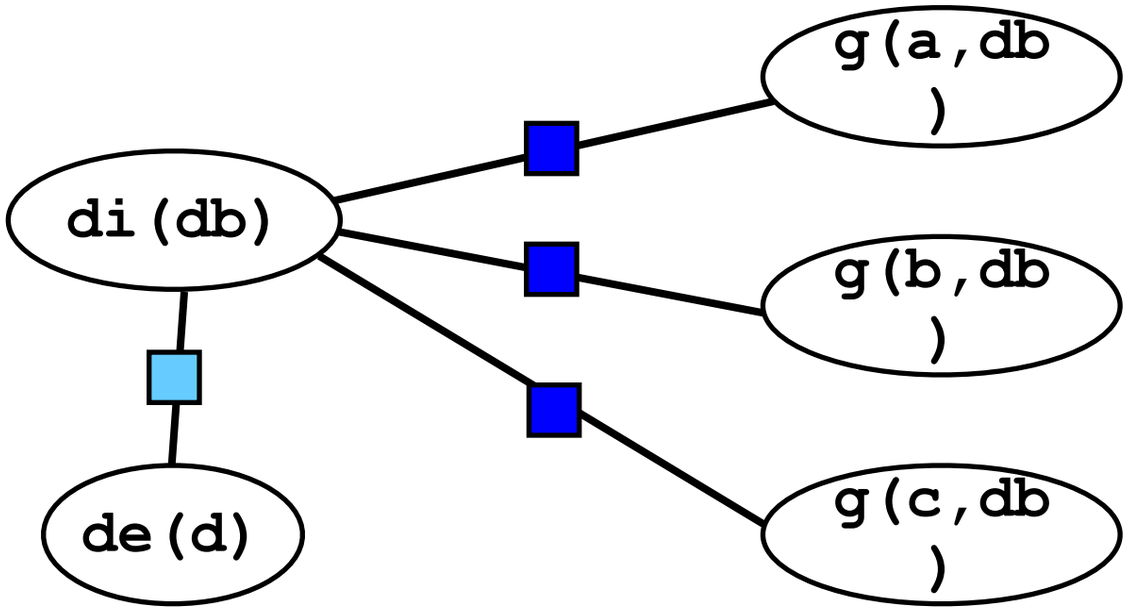


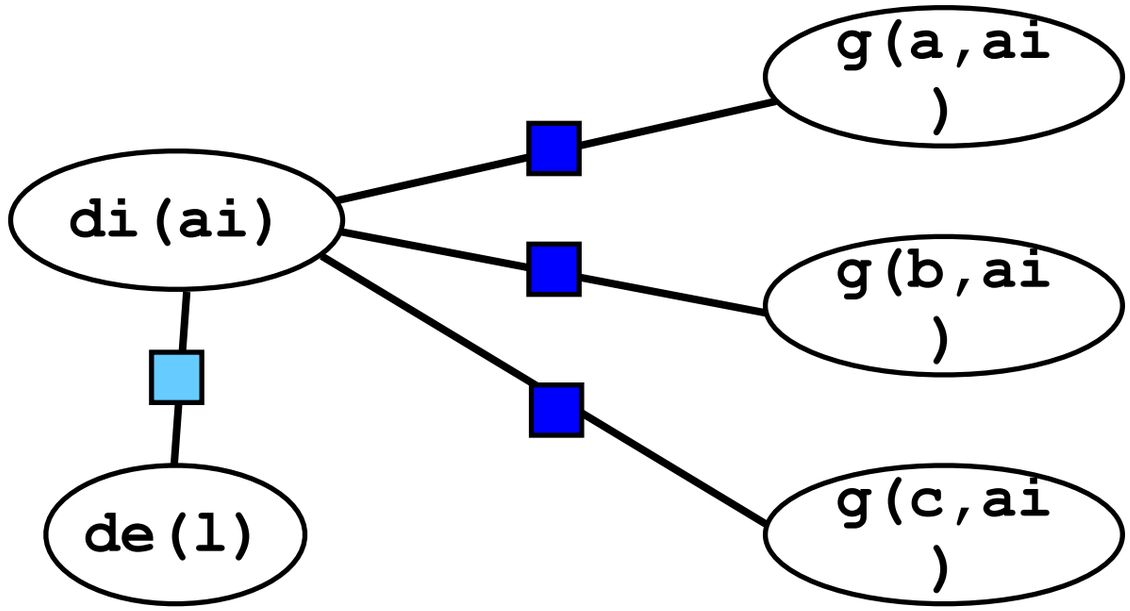
compute probability distribution over grades for every student-course pair

same computation for each pair!

$$1 - \prod_{\mathtt{P}} (1 - P(\mathtt{de(P)})) = 1 - (1 - P(\mathtt{de(someP)}))^{\#\mathtt{P}}$$

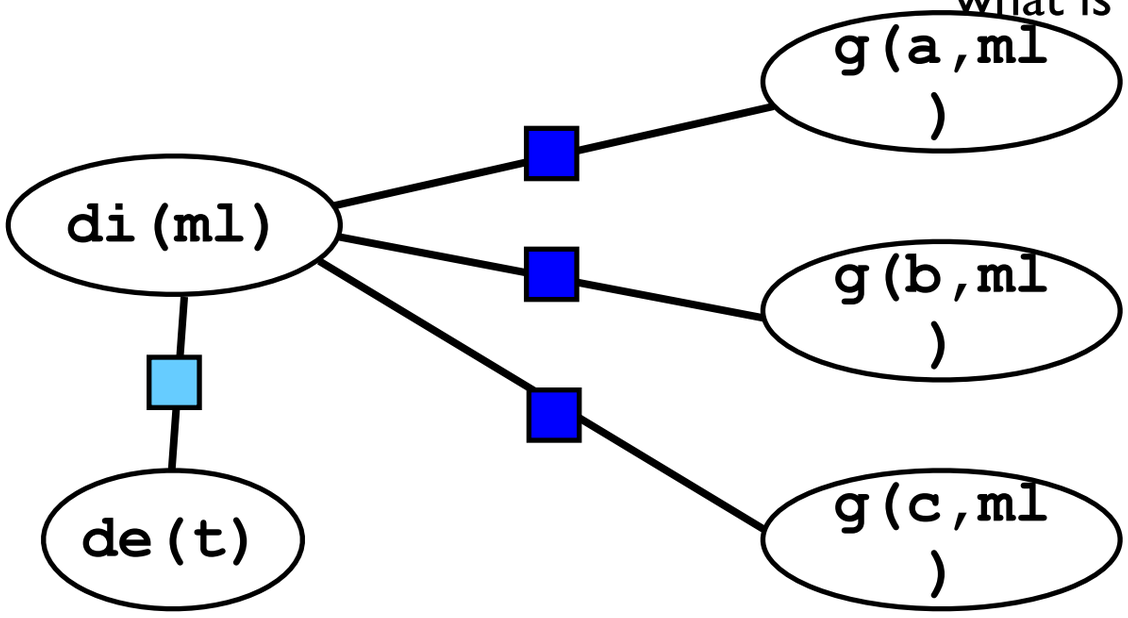
$$= 1 - (1 - P(\mathtt{de(r)}))^5$$

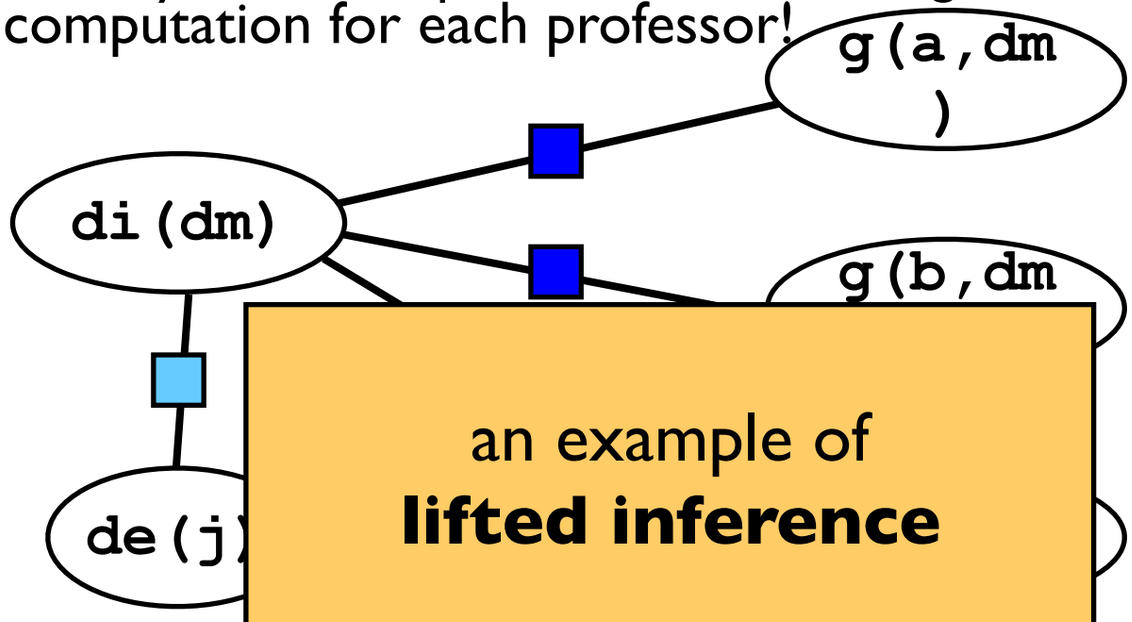




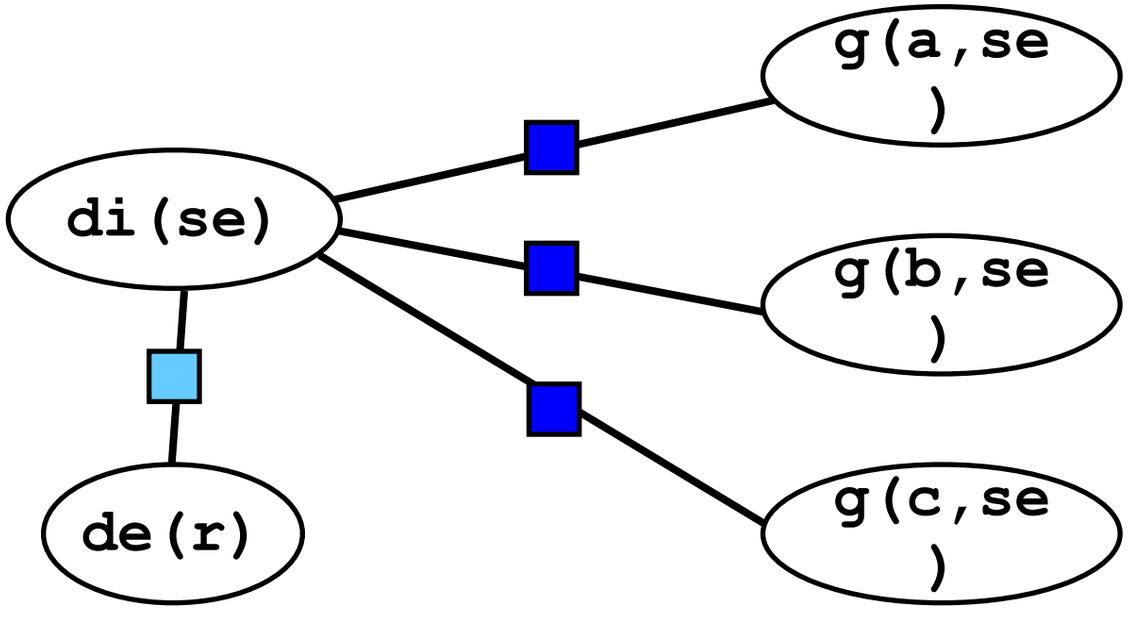
what is the probability that some professor is demanding?

same computation for each professor!





an example of
lifted inference



compute probability distribution over grades for every student-course pair

same computation for each pair!

$$1 - \prod_{\mathtt{P}} (1 - P(\mathtt{de(P)})) = 1 - (1 - P(\mathtt{de(someP)}))^{\#\mathtt{P}}$$

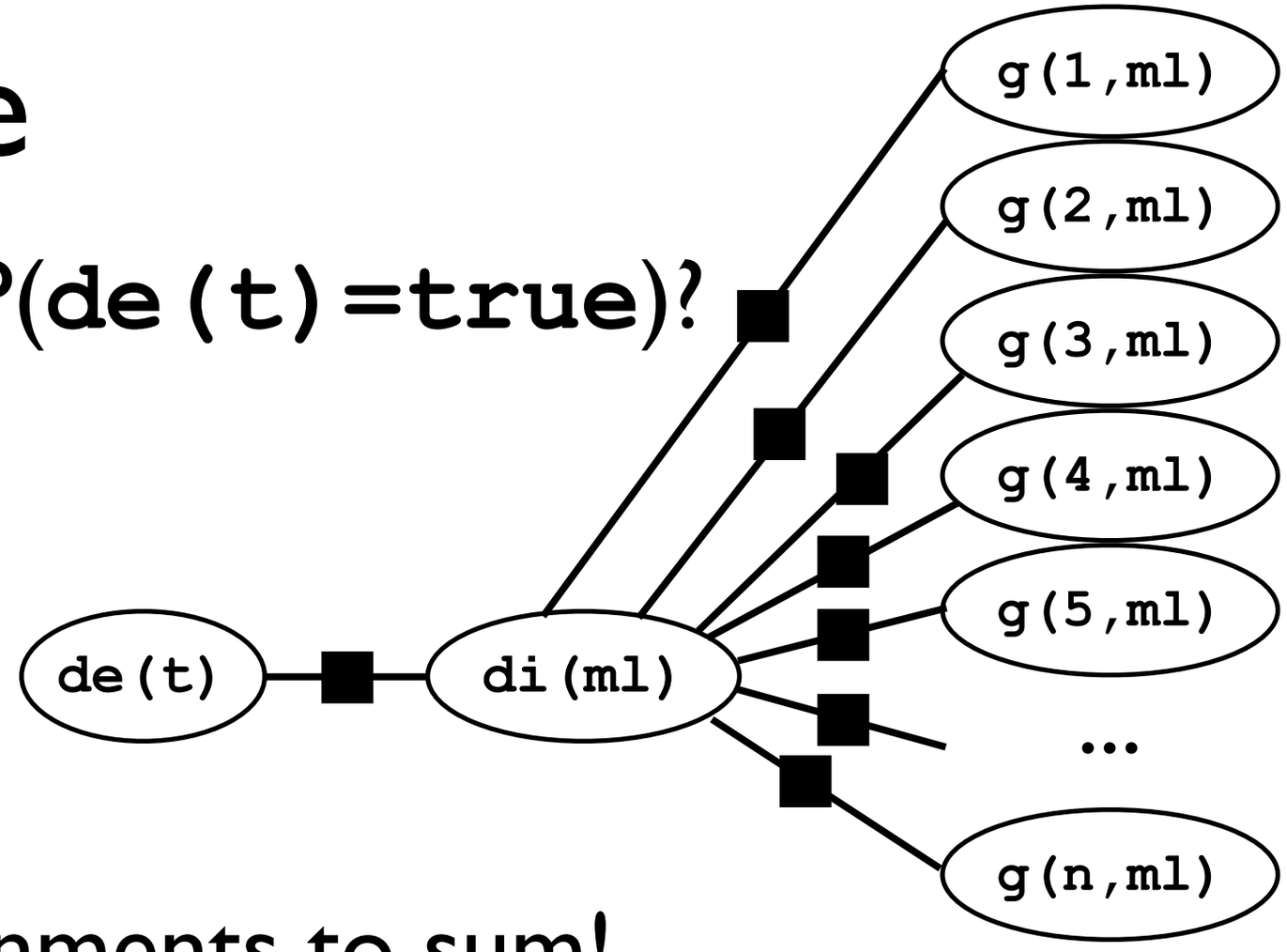
$$= 1 - (1 - P(\mathtt{de(r)}))^{5}$$

Another example

sum $\phi_2(\text{true}, d) \prod_{i=1..n} \phi_1(g_i, d)$

for all combinations of difficulty d and students' grades (g_1,...,g_n)

O(#grades$^{\#\text{students}}$) assignments to sum!



but: identity of students doesn't matter!
sufficient to count how often each grade m_1,...,m_k appears

sum $\phi_2(\text{true}, d) \prod_{i=1..k} \phi_1(m_i, d)^{\#m_i}$ instead

e.g., k=3, n=15: >14M grade vectors vs 136 count vectors

Lifted Inference

- Exploiting symmetries & repeated structure
- Reasoning on first order level as much as possible
- Aiming at independence from number of objects
- Approximation: grouping similar computations
- Very active research area

Lifted Weighted Model Counting

KU LEUVEN

STATISTICAL RELATIONAL MODELS

Statistical

Cloud	Rain	Pr(Rain|Cloudy)
T	T	0.8
T	F	0.2
F	T	0.01
F	F	0.99

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

Logical

$\text{Rain} \Rightarrow \text{Cloudy}$

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

Propositional

Relational

WHY WEIGHTED MODEL COUNTING?

Model Counting MC	Count the satisfying assignments of a propositional sentence
Weighted Model Counting WMC	Associate a weight with each assignment and compute sum of weights
Weighted First-Order Model Counting WFOMC	Exploit symmetries in a first-order sentence

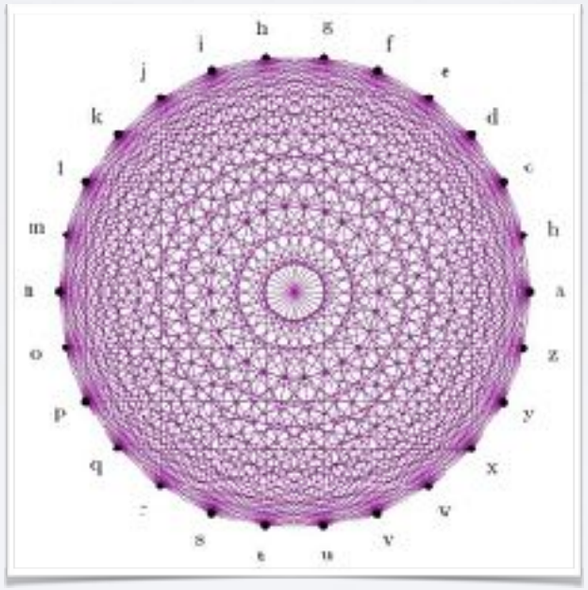
Applications in: Probabilistic graphical models
Statistical relational models
Probabilistic logic models

For example: Computing probabilities using lifted inference
Learning the weights from data

WFOMC FOR MLNS

3.14 FacultyPage(x) ∧ Linked(x, y) ⇒ CoursePage(y).

As a probabilistic graphical model:
26 pages, 728 random variables, 676 factors



Real data set:
1000 pages, 1002000 random variables, 1000000 factors
→ Highly intractable probability computation?
Lifted inference (WFOMC) in milliseconds!

WEIGHTED MODEL COUNTING

A logical theory and a weight function

Stress ⇒ Smokes.

literal → *weight*

Stress → 4

¬Stress → 1

Smokes → 1

¬Smokes → 2

Stress	Smokes		Δ	WM
4	1			4
4	2			8
1	1			1
1	2			2
		∑	3 models	7

MC = 3

WMC = 7

WEIGHTED FIRST-ORDER MODEL COUNTING

$$WMC(\Delta, \mathrm{w}) = \sum_{\omega \models \Delta} \prod_{l \in \omega} \mathrm{w}(l)$$

$$\mathrm{WFOMC}(\Delta, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \sum_{\omega \models_{\mathbf{D}} \Delta} \prod_{l \in \omega_0} \bar{\mathrm{w}}(\mathrm{pred}(l)) \prod_{l \in \omega_1} \mathrm{w}(\mathrm{pred}(l))$$

False literals

True literals

[Van den Broeck et al., IJCAI 2011]

WEIGHTED FIRST-ORDER MODEL COUNTING

A first-order logical theory and two weight functions

Stress(Alice) ⇒ Smokes(Alice).

pred → *weight*

$\mathbf{w}$

Stress → 4

Smokes → 1

$\overline{\mathbf{w}}$

Stress → 1

Smokes → 2

Stress	Smokes		Δ	WM
4	1			4
4	2			8
1	1			1
1	2			2
		Σ	3 models	7

MC = 3

WMC = 7

WEIGHTED FIRST-ORDER MODEL COUNTING

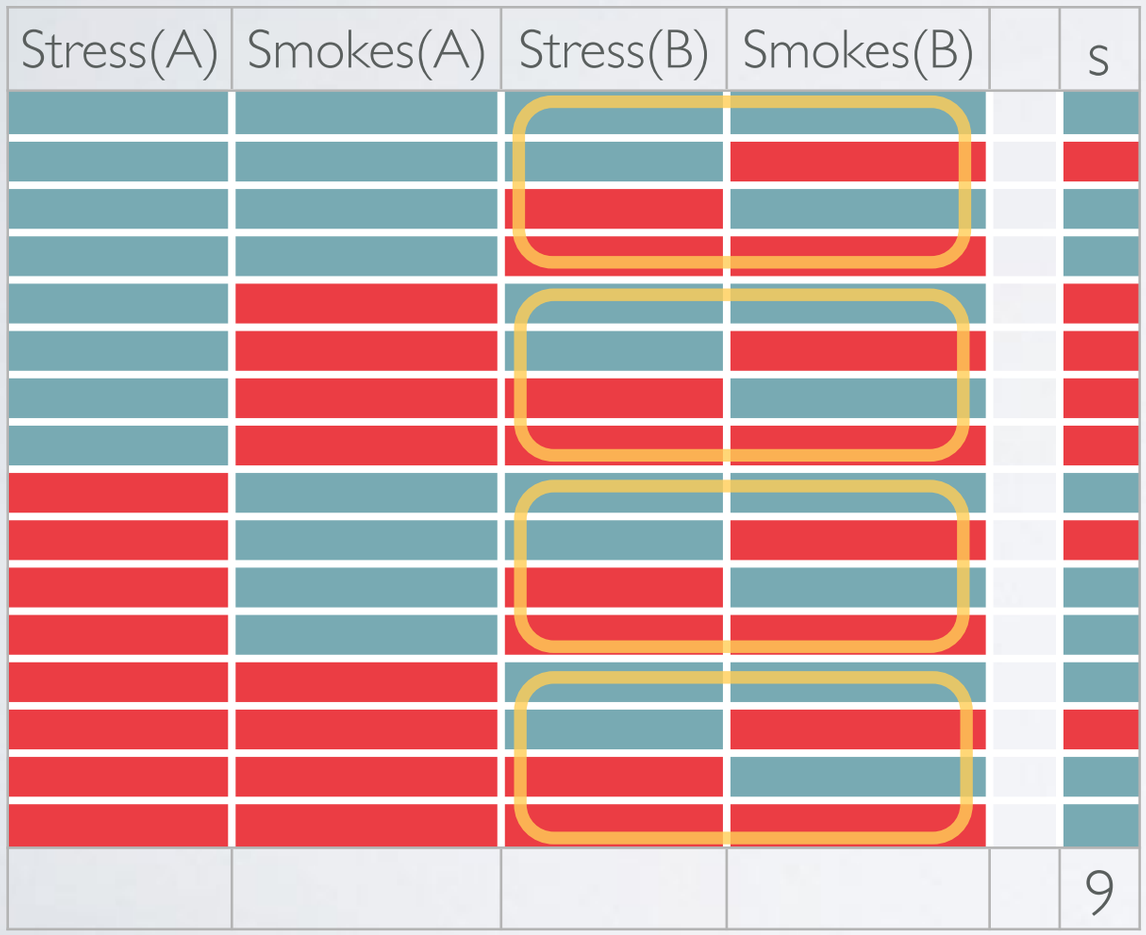
Stress(A) ⇒ Smokes(A).

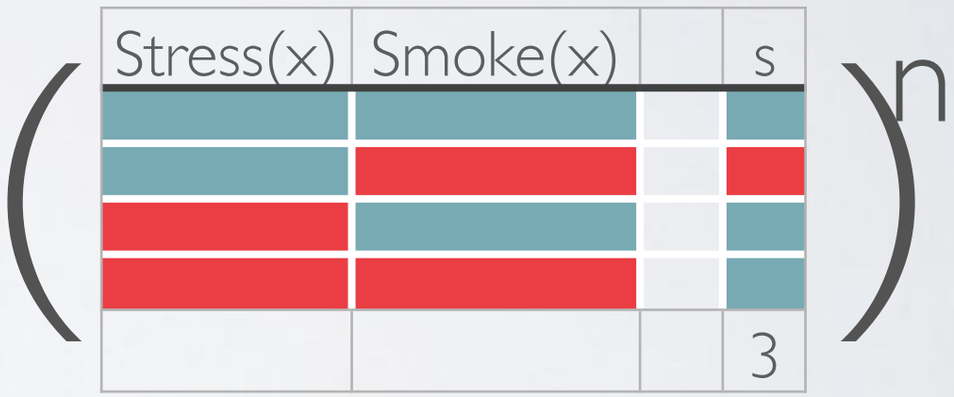
Stress(B) ⇒ Smokes(B).

$w(\cdot)=1$
$\overline{w}(\cdot)=1$

∀x, Stress(x) ⇒ Smokes(x).

D = {A,B}
|**D**| = n = 2





Expression independent of domain size

WMC = 9

WFOMC = $3^n = 9$

WFOMC

Domain: n people

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

if Female:	$\forall y,\ \text{ParentOf}(y) \Rightarrow \text{MotherOf}(y).$	$\text{WFOMC} = 3^n$
if ¬Female:	$\forall y,\ \text{true}.$	$\text{WFOMC} = 4^n$

$$\text{WFOMC} = 3^n + 4^n$$

WFOMC

Domain: n people

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

if x known: $\forall y, \text{ParentOf}(A,y) \wedge \text{Female}(A) \Rightarrow \text{MotherOf}(A,y).$

$$\text{WFOMC} = 3^n + 4^n$$

$$\text{WFOMC} = (3^n + 4^n)^n$$

WFOMC

Domain: n people

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

We know the k smokers:

4 disjunct groups

If Smokes(A) = true ∧ Smokes(B) = true

true ∧ Friends(A,B) ⇒ true.

If Smokes(A) = true ∧ Smokes(C) = false

true ∧ Friends(A,C) ⇒ false. → k(n-k)

If Smokes(D) = false ∧ Smokes(B) = true

false ∧ Friends(D,B) ⇒ true.

If Smokes(D) = false ∧ Smokes(C) = false

false ∧ Friends(D,C) ⇒ false.

$2^{n^2-k(n-k)}$ models

We know there are k smokers:

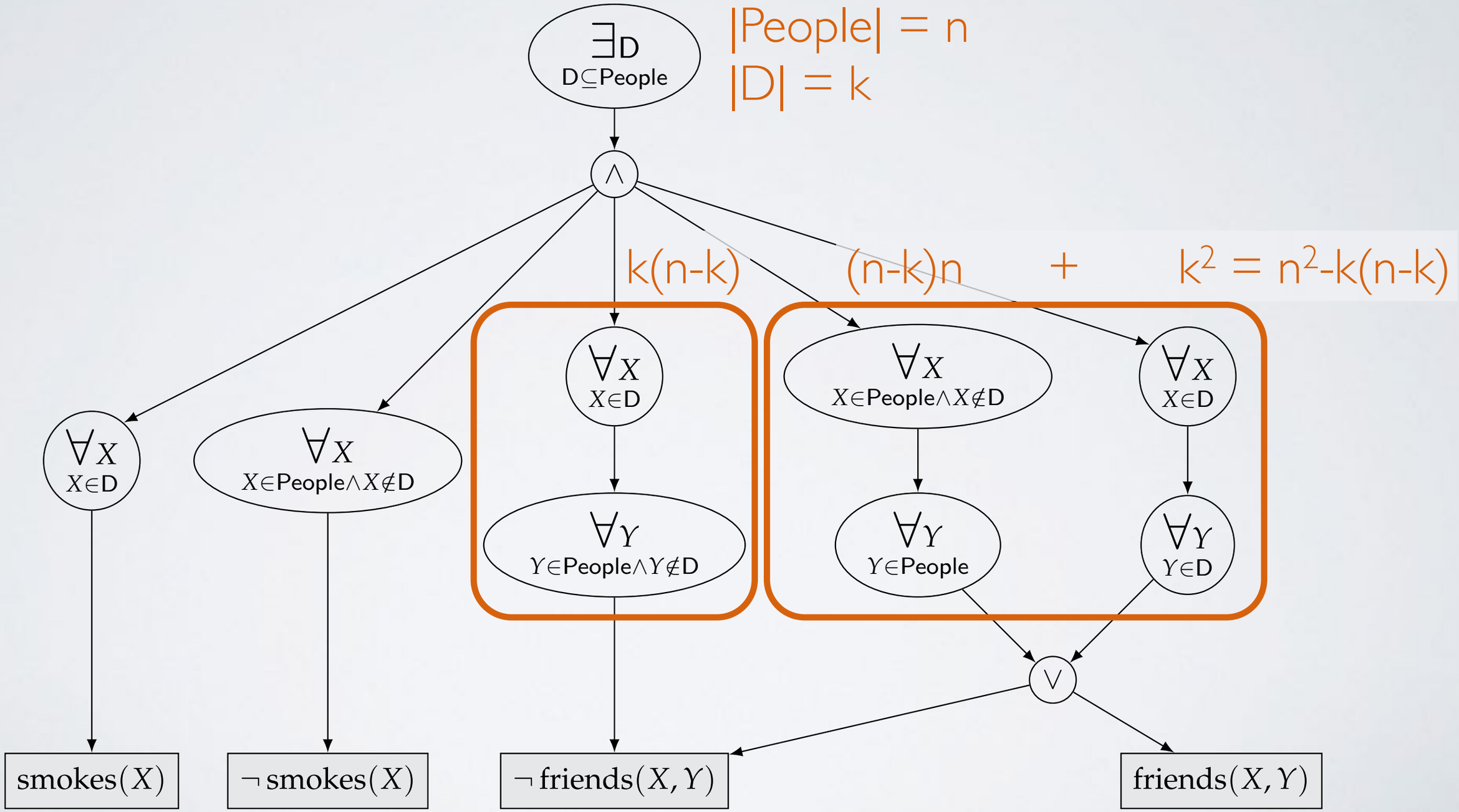
$\binom{n}{k} 2^{n^2-k(n-k)}$ models

WFOMC:

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

WFOMC WITH FO-D-DNNF

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



PROBABILISTIC LOGIC PROGRAM

0.1 :: Attends(x).
0.3::Series :- Attends(x).

~Noisy-OR

$\text{Series} \Leftrightarrow \exists x, \text{Attends}(x) \wedge \text{ToSeries}(x).$

$w(\text{Attends}(\cdot)) = 0.1 \quad w(\text{ToSeries}(\cdot)) = 0.3$
$w(\neg\text{Attends}(\cdot)) = 0.9 \quad w(\neg\text{ToSeries}(\cdot)) = 0.7$
$w(\cdot) = 1$

Weighted (First-Order) Model Counting

$\Pr(\text{Series} \mid \text{Attends}(A))$

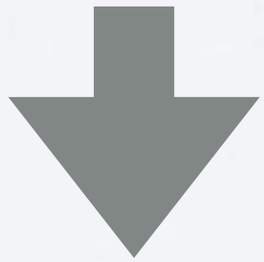
PROBABILISTIC LOGIC PROGRAM

$$\text{Series} \Leftrightarrow \exists x, \text{Attends}(x) \wedge \text{ToSeries}(x).$$

Domain = {A,B}

$$\text{Series} \Leftrightarrow [(\text{Attends}(A) \wedge \text{ToSeries}(A)) \vee (\text{Attends}(B) \wedge \text{ToSeries}(B))].$$

Lost first-order structure

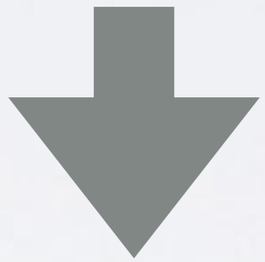


CNF, d-DNNF, AC, …
Weighted Model Counting

Pr(Series | Attends(A))

PROBABILISTIC LOGIC PROGRAM

0.1 :: Attends(x).
0.3::Series :- Attends(x).



Series $\Leftrightarrow \exists x$, Attends(x) $\wedge$ ToSeries(x).

$\mathbf{w}$(Attends) = 0.1 $\quad$ $\mathbf{w}$(ToSeries) = 0.3
$\overline{\mathbf{w}}$(Attends) = 0.9 $\quad$ $\overline{\mathbf{w}}$(ToSeries) = 0.7
$\mathbf{w}(\cdot)$ = 1 $\quad$ $\overline{\mathbf{w}}(\cdot)$ = 1

PROBABILISTIC LOGIC PROGRAM

Series ⇔ ∃x, Attends(x) ∧ ToSeries(x).

Series ⇔ Z

∀x, Z ∨ ¬[Attends(x) ∧ ToSeries(x)]

Z ∨ S

∀x, S ∨ ¬[Attends(x) ∧ ToSeries(x)]

$\mathbf{w}$(Attends) = 0.1 $\mathbf{w}$(ToSeries) = 0.3

$\bar{\mathbf{w}}$(Attends) = 0.9 $\bar{\mathbf{w}}$(ToSeries) = 0.7

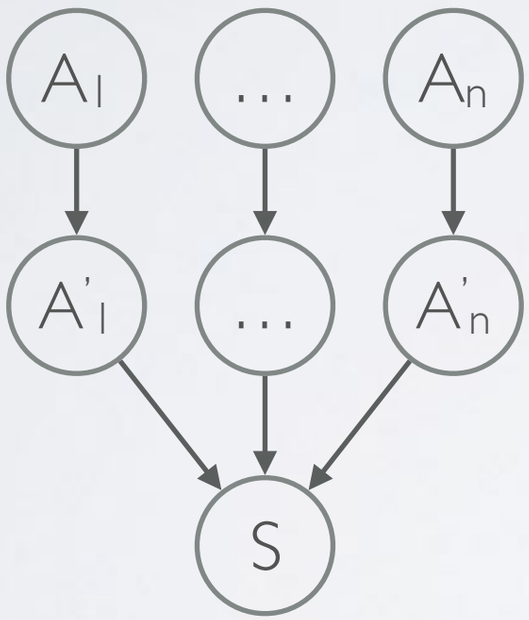
$\bar{\mathbf{w}}$(S) = -1

$\mathbf{w}(\cdot) = 1$ $\bar{\mathbf{w}}(\cdot) = 1$

NOISY-OR

Multiplicative factorisation (or tensor rank-one decomposition) for noisy-OR is a special case of the Skolemization procedure

```
0.1 :: Attends(x).
0.3::Series :- Attends(x).
```



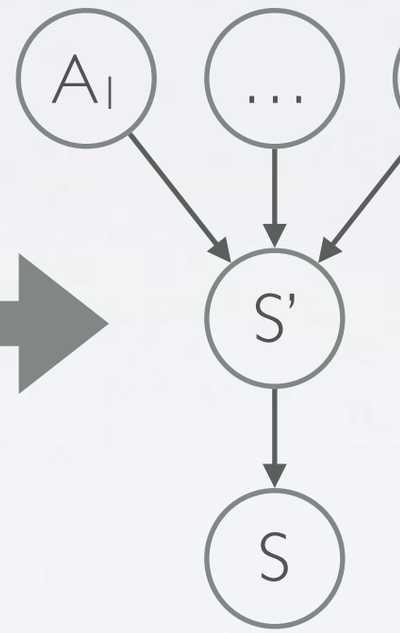
	A_i	
A'_i	f	t
f	1	1-0.3
t	0	0.3

$$S = \top \Leftrightarrow \exists A'_i, A'_i = \top$$

S is false if all A_i are inhibited

→ only 1 model

Relax for this model and then compensate



	A_i	
S'	f	t
f	1	1-0.3
t	1	1

	S'	
S	f	t
f	1	0
t	-1	1

[Diez et al, 2003]
[Takikawa et al, 1999]

C-FOVE operator

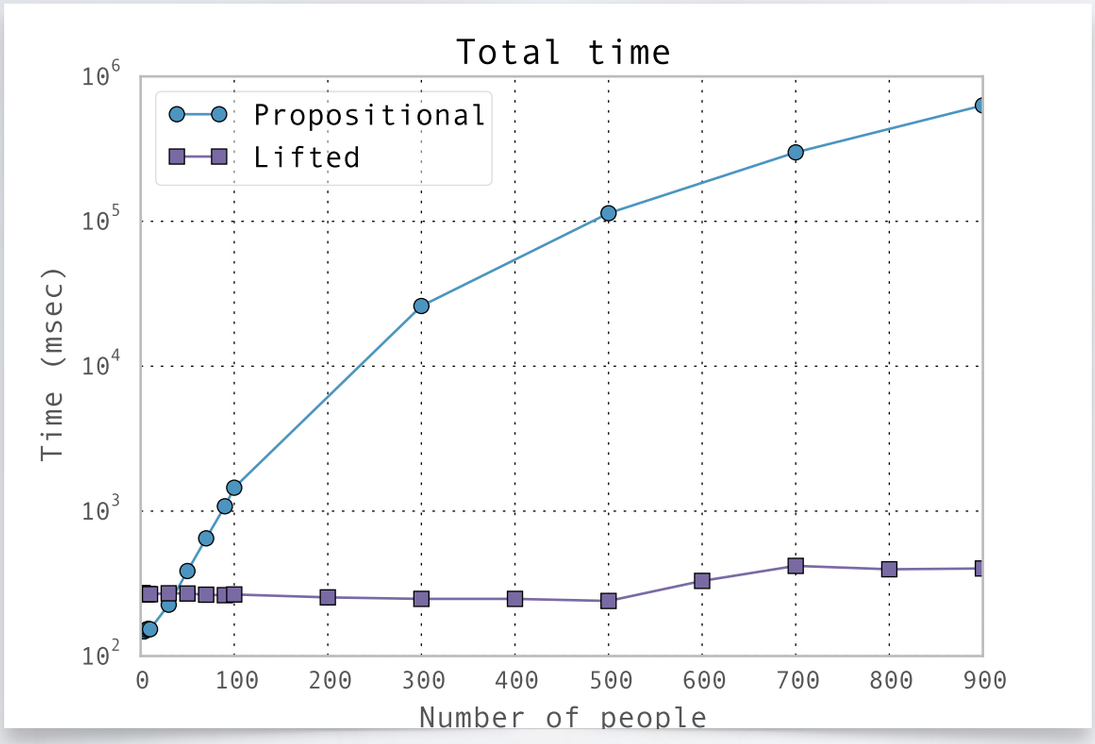
[Kisynski et al, IJCAI 2011]

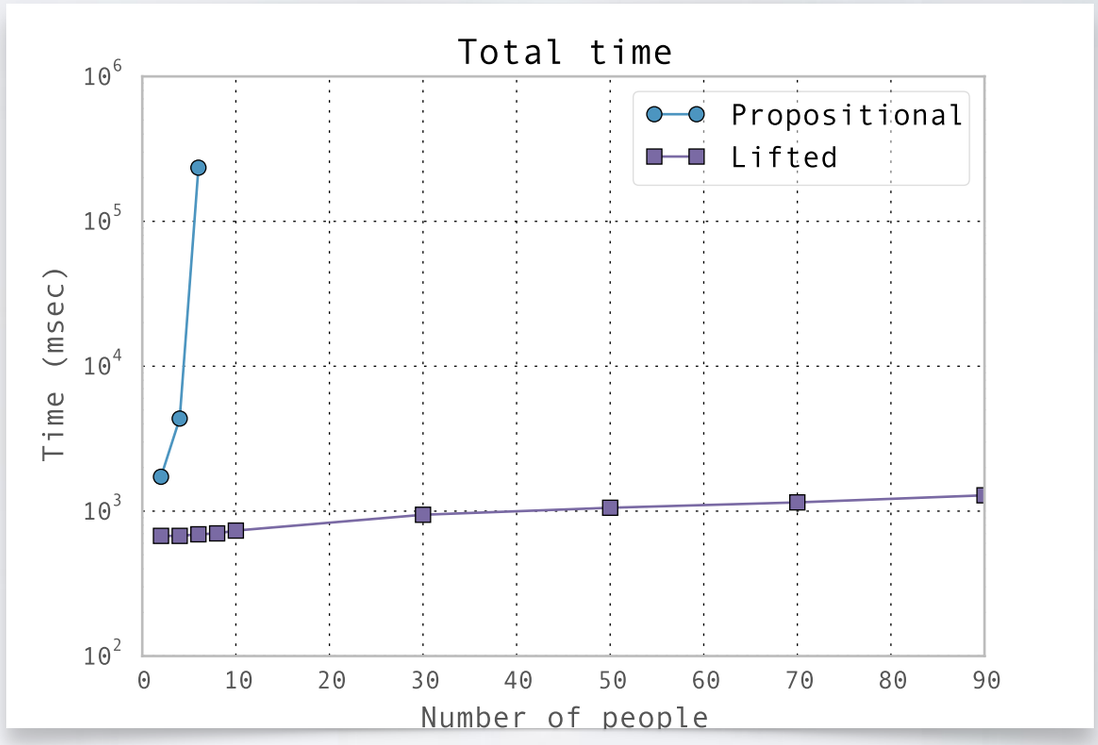
GC-FOVE operator

[Bellodi et al, ICLP 2014]

EXPERIMENTS

0.1 :: Attends(x).
0.3 :: Series :- Attends(x).





0.1 :: Attends(x).
0.3 :: Series :- Attends(x), Coauthor(x, y), Attends(y).

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

https://github.com/UCLA-StarAI/Forclift

DTAI

FWO VLAANDEREN

UCLA ENGINEERING
Computer Science

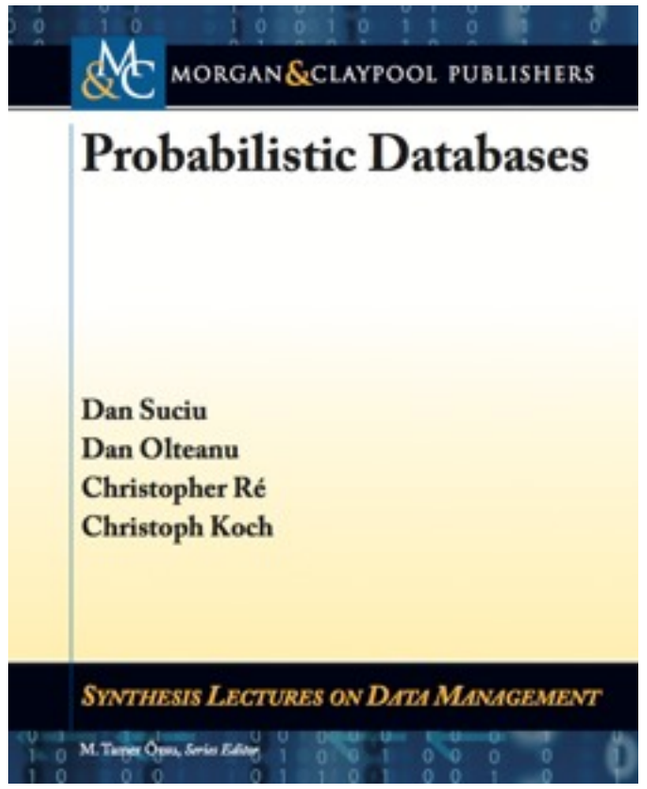
KU LEUVEN

Probabilistic Databases

KU LEUVEN

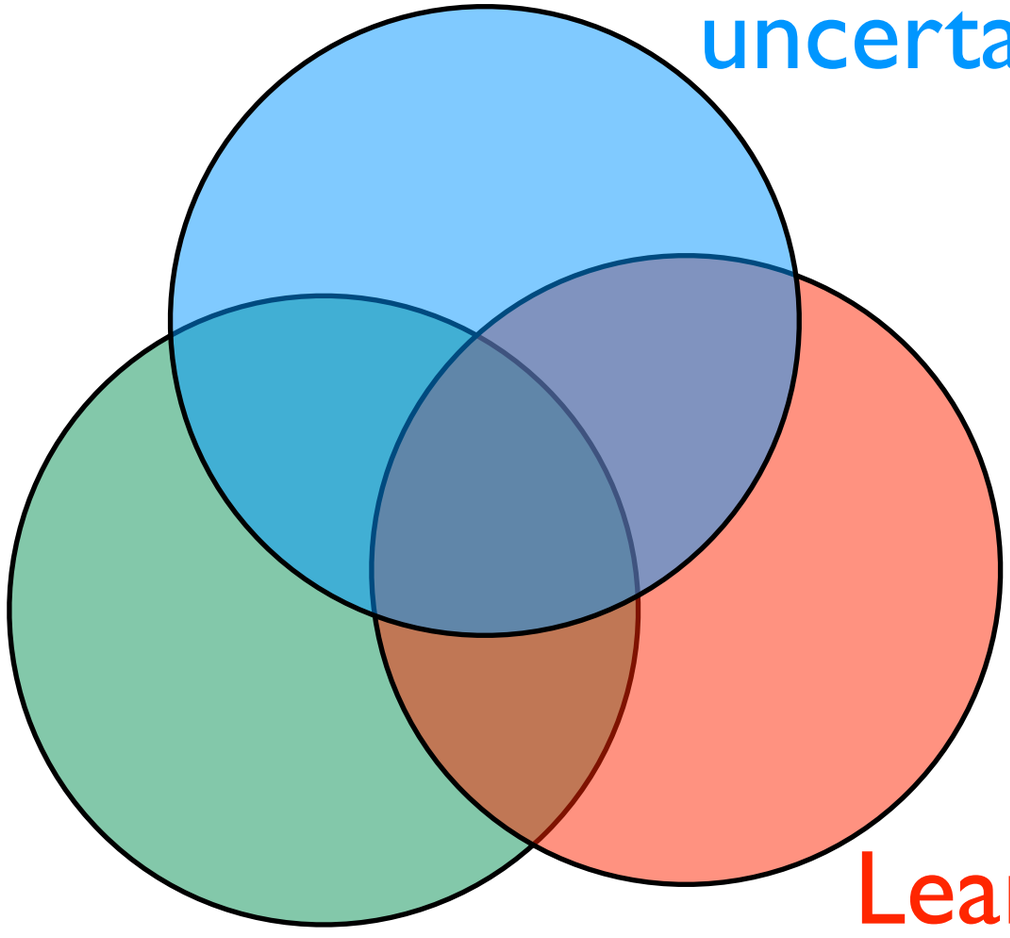
Probabilistic Databases

partly based on



Probabilistic Databases

Dealing with uncertainty



Reasoning with relational data

Learning

Probabilistic Databases

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

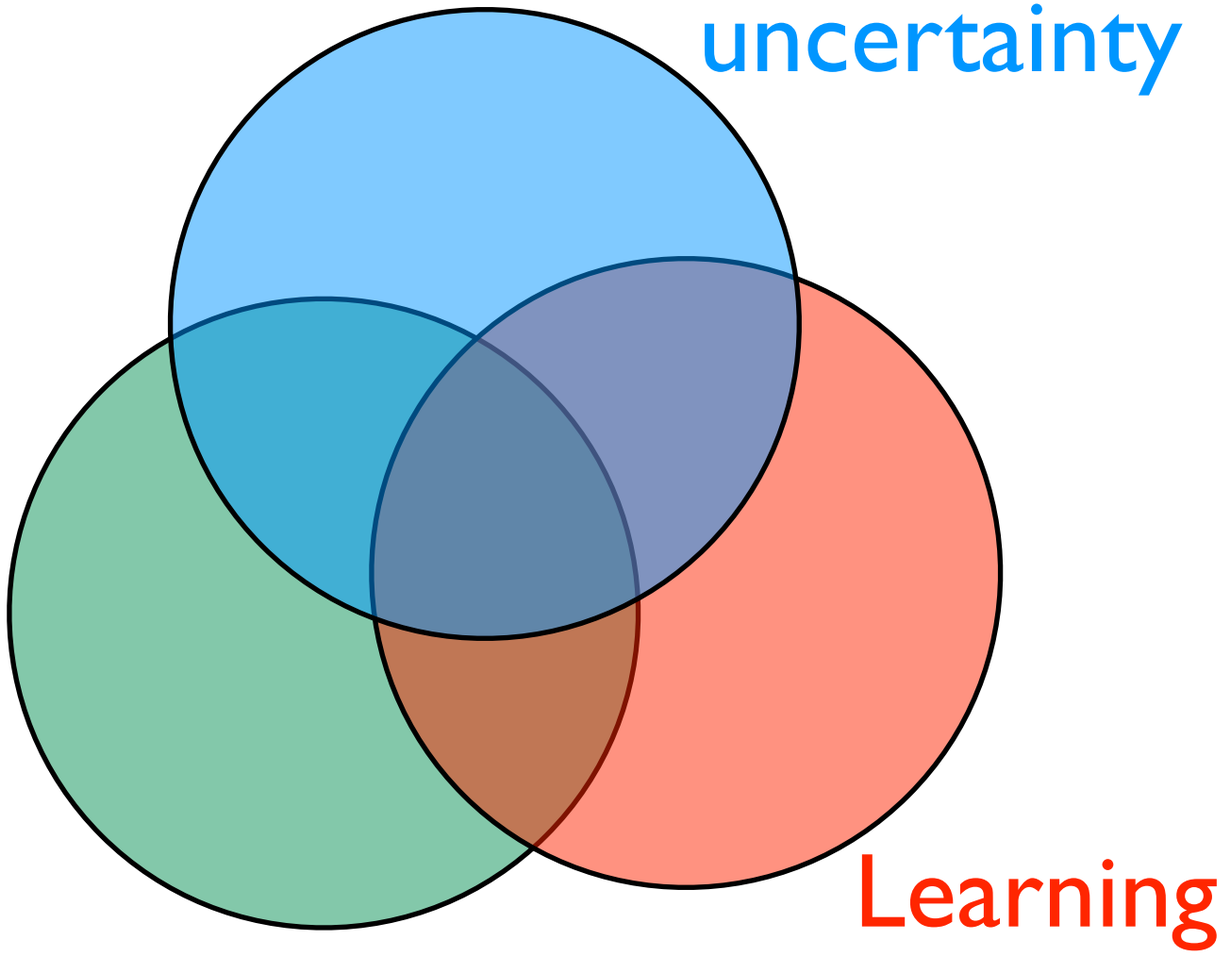
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Probabilistic Databases

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

one world

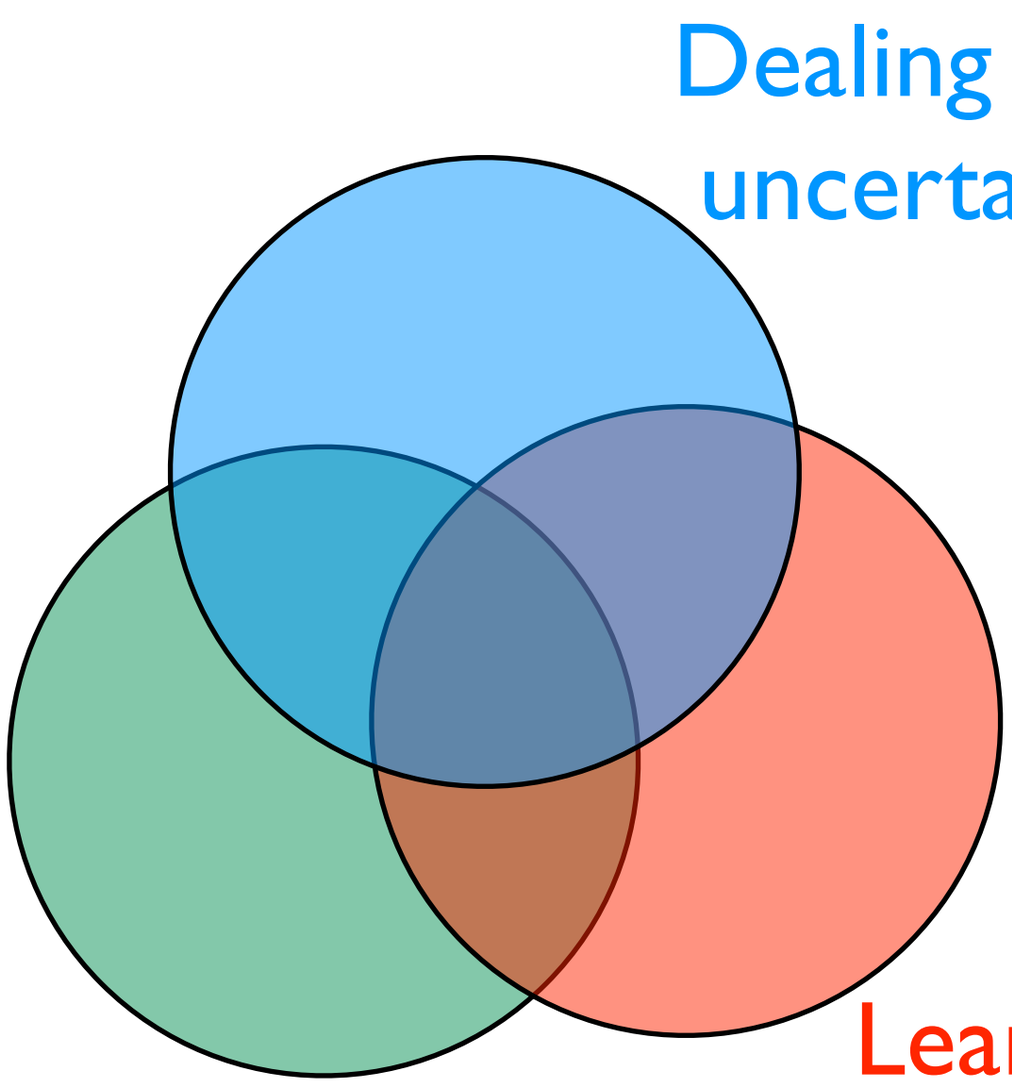
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Probabilistic Databases

bornIn

person	city	P
ann	london	0.87
bob	york	0.95
eve	new york	0.90
tom	paris	0.56

cityIn

city	country	P
london	uk	0.99
york	uk	0.75
paris	usa	0.40

tuples as random variables

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

one world

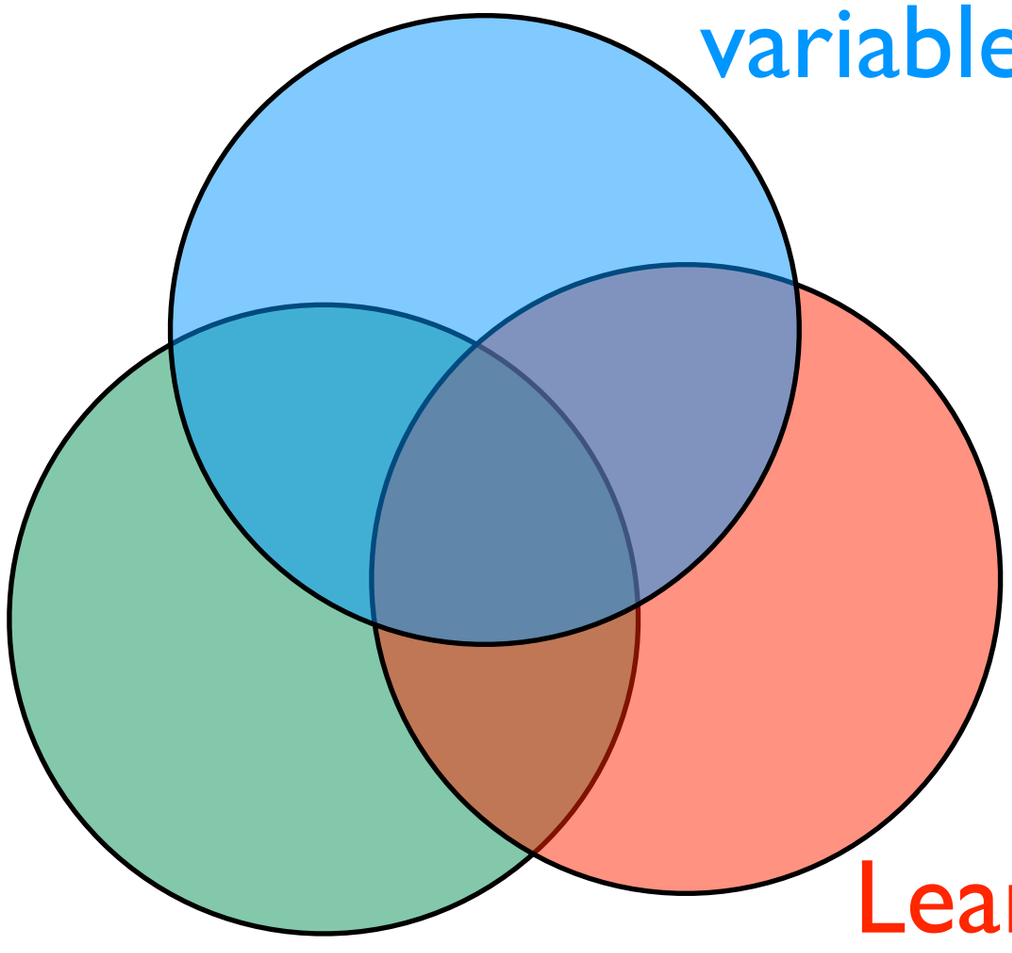
bornIn

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

several possible worlds

bornIn

person	city	P
ann	london	0.87
bob	york	0.95
eve	new york	0.90
tom	paris	0.56

cityIn

city	country	P
london	uk	0.99
york	uk	0.75
paris	usa	0.40

tuples as random variables

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

one world

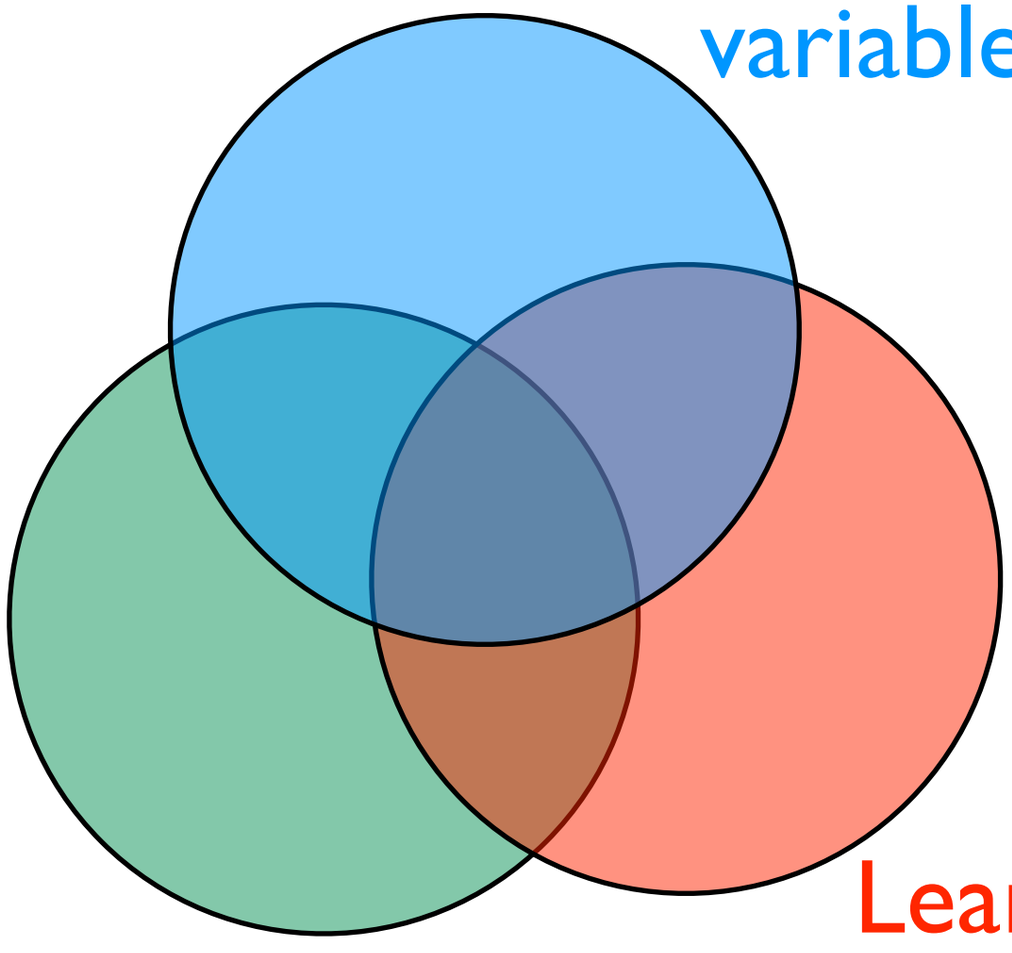
bornIn

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

several possible worlds

bornIn

person	city	P
ann	london	0.87
bob	york	0.95
		0.90
		0.56

cityIn

city	country	P
london	uk	0.99
york	uk	0.75
paris	usa	0.40

tuples as random

probabilistic tables + database queries
→ distribution over possible worlds

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

one world

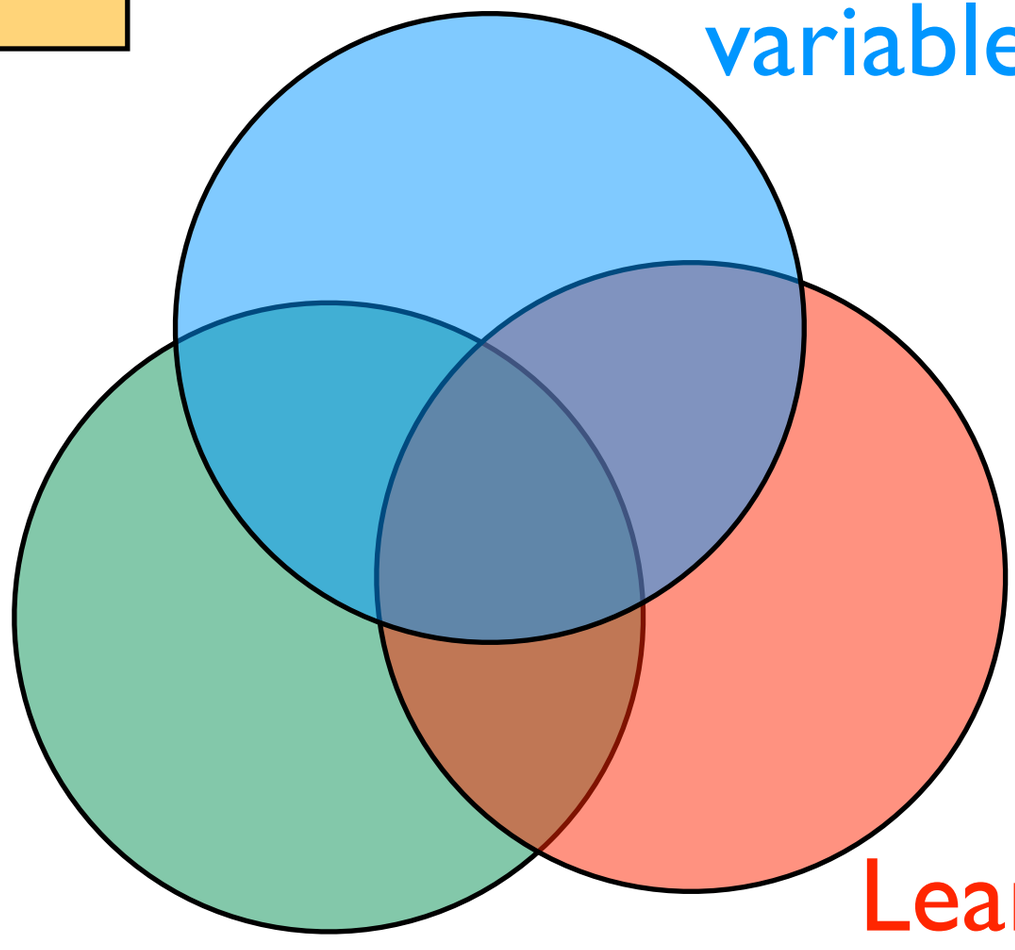
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Example: Information Extraction

Recently-Learned Facts twitter Refresh

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

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

Example: Information Extraction

Recently-Learned Facts twitter Refresh

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

instances for many different relations

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

Example: Information Extraction

Recently-Learned Facts twitter Refresh

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

instances for many different relations

degree of certainty

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

Querying: relational database

ProducesProduct

Company	Product
sony	walkman
microsoft	mac_os_x
ibm	personal_computer
microsoft	mac_os
adobe	adobe_indesign
adobe	adobe_dreamweaver
...	...

HeadquarteredIn

Company	City
microsoft	redmond
ibm	san_jose
emirates_airlines	dubai
honda	torrance
horizon	seattle
egyptair	cairo
adobe	san_jose
...	...

Querying: relational database

ProducesProduct

Company	Product
sony	walkman
microsoft	mac_os_x
ibm	personal_computer
microsoft	mac_os
adobe	adobe_indesign
adobe	adobe_dreamweaver
...	...

HeadquarteredIn

Company	City
microsoft	redmond
ibm	san_jose
emirates_airlines	dubai
honda	torrance
horizon	seattle
egyptair	cairo
adobe	san_jose
...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Querying: relational database

ProducesProduct

Company	Product
sony	walkman
microsoft	mac_os_x
ibm	personal_computer
microsoft	mac_os
adobe	adobe_indesign
adobe	adobe_dreamweaver
...	...

HeadquarteredIn

Company	City
microsoft	redmond
ibm	san_jose
emirates_airlines	dubai
honda	torrance
horizon	seattle
egyptair	cairo
adobe	san_jose
...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Product	Company
personal_computer	ibm
adobe_indesign	adobe
adobe_dreamweaver	adobe

Querying: relational database

ProducesProduct

Company	Product
sony	walkman
microsoft	mac_os_x
ibm	personal_computer
microsoft	mac_os
adobe	adobe_indesign
adobe	adobe_dreamweaver
...	...

HeadquarteredIn

Company	City
microsoft	redmond
ibm	san_jose
emirates_airlines	dubai
honda	torrance
horizon	seattle
egyptair	cairo
adobe	san_jose
...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Product	Company
personal_computer	ibm
adobe_indesign	adobe
adobe_dreamweaver	adobe

Querying: relational database

ProducesProduct

Company	Product
sony	walkman
microsoft	mac_os_x
ibm	personal_computer
microsoft	mac_os
adobe	adobe_indesign
adobe	adobe_dreamweaver
...	...

HeadquarteredIn

Company	City
microsoft	redmond
ibm	san_jose
emirates_airlines	dubai
honda	torrance
horizon	seattle
egyptair	cairo
adobe	san_jose
...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Product	Company
personal_computer	ibm
adobe_indesign	adobe
adobe_dreamweaver	adobe

Querying: relational database

ProducesProduct

Company	Product
sony	walkman
microsoft	mac_os_x
ibm	personal_computer
microsoft	mac_os
adobe	adobe_indesign
adobe	adobe_dreamweaver
…	…

HeadquarteredIn

Company	City
microsoft	redmond
ibm	san_jose
emirates_airlines	dubai
honda	torrance
horizon	seattle
egyptair	cairo
adobe	san_jose
…	…

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Product	Company
personal_computer	ibm
adobe_indesign	adobe
adobe_dreamweaver	adobe

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

same query -

probabilities handled implicitly

(more on this later)

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

0.96x0.99=0.95

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

0.9x0.93=0.83

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

0.87x0.93=0.80

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

answer tuples ranked by probability

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

PDB with tuple-level uncertainty in ProbLog?

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

PDB with tuple-level uncertainty in ProbLog?

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

PDB with tuple-level uncertainty in ProbLog?

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

```
0.96::producesProduct(sony,walkman).
0.96::producesProduct(microsoft,mac_os_x).
0.96::producesProduct(ibm,personal_computer).
0.9::producesProduct(microsoft,mac_os).
0.9::producesProduct(adobe,adobe_indesign).
0.87::producesProduct(adobe,adobe_dreamweaver).
...
```

PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

```
result(Product,Company) :-
    producesProduct(Company,Product),
    headquarteredIn(Company,san_jose).
query(result(_,_)).
```

PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

```
result(Product,Company) :-
    producesProduct(Company,Product),
    headquarteredIn(Company,san_jose).
query(result(_,_)).
```

PDB with tuple-level uncertainty in ProbLog?

```
0.96::producesProduct(sony,walkman).
0.96::producesProduct(microsoft,mac_os_x).
0.96::producesProduct(ibm,personal_computer).
0.9::producesProduct(microsoft,mac_os).
0.9::producesProduct(adobe,adobe_indesign).
0.87::producesProduct(adobe,adobe_dreamweaver).
...
1.00::headquarteredIn(microsoft,redmond).
0.99::headquarteredIn(ibm,san_jose).
0.93::headquarteredIn(emirates_airlines,dubai).
0.93::headquarteredIn(honda,torrance).
0.93::headquarteredIn(horizon,seattle).
0.93::headquarteredIn(egyptair,cairo).
0.93::headquarteredIn(adobe,san_jose).
...
      result(Product,Company) :- producesProduct(Company,Product),
                                 headquarteredIn(Company,san_jose).

      query(result(_,_)).
```

PDB with attribute-level uncertainty in ProbLog?

color

item	color	P
mug	green blue	0.65 0.35
plate	pink red purple	0.23 0.14 0.63

PDB with attribute-level uncertainty in ProbLog?

color

item	color	P
mug	green blue	0.65 0.35
plate	pink red purple	0.23 0.14 0.63

```
0.65::color(mug,green); 0.35::color(mug,blue) <- true.
0.23::color(plate,pink); 0.14::color(plate,red);
                                0.63::color(plate,purple) <- true.
```

Complexity of querying

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

0.96x0.99=0.95

0.9x0.93=0.83

0.87x0.93=0.80

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

Complexity of querying

ProducesProduct

Company	Product	P
...	...	...

HeadquarteredIn

Company	City	P
...	...	...

```
select distinct x.Product, x.Company,
                x.P * y.P as P
from ProducesProduct x, HeadquarteredIn y
where x.Company = y.Company
      and y.City = 'san_jose'
order by P desc
```

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

0.96x0.99=0.95

0.9x0.93=0.83

0.87x0.93=0.80

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

Complexity of querying

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
result(Product,Company) :-
   producesProduct(Company,Product),
   headquarteredIn(Company,san_jose).
query(result(_,_)).
```

Complexity of querying

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
result(Product,Company) :-
    producesProduct(Company,Product),
    headquarteredIn(Company,san_jose).
query(result(_,_)).
```

each ground query has a **single proof**

→ no disjoint-sum-problem,

easy evaluation

Complexity of querying

- Singe proof per query is a rare case
- In general: more complex reasoning
  - Lifted inference
  - Grounded inference
- Let use take a step back and
  - recap databases
  - introduce formal language
  - analyze complexity

What Everyone Should Know about Databases

- Database = several relations (a.k.a. tables)
- SQL Query = FO Formula
- Boolean Query = FO Sentence

What Everyone Should Know about Databases

Database: relations (= tables)

D =

Smoker

X	Y
Alice	2009
Alice	2010
Bob	2009
Carol	2010

Friend

X	Z
Alice	Bob
Alice	Carol
Bob	Carol
Carol	Bob

What Everyone Should Know about Databases

Database: relations (= tables)

D =

Smoker

X	Y
Alice	2009
Alice	2010
Bob	2009
Carol	2010

Friend

X	Z
Alice	Bob
Alice	Carol
Bob	Carol
Carol	Bob

Query: First Order Formula

$Q(z) = \exists x\,(\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$

Find friends of smokers in 2009

Query answer: Q(D) =

Z
Bob
Carol

Conjunctive Queries CQ = FO(∃, ∧)
Union of Conjunctive Queries UCQ = FO(∃, ∧, ∨)

What Everyone Should Know about Databases

Database: relations (= tables)

D =

Smoker

X	Y
Alice	2009
Alice	2010
Bob	2009
Carol	2010

Friend

X	Z
Alice	Bob
Alice	Carol
Bob	Carol
Carol	Bob

Query: First Order Formula

$Q(z) = \exists x\,(\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$

Find friends of smokers in 2009

Query answer: Q(D) =

Z
Bob
Carol

Conjunctive Queries CQ = FO(∃, ∧)
Union of Conjunctive Queries UCQ = FO(∃, ∧, ∨)

Boolean Query: FO Sentence

$Q = \exists x\,(\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,\text{'Bob'}))$

Query answer: Q(D) = TRUE

What Everyone Should Know about Databases

Declarative Query	→	Query Plan
"what"	→	*"how"*

What Everyone Should Know about Databases

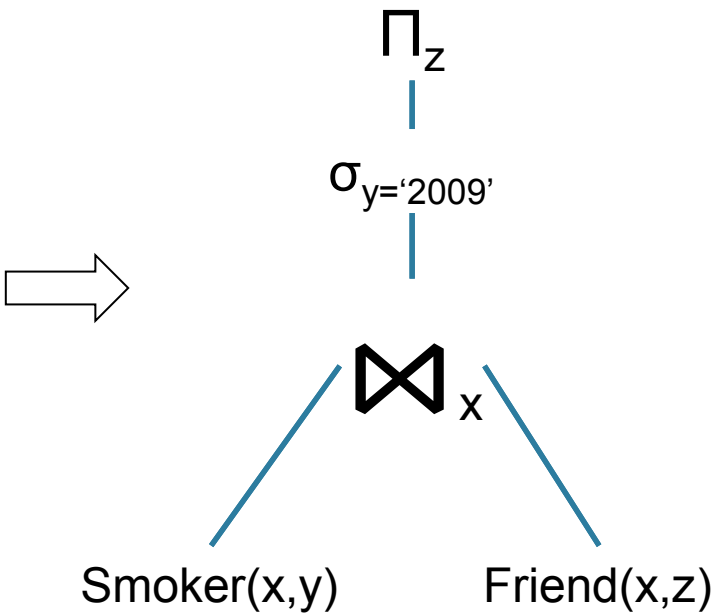
Declarative Query	→	Query Plan
"what"	→	*"how"*

$Q(z) = \exists x\ (\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$

What Everyone Should Know about Databases

Declarative Query → Query Plan
"what" → *"how"*

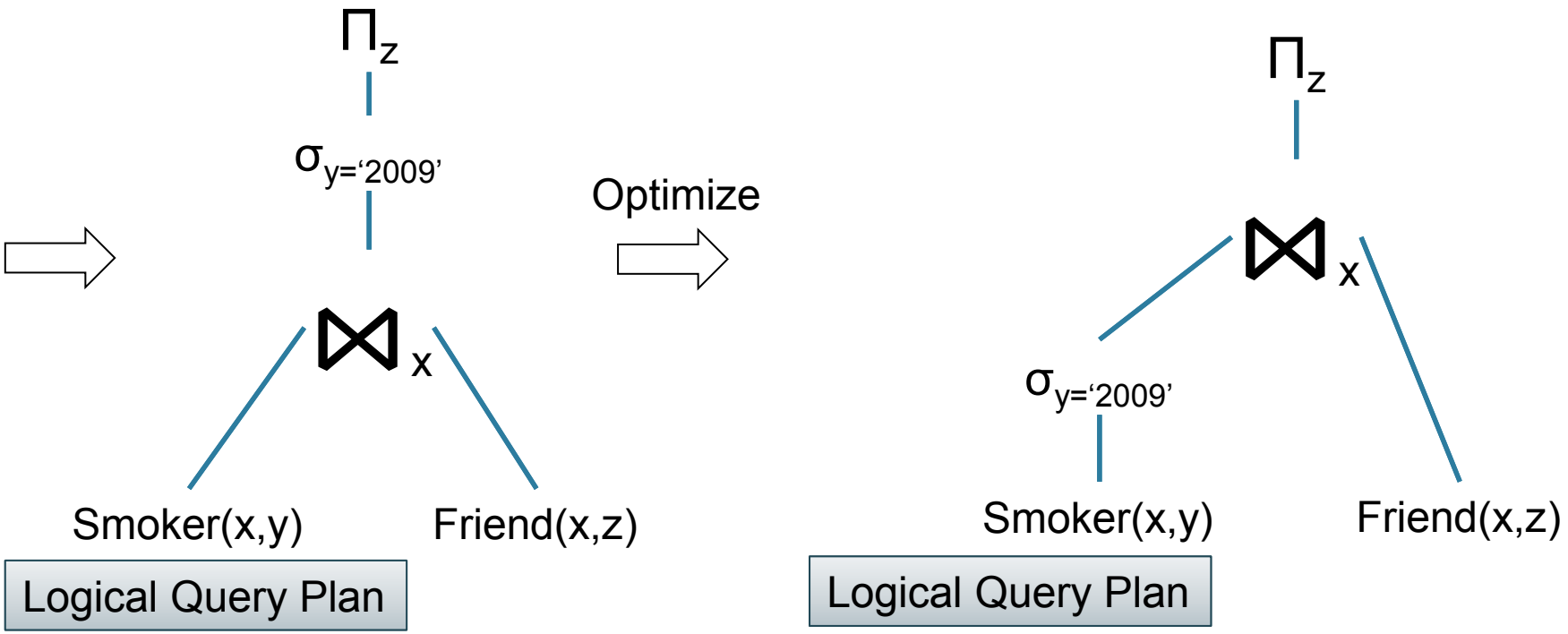
$Q(z) = \exists x\ (Smoker(x,'2009') \wedge Friend(x,z))$



What Everyone Should Know about Databases

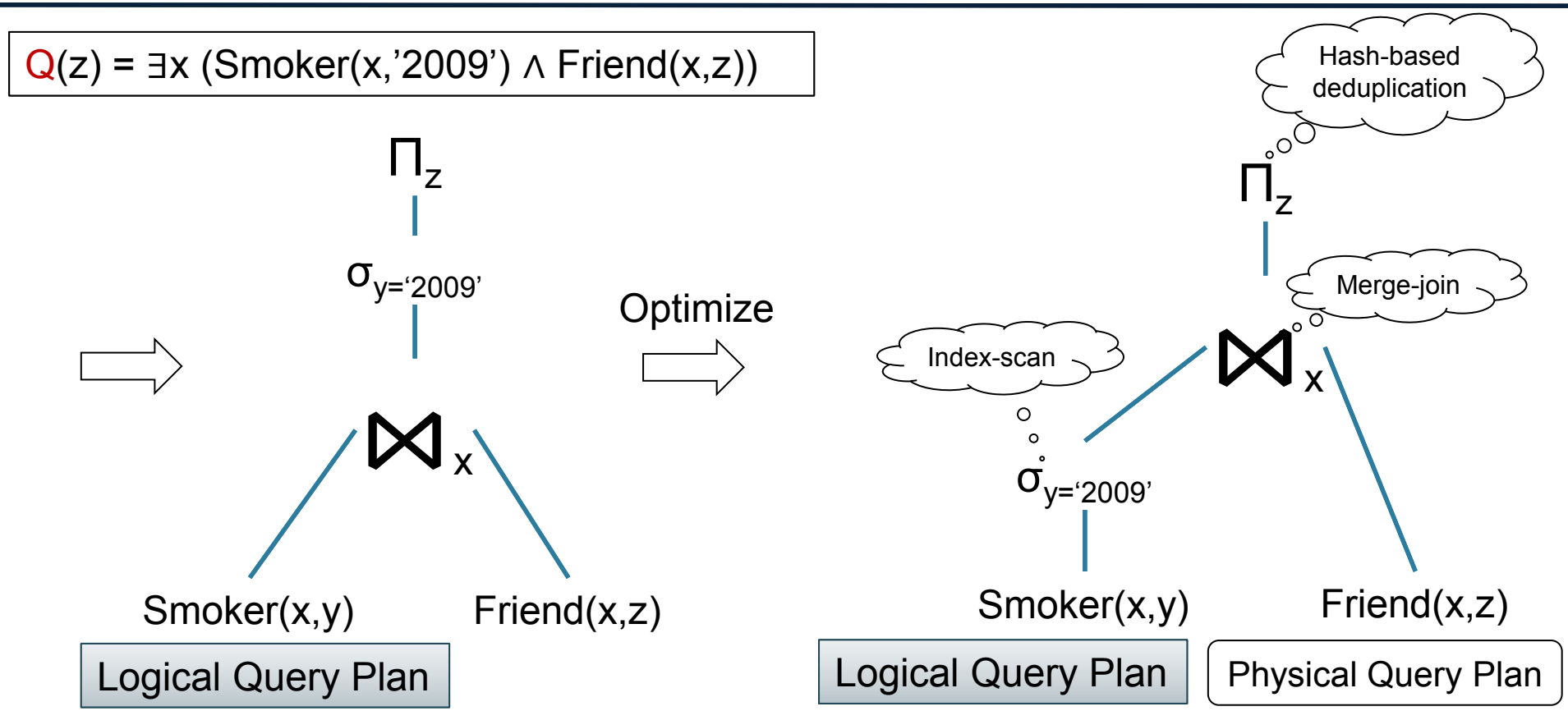
Declarative Query → Query Plan
"what" → *"how"*

$Q(z) = \exists x\ (Smoker(x,'2009') \wedge Friend(x,z))$



What Everyone Should Know about Databases

Declarative Query → Query Plan
"what" → *"how"*



What Every Researcher Should Know about Databases

Problem: compute Q(D)

Moshe Vardi [Vardi'82]
2008 ACM SIGMOD Contribution Award

[Vardi'82]

What Every Researcher Should Know about Databases

Problem: compute Q(D)

- Data complexity:
  fix Q, complexity = f(D)

Moshe Vardi [Vardi'82]
2008 ACM SIGMOD Contribution Award

[Vardi'82]

What Every Researcher Should Know about Databases

Problem: compute Q(D)

Moshe Vardi [Vardi'82]
2008 ACM SIGMOD Contribution Award

- Data complexity:
  fix Q, complexity = f(D)

  Query complexity: (expression complexity)
  fix D, complexity = f(Q)

- Combined complexity:
  complexity = f(D,Q)

[Vardi'82]

Probabilistic Databases

- A probabilistic database = relational database where each tuple has an associated probability

- Semantics = probability distribution over possible worlds (deterministic databases)

Example

Probabilistic database D:

Friend

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

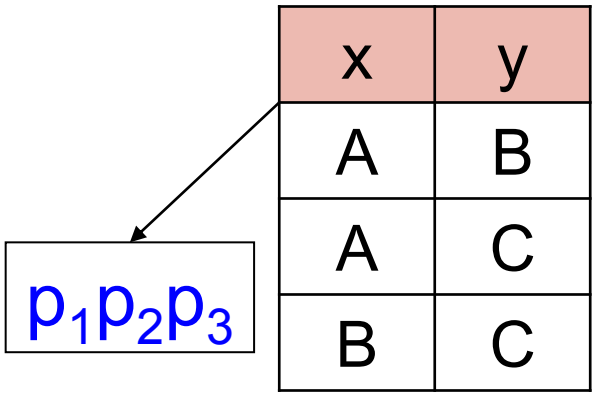
Example

Probabilistic database D:

Friend

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Possible worlds semantics:



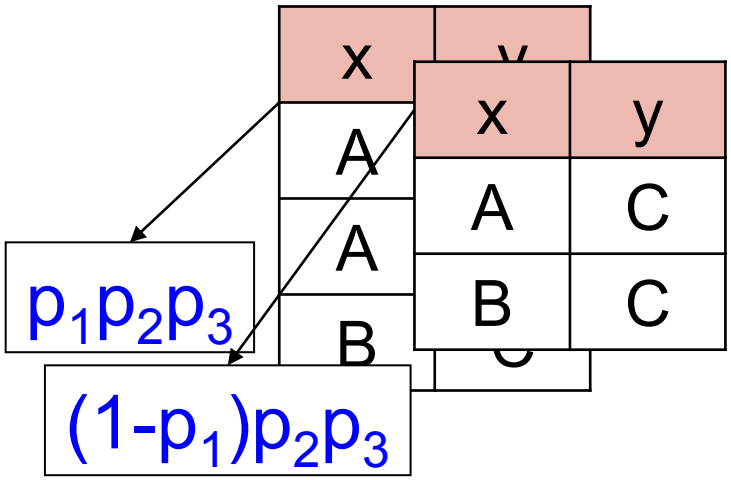
Example

Probabilistic database D:

Friend

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Possible worlds semantics:



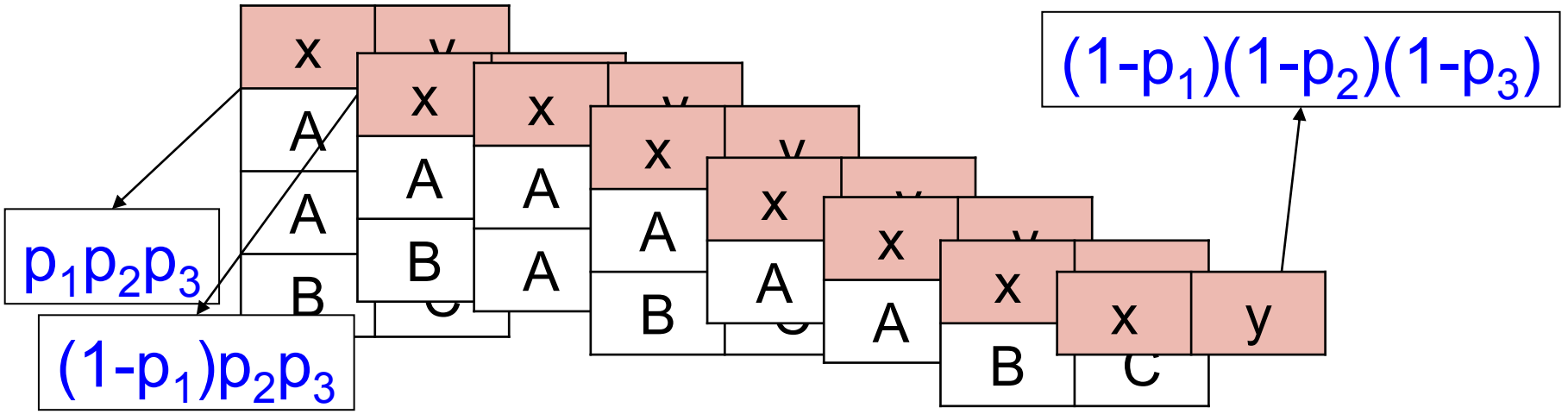
Example

Probabilistic database D:

Friend

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Possible worlds semantics:



Query Semantics

Fix a Boolean query Q

Fix a probabilistic database D:

P (Q | D) = marginal probability of Q
on possible words of D

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$P(Q \mid D) =$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$P(Q \mid D) =$ $1-(1-q_1)*(1-q_2)$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$P(Q \mid D) = p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$P(Q \mid D) =$

$$p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$$
$$1-(1-q_3)*(1-q_4)*(1-q_5)$$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$P(Q \mid D) =$

$p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$

$p_2 * [\ 1-(1-q_3)*(1-q_4)*(1-q_5)\ ]$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$$P(Q \mid D) = 1 - \{1 - p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]\} * \{1 - p_2 * [\ 1-(1-q_3)*(1-q_4)*(1-q_5)\ ]\}$$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$$P(Q \mid D) = 1 - \{1 - p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]\} * \{1 - p_2 * [\ 1-(1-q_3)*(1-q_4)*(1-q_5)\ ]\}$$

One can compute $P(Q \mid D)$ in PTIME in the size of the database D

Smoker

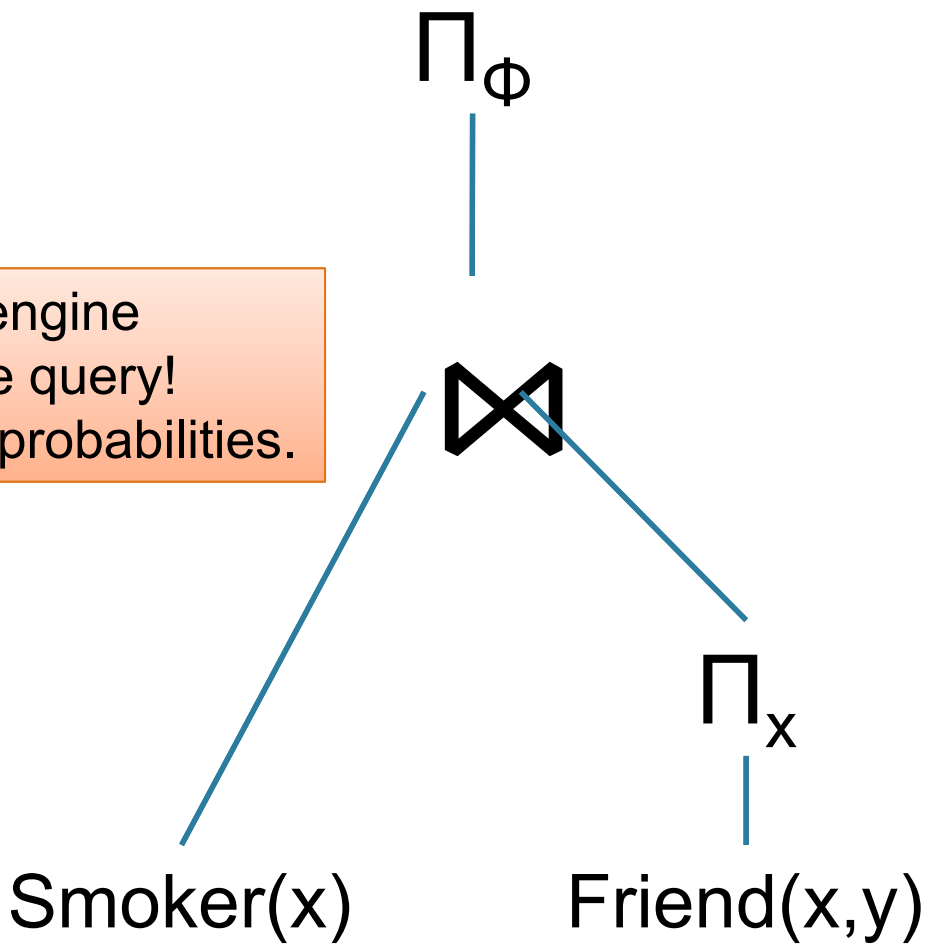
x	P
A	p_1
B	p_2
C	p_3

Friend

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example



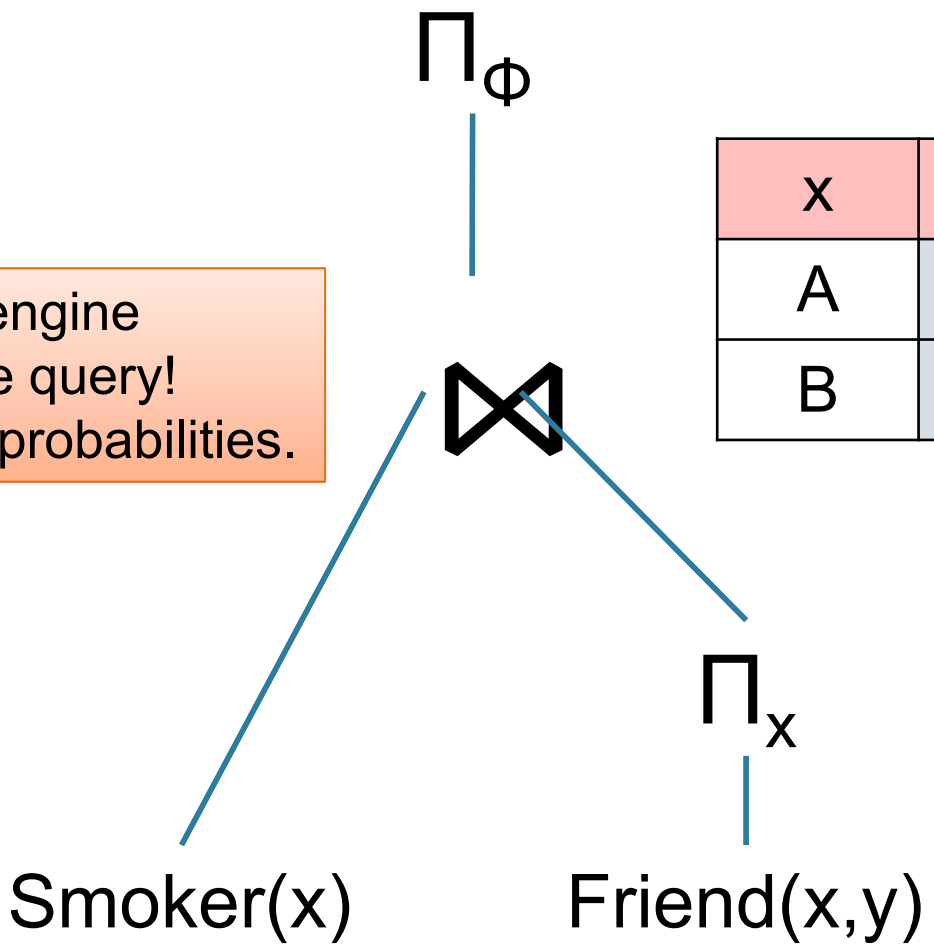
Use the SQL engine
to compute the query!
Aggregate on probabilities.

x	P
A	p_1
B	p_2
C	p_3

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example



Use the SQL engine to compute the query! Aggregate on probabilities.

x	P
A	$1-(1-q_1)(1-q_2)$
B	$1-(1-q_4)(1-q_5)\ (1-q_6)$

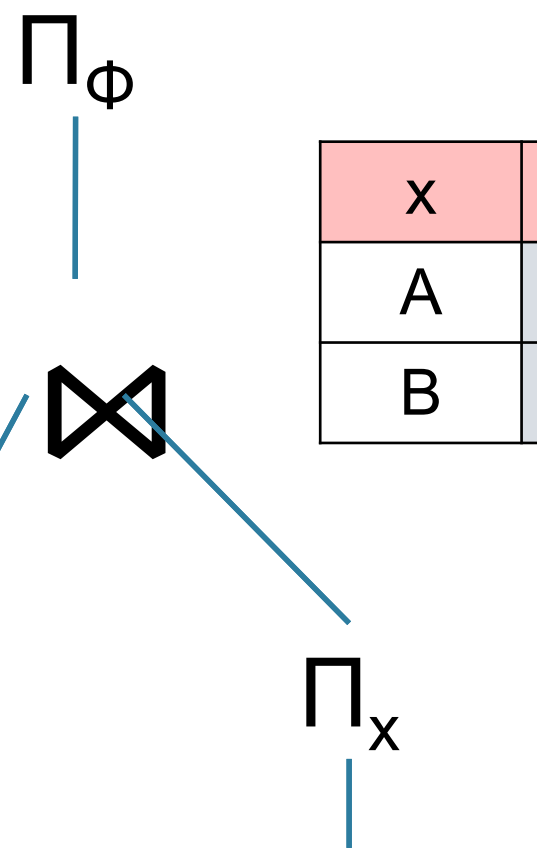
x	P
A	p_1
B	p_2
C	p_3

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$1-\{1-p_1[1-(1-q_1)(1-q_2)]\}^* \{1-p_2[1-(1-q_4)(1-q_5)\ (1-q_6)]\}$



Use the SQL engine to compute the query! Aggregate on probabilities.

x	P
A	$1-(1-q_1)(1-q_2)$
B	$1-(1-q_4)(1-q_5)\ (1-q_6)$

x	P
A	p_1
B	p_2
C	p_3

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Problem Statement

Given: probabilistic database D, query Q

Compute: P(Q | D)

Data complexity: fix Q, complexity = f(|D|)

Approaches to Compute P(Q | D)

- Propositional inference:
  - Ground the query Q → $F_{Q,D}$, compute P($F_{Q,D}$)
  - This is Weighted Model Counting (see ProbLog)
  - Works for every query Q
  - But: may be exponential in |D| (data complexity)

- Lifted inference:
  - Compute a query plan for Q, execute plan on D
  - Always polynomial time in |D| (data complexity)
  - But: does not work for all queries Q

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

- If $Q[C_1/x]$, $Q[C_2/x]$, … are independent
  $\forall$-Rule: $P(\forall z\, Q) = \prod_{C \in Domain} P(Q[C/z])$
  $\exists$-Rule: $P(\exists z\, Q) = 1 - \prod_{C \in Domain} (1 - P(Q[C/z]))$

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

- If $Q[C_1/x]$, $Q[C_2/x]$, … are independent
  $\forall$-Rule: $P(\forall z\, Q) = \prod_{C \in Domain} P(Q[C/z])$
  $\exists$-Rule: $P(\exists z\, Q) = 1 - \prod_{C \in Domain} (1 - P(Q[C/z]))$

- Inclusion/Exclusion formula:
  $P(Q_1 \vee Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \wedge Q_2)$
  $P(Q_1 \wedge Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \vee Q_2)$

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

- If $Q[C_1/x]$, $Q[C_2/x]$, … are independent
  $\forall$-Rule: $P(\forall z\, Q) = \prod_{C \in Domain} P(Q[C/z])$
  $\exists$-Rule: $P(\exists z\, Q) = 1 - \prod_{C \in Domain} (1 - P(Q[C/z]))$

- Inclusion/Exclusion formula:
  $P(Q_1 \vee Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \wedge Q_2)$
  $P(Q_1 \wedge Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \vee Q_2)$

Negation: $P(\neg Q) = 1 - P(Q)$

Example

$Q = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y))$ $= \forall x\, (\text{Smoker}(x) \vee \forall y\, \text{Friend}(x,y))$

∀-Rule

$P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\, \text{Friend}(A,y))$

Check independence:

Smoker(Alice) ∨ ∀y Friend(Alice,y)

Smoker(Bob) ∨ ∀y Friend(Bob,y)

Example

$Q = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y))$ $= \forall x\, (\text{Smoker}(x) \vee \forall y\, \text{Friend}(x,y))$

∀-Rule

$P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\, \text{Friend}(A,y))$

Check independence:
Smoker(Alice) ∨ ∀y Friend(Alice,y)
Smoker(Bob) ∨ ∀y Friend(Bob,y)

$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y\, \text{Friend}(A,y)))$

OR-Rule

Example

$Q = \forall x \forall y\ (\text{Smoker}(x) \vee \text{Friend}(x,y))$ $= \forall x\ (\text{Smoker}(x) \vee \forall y\ \text{Friend}(x,y))$

∀-Rule

$P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\ \text{Friend}(A,y))$

Check independence:
Smoker(Alice) ∨ ∀y Friend(Alice,y)
Smoker(Bob) ∨ ∀y Friend(Bob,y)

$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y\ \text{Friend}(A,y)))$

OR-Rule

$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - \prod_{B \in \text{Domain}} P(\text{Friend}(A,B)))$

∀-Rule

Example

$Q = \forall x \forall y\ (\text{Smoker}(x) \vee \text{Friend}(x,y))$ $= \forall x\ (\text{Smoker}(x) \vee \forall y\ \text{Friend}(x,y))$

∀-Rule

$P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\ \text{Friend}(A,y))$

Check independence:
Smoker(Alice) ∨ ∀y Friend(Alice,y)
Smoker(Bob) ∨ ∀y Friend(Bob,y)

$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y\ \text{Friend}(A,y)))$

OR-Rule

$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - \prod_{B \in \text{Domain}} P(\text{Friend}(A,B)))$

Lookup the probabilities in the database

∀-Rule

Example

$Q = \forall x \forall y\ (\text{Smoker}(x) \vee \text{Friend}(x,y))$ $= \forall x\ (\text{Smoker}(x) \vee \forall y\ \text{Friend}(x,y))$

∀-Rule

$$P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\ \text{Friend}(A,y))$$

Check independence:
Smoker(Alice) ∨ ∀y Friend(Alice,y)
Smoker(Bob) ∨ ∀y Friend(Bob,y)

$$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y\ \text{Friend}(A,y)))$$

OR-Rule

$$P(Q) = \prod_{A \in \text{Domain}} 1 - (1 - P(\text{Smoker}(A))) \times (1 - \prod_{B \in \text{Domain}} P(\text{Friend}(A,B)))$$

Lookup the probabilities in the database

∀-Rule

Runtime = $O(n^2)$.

Discussion: CNF vs. DNF

Databases		KR/AI	
Conjunctive Queries CQ	FO(∃, ∧)	Positive Clause	FO(∀, ∨)
Union of Conjunctive Queries UCQ	FO(∃, ∧, ∨) = ∃ Positive-DNF	Positive FO	FO(∀, ∧, ∨) = ∀ Positive-CNF
UCQ with "safe negation" $UCQ^{\neg}$	∃ DNF	First Order CNF	∀ CNF

Q = ∃x,∃y, Smoker(x) ∧ Friend(x,y)

Q = ∀x∀y (Smoker(x) ∨ Friend(x,y))

By duality we can reduce one problem to the other:

∃x,∃y, Smoker(x) ∧ Friend(x,y) = ¬ ∀x,∀y, (¬Smoker(x) ∨ ¬Friend(x,y))

P(∃x,∃y, Smoker(x) ∧ Friend(x,y)) = 1-P(∀x,∀y, (¬Smoker(x) ∨ ¬Friend(x,y)))

Discussion

Lifted Inference Sometimes Fails

$H_0 = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y))$

No rule applies here!

The ∀-rule does not apply, because H_0[Alice/x] and H_0[Bob/x] are dependent:

$H_0[\text{Alice}/x] = \forall y\, (\text{Smoker}(\text{Alice}) \vee \text{Friend}(\text{Alice},y) \vee \text{Jogger}(y))$
$H_0[\text{Bob}/x] = \forall y\, (\text{Smoker}(\text{Bob}) \vee \text{Friend}(\text{Bob},y) \vee \text{Jogger}(y))$

Dependent

Discussion

Lifted Inference Sometimes Fails

$H_0 = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y))$

No rule applies here!

The ∀-rule does not apply, because $H_0[\text{Alice}/x]$ and $H_0[\text{Bob}/x]$ are dependent:

$$H_0[\text{Alice}/x] = \forall y\, (\text{Smoker}(\text{Alice}) \vee \text{Friend}(\text{Alice},y) \vee \text{Jogger}(y))$$
$$H_0[\text{Bob}/x] = \forall y\, (\text{Smoker}(\text{Bob}) \vee \text{Friend}(\text{Bob},y) \vee \text{Jogger}(y))$$

Dependent

Theorem. [Dalvi'04] Computing $P(H_0 \mid D)$ is #P-hard in $|D|$

See next slides

Discussion

Lifted Inference Sometimes Fails

$H_0 = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y))$

No rule applies here!

The $\forall$-rule does not apply, because $H_0[\text{Alice}/x]$ and $H_0[\text{Bob}/x]$ are dependent:

$$H_0[\text{Alice}/x] = \forall y\, (\text{Smoker}(\text{Alice}) \vee \text{Friend}(\text{Alice},y) \vee \text{Jogger}(y))$$
$$H_0[\text{Bob}/x] = \forall y\, (\text{Smoker}(\text{Bob}) \vee \text{Friend}(\text{Bob},y) \vee \text{Jogger}(y))$$

Dependent

Theorem. [Dalvi'04] Computing $P(H_0 \mid D)$ is #P-hard in $|D|$

See next slides

Consequence: assuming PTIME ≠ #P, H_0 is not liftable!

Two Questions

- Q1: Are the lifted rules complete?
  - We know that they get stuck on some queries
  - Do we need to add more rules?

- Q2: Are lifted rules stronger than grounded?
  - Some lifted rules easily correspond to operations on grounded formulas (e.g. Independent-AND)
  - Can we simulate every lifted inference directly on the grounded formula?

Two Questions

- Q1: Are the lifted rules complete?
  - We know that they get stuck on some queries
  - Do we need to add more rules?

Complete for Positive CNF-FO, for UCQ

- Q2: Are lifted rules stronger than grounded?
  - Some lifted rules easily correspond to operations on grounded formulas (e.g. Independent-AND)
  - Can we simulate every lifted inference directly on the grounded formula?

Strictly stronger than existing grounded algorithms

NP vs. #P

- SAT = Satisfiability Problem
- SAT is NP-complete [Cook'71]
- NP = *decision problems* solved by polynomial-time, nondeterministic TM
- NP-hard means at least as difficult as any problem in NP
- NP-complete problems are (i) in NP and (ii) NP-hard

- #SAT = model counting (number of SAT solutions)
- #SAT is #P-complete [Valiant'79]
- #P = *numerical functions* solved by polynomial-time, nondeterministic TM, answer = #accepting computations

Note: it would be wrong to say "#SAT is NP-complete"

1. Are the Lifted Rules Complete?

We use complexity classes

- Inference rules: PTIME data complexity
- Some queries: #P-hard data complexity

Dichotomy Theorem for Positive CNF-FO:

- If lifted rules succeed, then query in PTIME
- If lifted rules fail, then query is #P-hard

Implies lifted rules are complete for Positive CNF-FO

A Simple Propositional Formula that is Hard

A **Positive, Partitioned 2CNF** Formula is a formula of the form:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

where E = set of tuples with disjoint X and Y.

A Simple Propositional Formula that is Hard

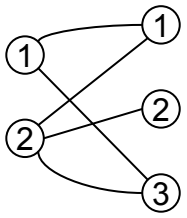
A **Positive, Partitioned 2CNF** Formula is a formula of the form:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

where E = set of tuples with disjoint X and Y.

$$F = (x_1 \vee y_1) \wedge (x_2 \vee y_1) \wedge (x_2 \vee y_3) \wedge (x_1 \vee y_3) \wedge (x_2 \vee y_2)$$

E:



A Simple Propositional Formula that is Hard

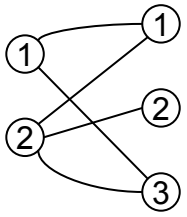
A **Positive, Partitioned 2CNF** Formula is a formula of the form:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

where E = set of tuples with disjoint X and Y.

$$F = (x_1 \vee y_1) \wedge (x_2 \vee y_1) \wedge (x_2 \vee y_3) \wedge (x_1 \vee y_3) \wedge (x_2 \vee y_2)$$

E:



Theorem [Provan'83] #SAT for PP2CNF is #P-hard

A Query That is #P-Hard

$H_0 = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y))$

Theorem. Computing $P(H_0 \mid D)$ is #P-hard in $|D|$

[Dalvi'04]

Proof: Reduction from PP2CNF. Given a PP2CNF F defined by edge relation E, set:

$P(\text{Friend}(a,b)) = 1$ if $(a,b) \in E$

$P(\text{Friend}(a,b)) = 0$ if $(a,b) \notin E$

$P(\text{Smoker}(.)) = P(\text{Jogger}(.)) = 0.5$

Then the grounding of H_0 is: $\bigwedge_{(i,j) \in E} (\text{Smoker}(i) \vee \text{Jogger}(j))$

Hence, #SAT = $P(H_0 \mid D) \times 2^{(|\text{Smoker}|+|\text{Jogger}|)}$

Lesson: no lifted inference rules will ever compute $P(H_0 \mid D)$

Hierarchical Clause

at(x) = set of atoms containing the variable x

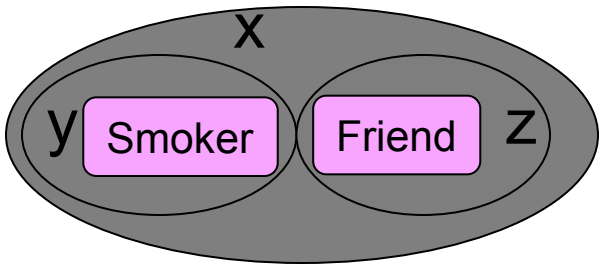
Definition A clause Q is hierarchical if forall variables x, y:

$at(x) \supseteq at(y)$ or $at(x) \supseteq at(y)$ or $at(x) \cap at(y) = \varnothing$

Hierarchical

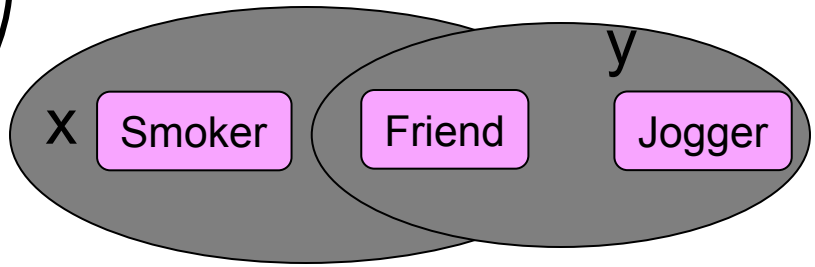
$Q = (Smoker(x,y) \vee Friend(x,z))$

$= \forall x[\forall y\ Smoker(x,y)] \vee [\forall z\ Friend(x,z)]$



Non-hierarchical

$H_0 = Smoker(x) \vee Friend(x,y) \vee Jogger(y)$



Small Dichotomy Theorem

Let Q be a single clause, w/o repeating relation symbols

Theorem [Dalvi'04] Dichotomy:

- If Q is hierarchical, then Q is liftable (PTIME data complexity)
- If Q is not hierarchical, Q is #P-hard

And, moreover, the OR-rule and ∀-rule are complete.

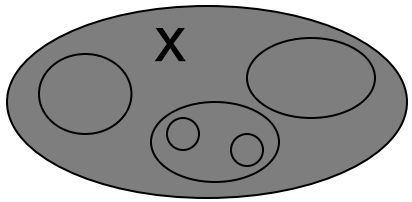
Proof

Hierarchical → PTIME

Proof

Hierarchical → PTIME

Case 1:

Q= 

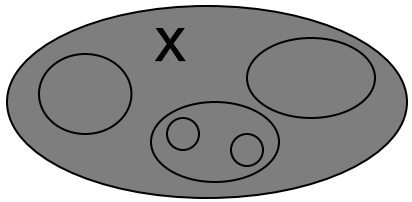
∀-Rule:

$P(\forall x\ Q) = \Pi_a\ P(Q[a/x])$

Proof

Hierarchical → PTIME

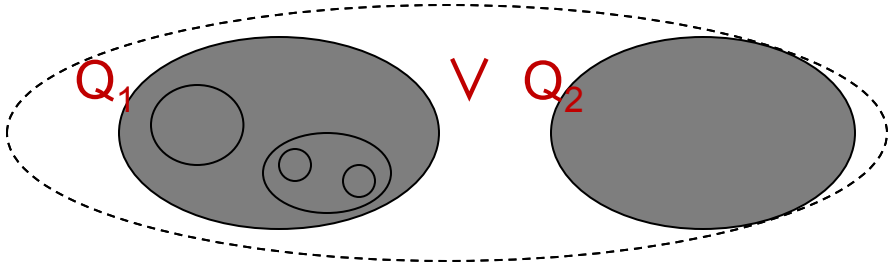
Case 1:

Q= 

∀-Rule:

$P(\forall x\, Q) = \Pi_a P(Q[a/x])$

Case 2:

Q= 

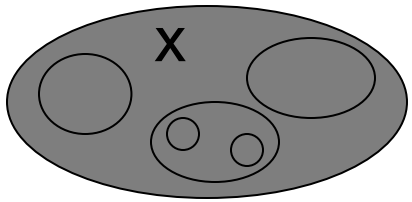
V −Rule:

$P(Q) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

Proof

Hierarchical → PTIME

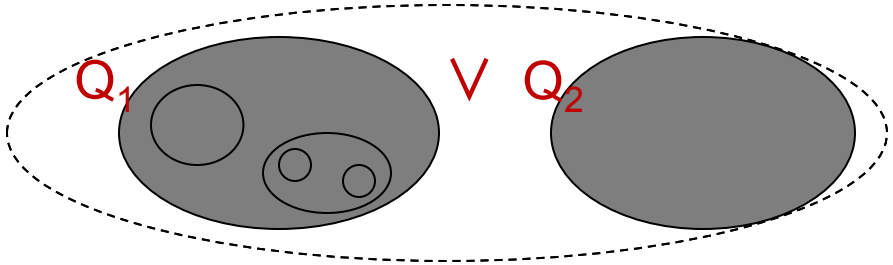
Case 1:

Q= 

∀-Rule:

$P(\forall x\, Q) = \Pi_a P(Q[a/x])$

Case 2:

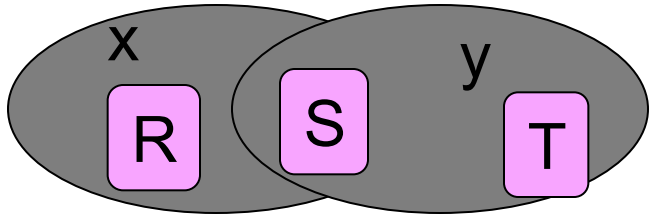
Q= 

V −Rule:

$P(Q) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

Non-hierarchical → #P-hard

Reduction from H_0:



$Q = \ldots R(x, \ldots) \vee S(x,y,\ldots) \vee T(y,\ldots), \ldots$

The Big Dichotomy Theorem

Dichotomy Theorem [Dalvi'12] Fix a Positive-CNF Q.

1. If Q is liftable, then P(Q) is in PTIME (obviously)
2. If Q is not liftable, then P(Q) is #P-complete

Note 1: for the theorem to hold one needs a generalization of the inclusion/exclusion

Note 2: Original formulation for UCQ; holds for Positive CNF-FO by duality.

Summary

- Database D = relations
- Query Q = FO
- Query plans, query optimization
- Data complexity: fix Q, complexity f(D)
- Probabilistic DB's = independent tuples
- Lifted inference: simple, but fails sometimes
- Lifted inference rules complete for Positive CNF and UCQ

Probabilistic Programming with Church

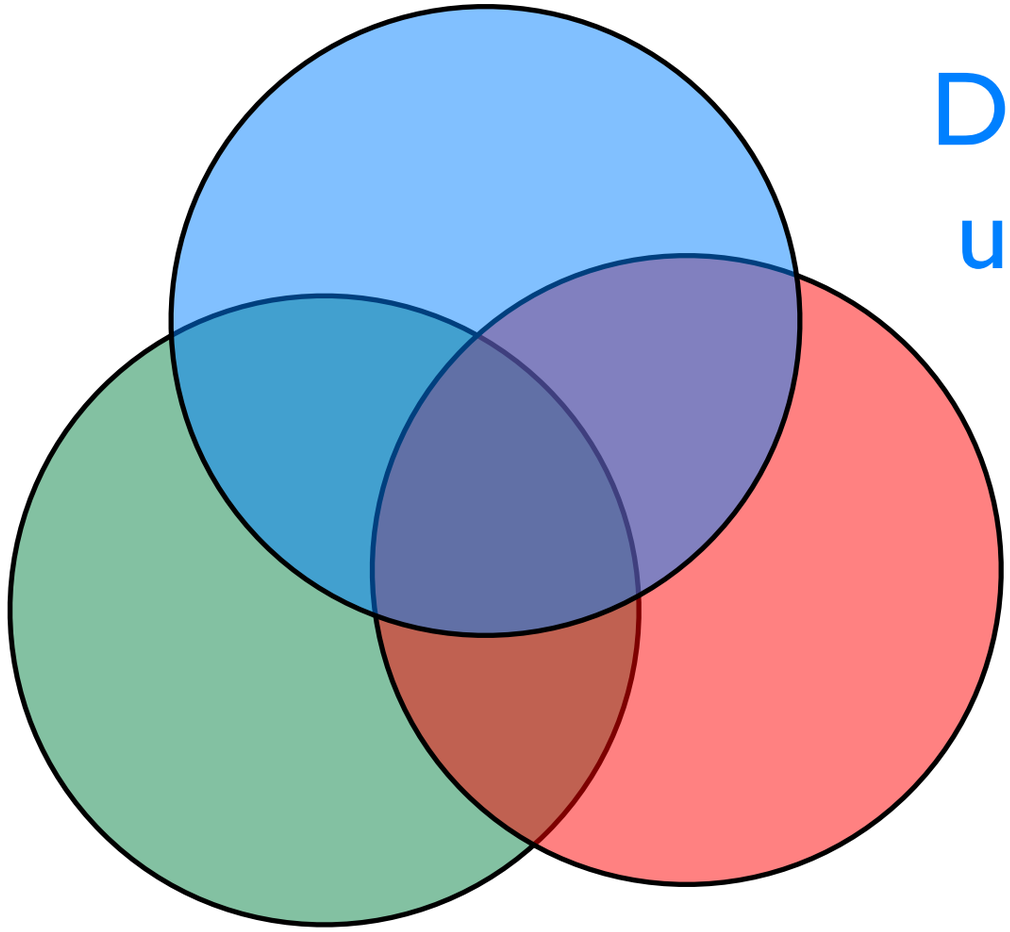
And related languages such as Anglican, Venture, …

http://probmods.org

Church

probabilistic functional programming

[Goodman et al, UAI 08]



Dealing with uncertainty

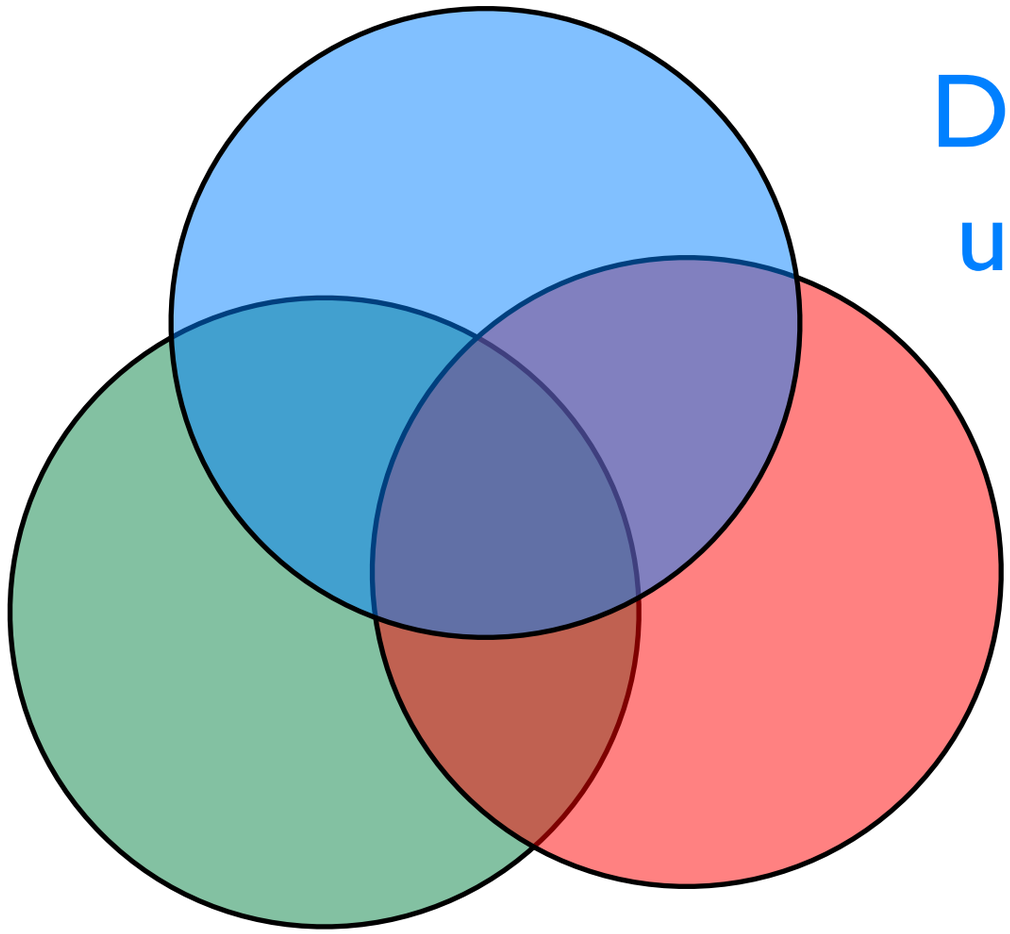
Reasoning with relational data

Learning

Church

probabilistic functional programming

[Goodman et al, UAI 08]



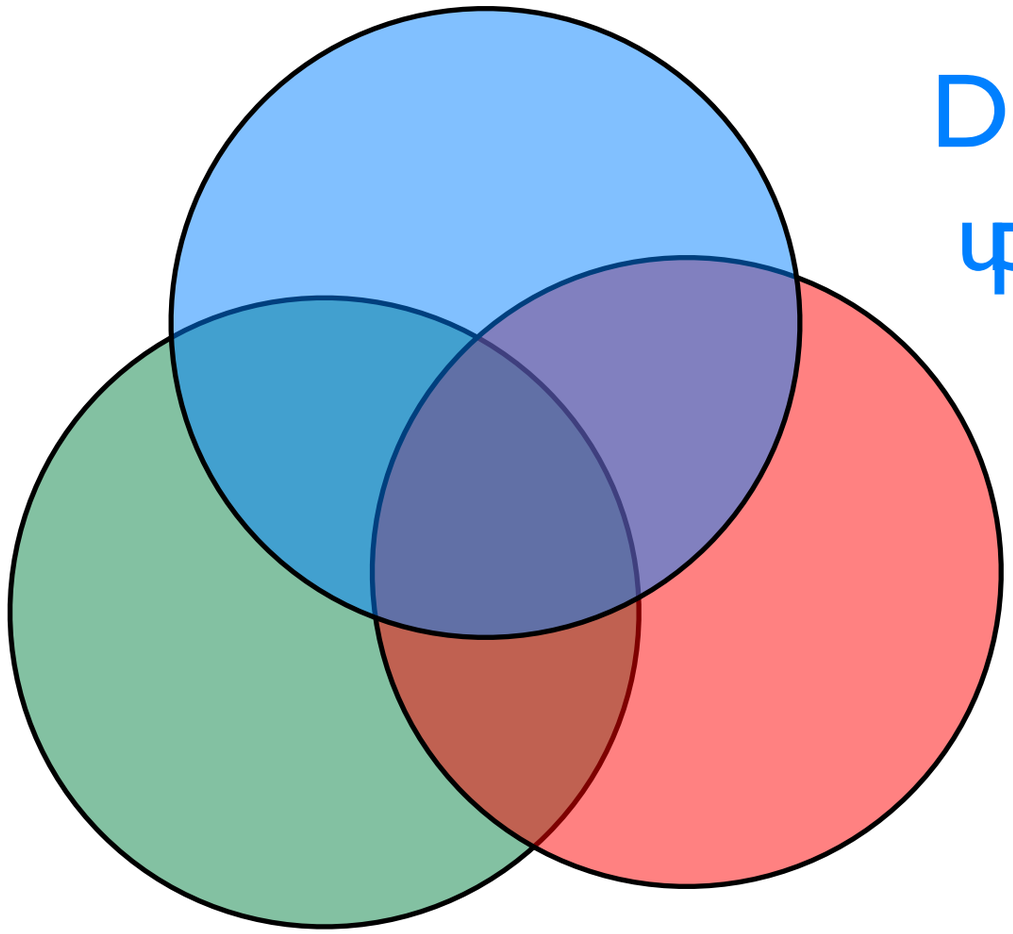

```
(define plus5 (lambda (x) (+ x 5)))(map plus5
'(1 2 3))
```

Church

probabilistic functional programming

[Goodman et al, UAI 08]

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))(map ran
```



Dealing with uncertainty

Reasoning with logical data

Learning

one execution

```
(define plus5 (lambda (x) (+ x 5)))(map plus5
'(1 2 3))
```

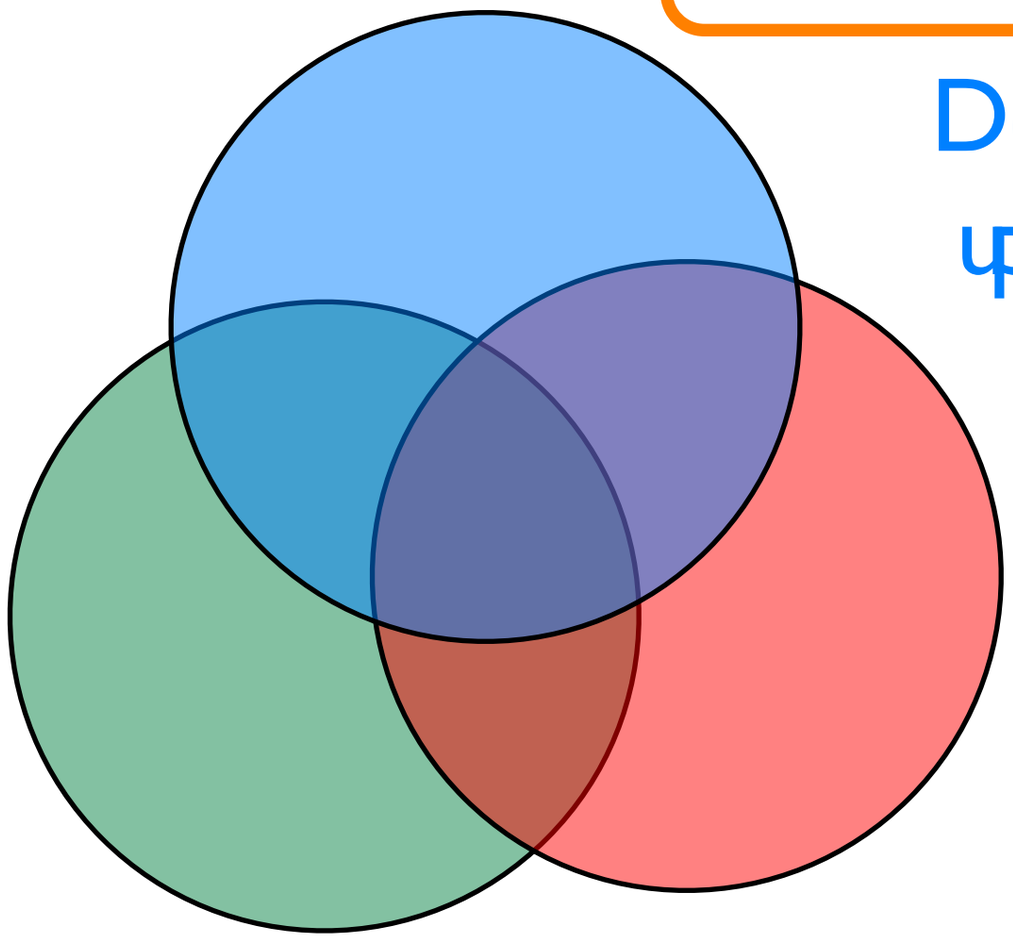
Church

probabilistic functional programming

[Goodman et al, UAI 08]

several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))(map ran
```



Dealing with uncertainty

Reasoning with logical data

Reasoning with relational data

one execution

```
(define plus5 (lambda (x) (+ x 5)))(map plus5
'(1 2 3))
```

Learning

Church

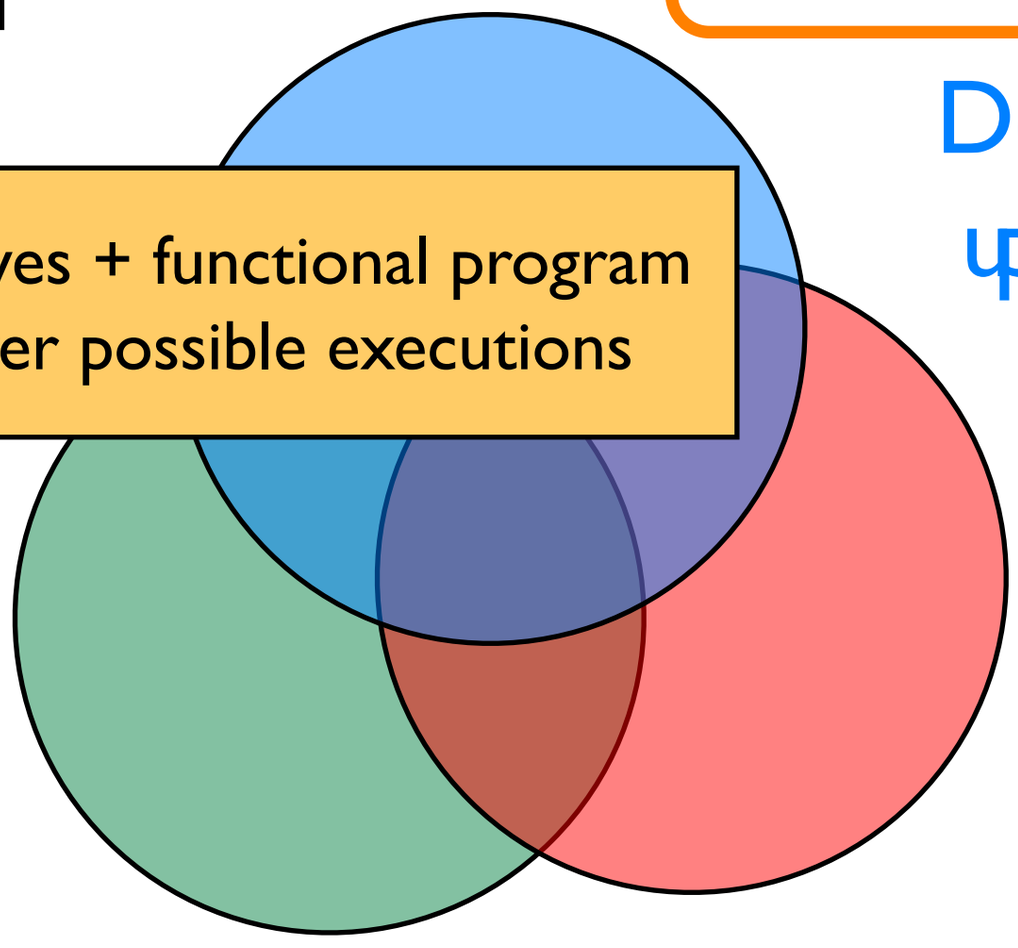
probabilistic functional programming

[Goodman et al, UAI 08]

several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))(map ran
```

probabilistic primitives + functional program
→ distribution over possible executions



one execution

```
(define plus5 (lambda (x) (+ x 5)))(map plus5
'(1 2 3))
```

Church Example

function

arguments

definition

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2 3))
```

apply function to each element

list

random primitive
P(true) = 0.6

Church Example

function

arguments

definition

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2 3))
```

apply function to each element

list

random primitive
P(true) = 0.6

direct execution: samples result

Church Example

function

arguments

definition

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2 3))
```

apply function to each element

list

random primitive
P(true) = 0.6

direct execution: samples result

sampling also supports continuous RVs, e.g.,
`(* (gaussian 0 1) (gaussian 0 1))`

Computing probability distribution

```
(
enumeration-query

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2))

true
)
```

enumerates all executions & sums probabilities per result

query

evidence

(((1 2) (1 7) (6 2) (6 7)) (0.16 0.24 0.24 0.36))

in ProbLog?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2))
```

in ProbLog?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2))
```

```
0.4::p5(N,N);0.6::p5(N,M) <- M is N+5.lp5([],[]).lp5([N|L],
[M|K]) :-    p5(N,M),    lp5(L,K).query(lp5([1,2],_)).
```

in ProbLog?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2))
```

```
0.4::p5(N,N);0.6::p5(N,M) <- M is N+5.lp5([],[]).lp5([N|L],
[M|K]) :-    p5(N,M),    lp5(L,K).query(lp5([1,2],_)).
```

what if the list is [1,1]?

in ProbLog?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2))
```

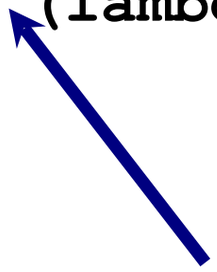
```
0.4::p5(N,N);0.6::p5(N,M) <- M is N+5.lp5([],[]).lp5([N|L],
[M|K]) :-    p5(N,M),    lp5(L,K).query(lp5([1,2],_)).
```

what if the list is [1,1]?

```
0.4::p5(N,N,ID);0.6::p5(N,M,ID) <- M is N+5.lp5([],
[]).lp5([N|L],[M|K]) :-    p5(N,M,L),
lp5(L,K).query(lp5([1,2],_)).
```

Stochastic Memoization

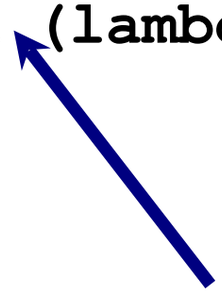
```
(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))
(map randplus5 '(1 1))
```



remember first value &
reuse for all later calls

Stochastic Memoization

```
(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))
(map randplus5 '(1 1))
```

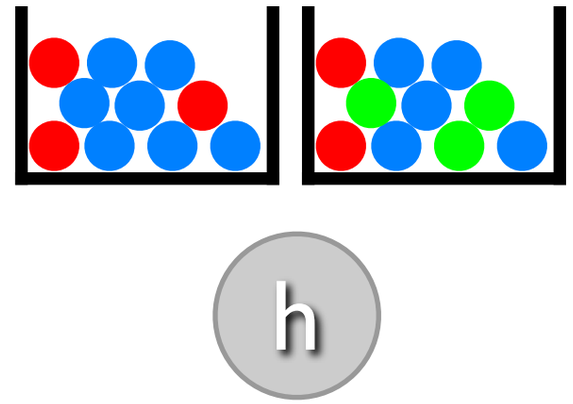


remember first value &
reuse for all later calls

ProbLog memoizes by default
Church does not

Church by example:

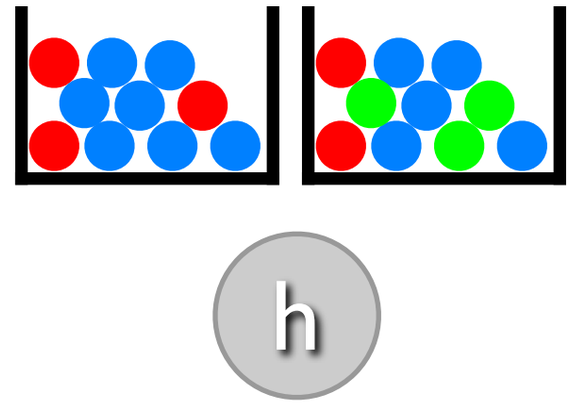
A bit of gambling



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

Church by example:

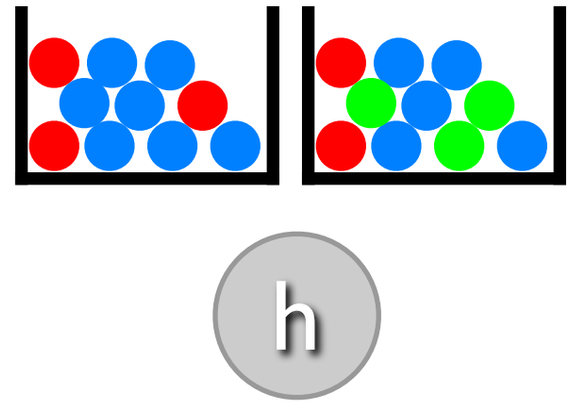
A bit of gambling



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

Church by example:

A bit of gambling

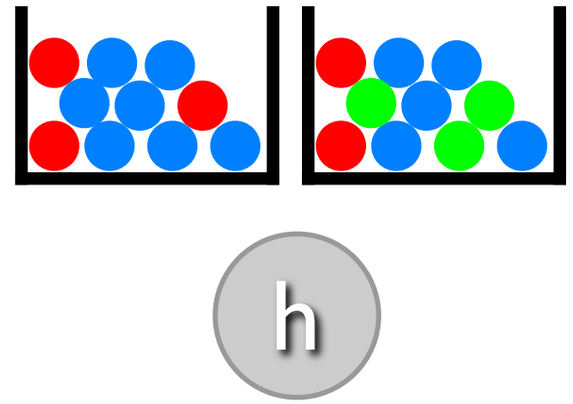


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same colo

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

Church by example:

A bit of gambling

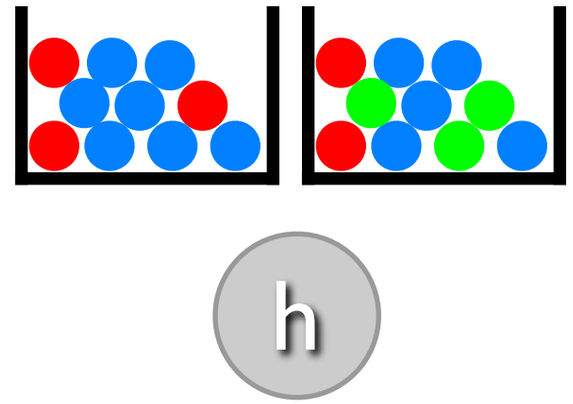


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same colo

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

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

Church by example:

A bit of gambling

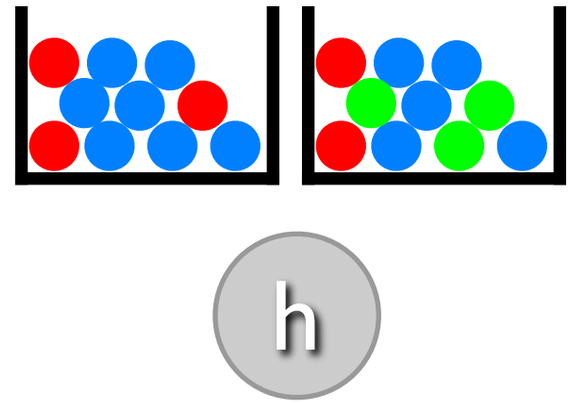


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same colo

```
(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)))))
```

Church by example:

A bit of gambling

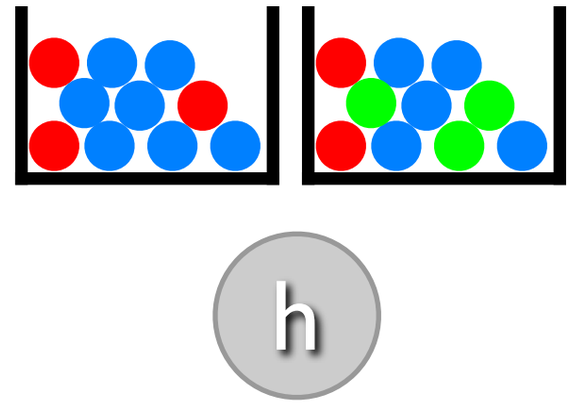


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same colo

```
(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)))
```

Church by example:

A bit of gambling

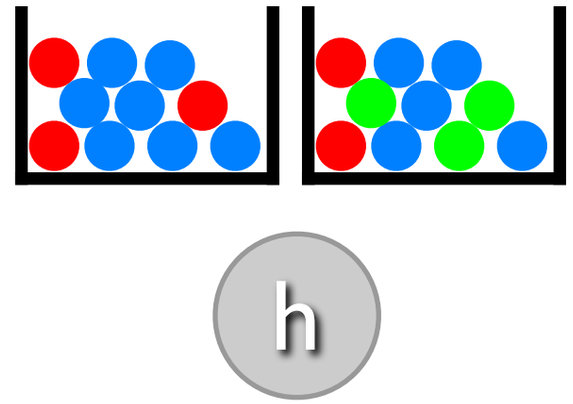


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same colo

```
(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))
```

Church by example:

A bit of gambling

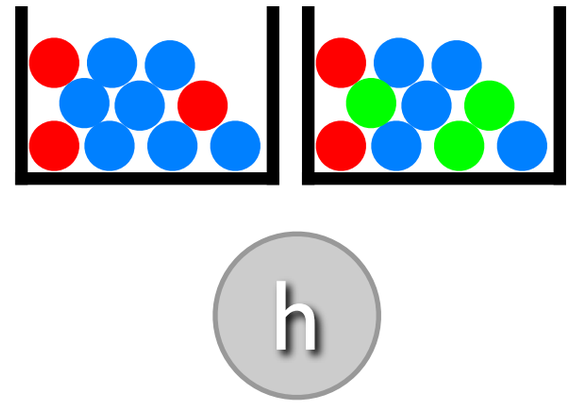


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same colo

```
(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)))
```

Church 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 colo

```
(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))
```

Sampling execution

```
(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))
```

```
win
```

query

Marginals via enumeration

```
(enumeration-query
 (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))

win
true )
```

query

evidence

Histogram via sampling

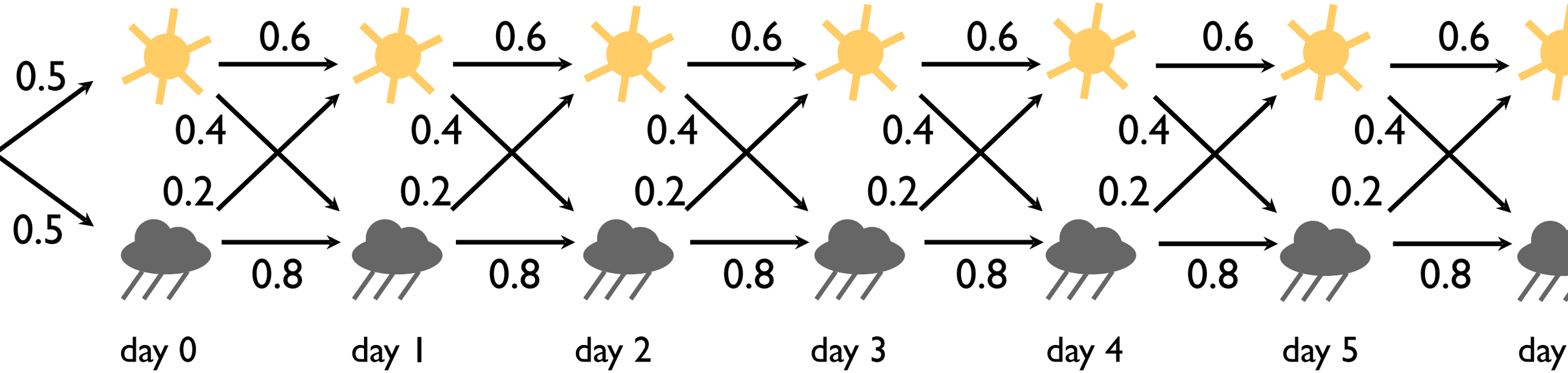
```
(hist (repeat 1000 (lambda ()
                     (rejection-query

(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))

win
true ))))
```

query

evidence

Church by example:

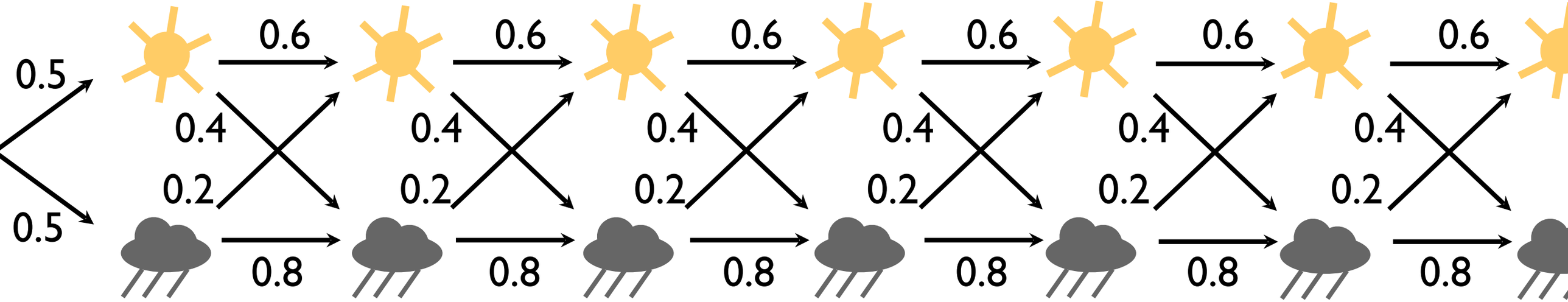
Rain or sun?




```
(define weather (mem (lambda (day) (if (equal? day 0)
(weather0)                          (weatherN day (- day 1))))))(defin
(lambda () (if (flip 0.5) 'sun 'rain)))(define weatherN (lambda (toda
yesterday)                    (if (equal? (weather yesterday) 'rain)
(if (flip 0.2) 'sun 'rain)                          (if (flip 0.6) 'sun
```

Church by example:

Rain or sun?




```
(define weather (mem (lambda (day) (if (equal? day 0)
(weather0) (weatherN day (- day 1))))))(defin
(lambda () (if (flip 0.5) 'sun 'rain)))(define weatherN (lambda (toda
yesterday) (if (equal? (weather yesterday) 'rain)
(if (flip 0.2) 'sun 'rain) (if (flip 0.6) 'sun

(list (weather 0) (weather 1) (weather 2))
```

Probabilistic Programming Summary

- Church: functional programming + random primitives
- Probabilistic generative model
- stochastic memoization
- Sampling
- Increasing number of probabilistic programming languages using various underlying paradigms (Venture, Anglican, …)

Language Extensions and Variants

- Dynamics under uncertainty
- Continuous-valued random variables
- Decision making
- Constraint

not included in ProbLog system

Language Extensions and Variants

variants

Causal Probabilistic Time-Logic *dynamics*

Distributional Clauses *continuous distributions*

ProbLog

extensions

utilities and decisions Decision-Theoretic ProbLog

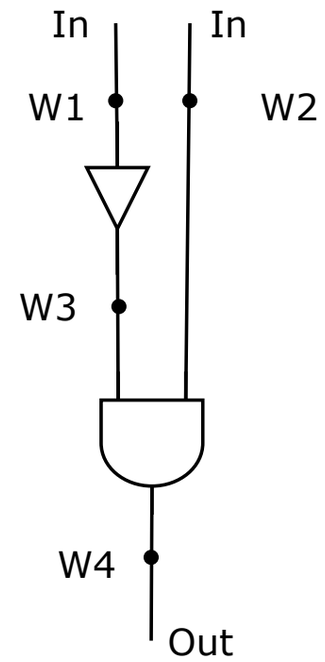
constraints ProbLog with constraints

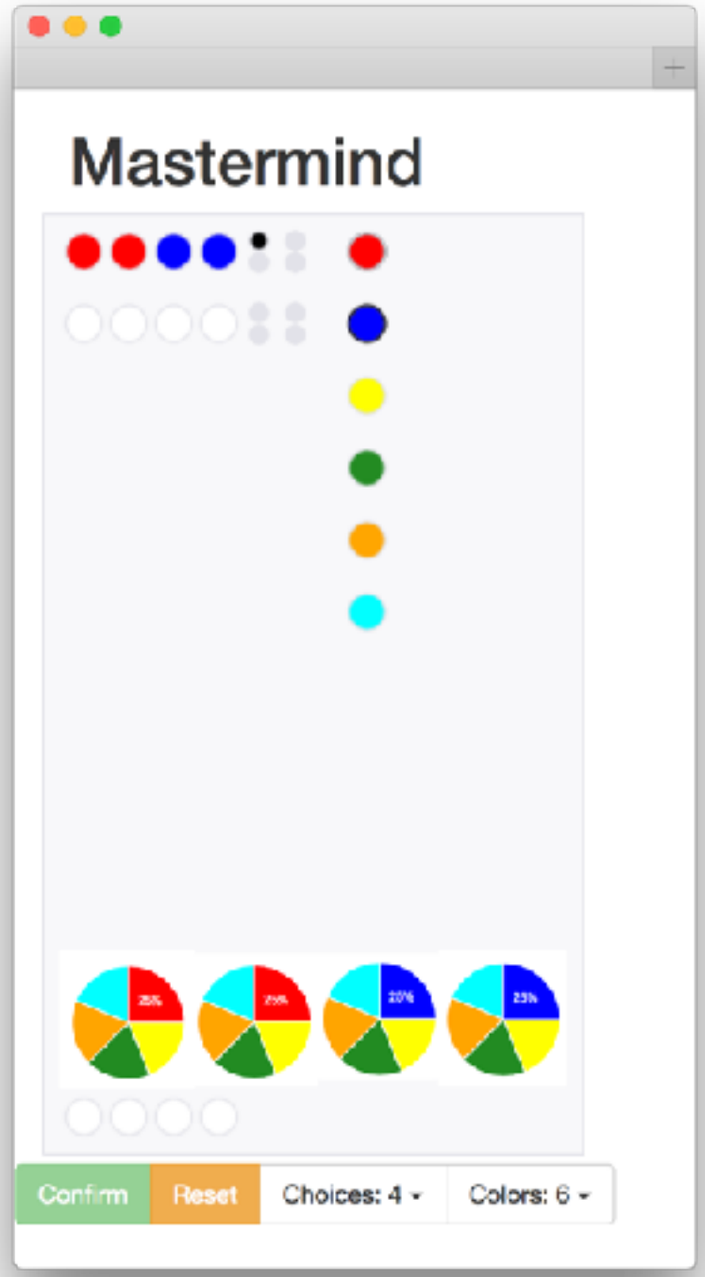
Dynamic ProbLog

Efficient Probabilistic Inference for Dynamic Relational Models

Jonas Vlasselaer, Wannes Meert, Guy Van den Broeck and Luc De Raedt

```
wire(1). wire(2).
wire(3). wire(4).
in(1). in(2). out(4).
gate(a,not,[1],3).
gate(b,and,[3,2],4).

0.990::healthy(G,0) :- gate(G,_,_,_).
0.990::healthy(G,T) :- T>0,    healthy(G,T-1).
0.001::healthy(G,T) :- T>0, \+healthy(G,T-1).

0.5::high(W,T) :- in(W).
0.5::high(W,T) :- gate(G,_,_,W), \+healthy(G,T).
high(W,T) :- gate(G,not,[I],W), healthy(G,T),
             \+high(I,T).
high(W,T) :- gate(G,and,[I,J],W), healthy(G,T),
             high(I,T), high(J,T).

input(W,T) :- in(W), high(W,T).
output(W,T) :- out(W), high(W,T).
```





Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

random variable with Gaussian distribution

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
```

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
    ≃length(OBot) ≥ ≃length(OTop),
    ≃width(OBot) ≥ ≃width(OTop).
```

comparing values of random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,
                            0 : pitcher, 0.8676 : plate,
                            0.0284 : bowl, 0 : serving,
                            0.1016 : none])
                          :- obj(Obj), on(Obj,O2), type(O2,plate).
```

random variable with discrete distribution

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

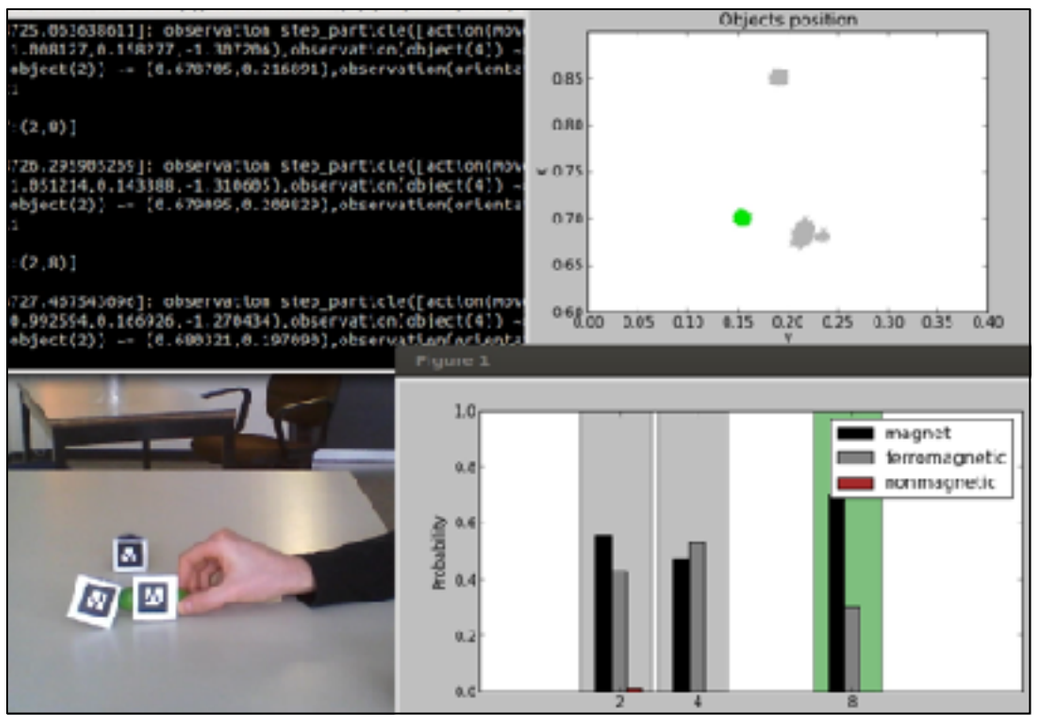
```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,
                            0 : pitcher, 0.8676 : plate,
                            0.0284 : bowl, 0 : serving,
                            0.1016 : none])
                   :- obj(Obj), on(Obj,O2), type(O2,plate).
```

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions

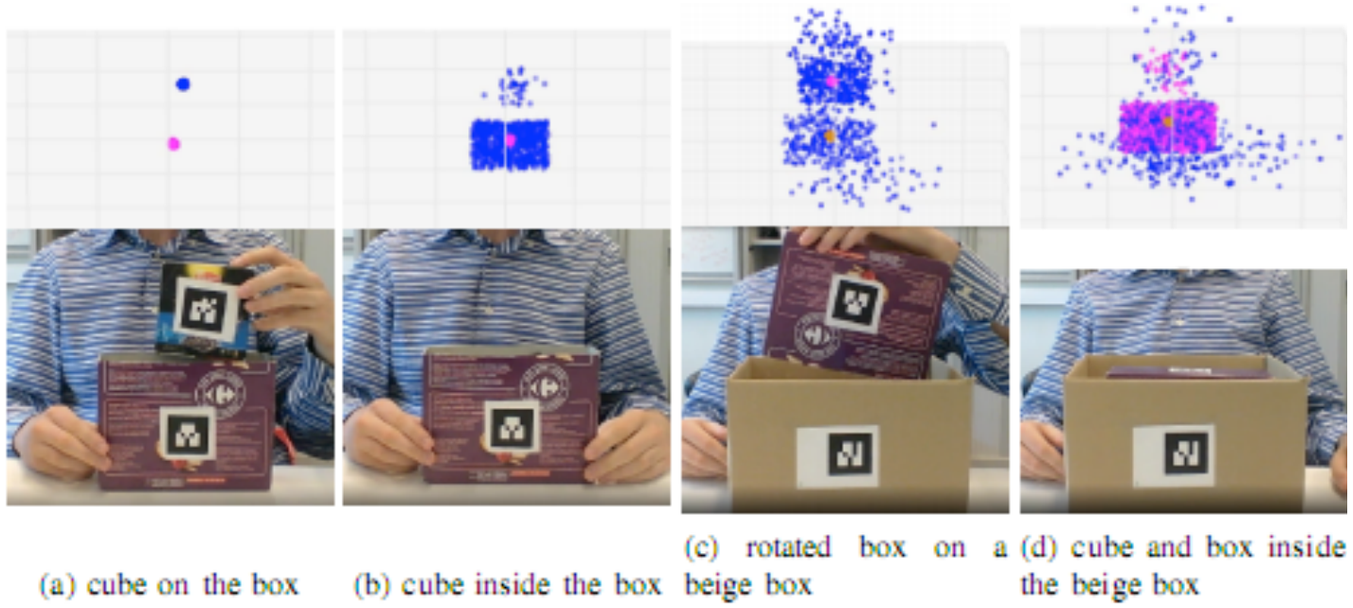


[Nitti et al, IROS 13]

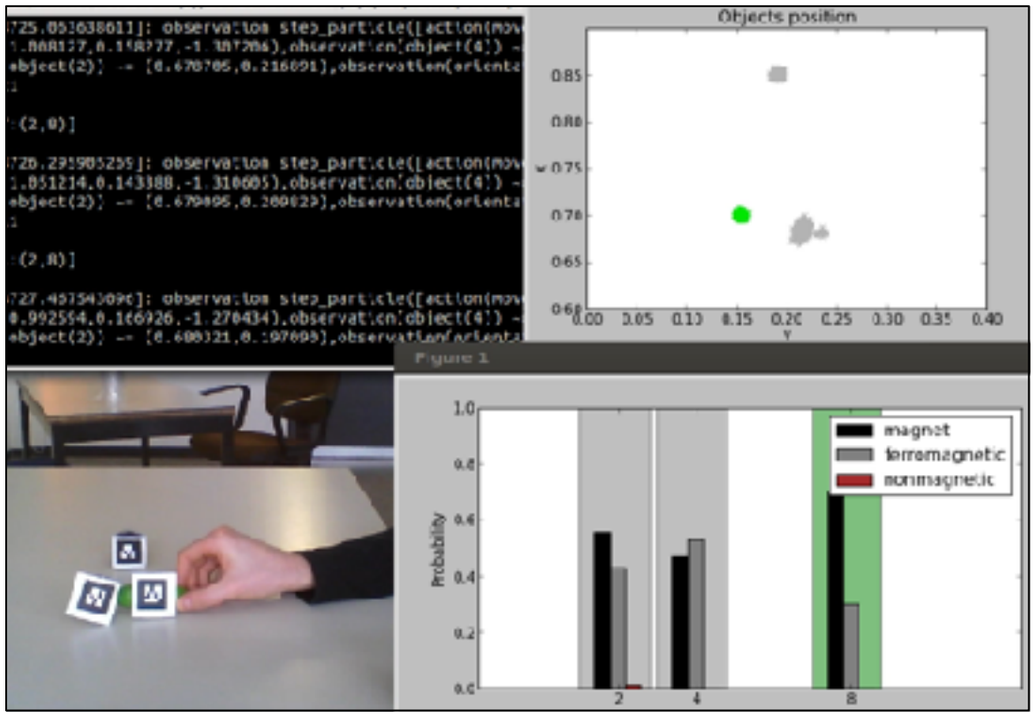
Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions



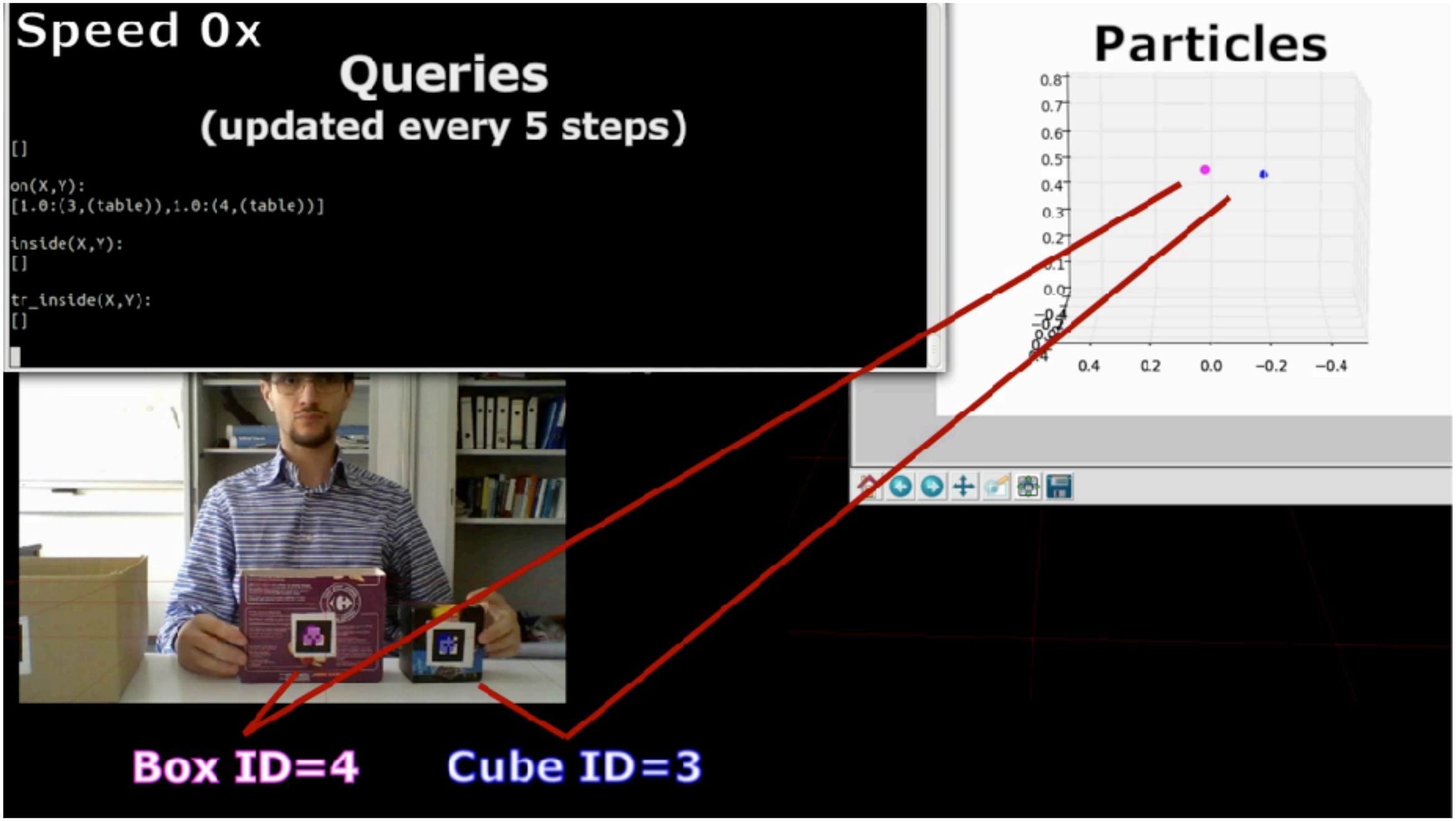
(a) cube on the box (b) cube inside the box (c) rotated box on a beige box (d) cube and box inside the beige box



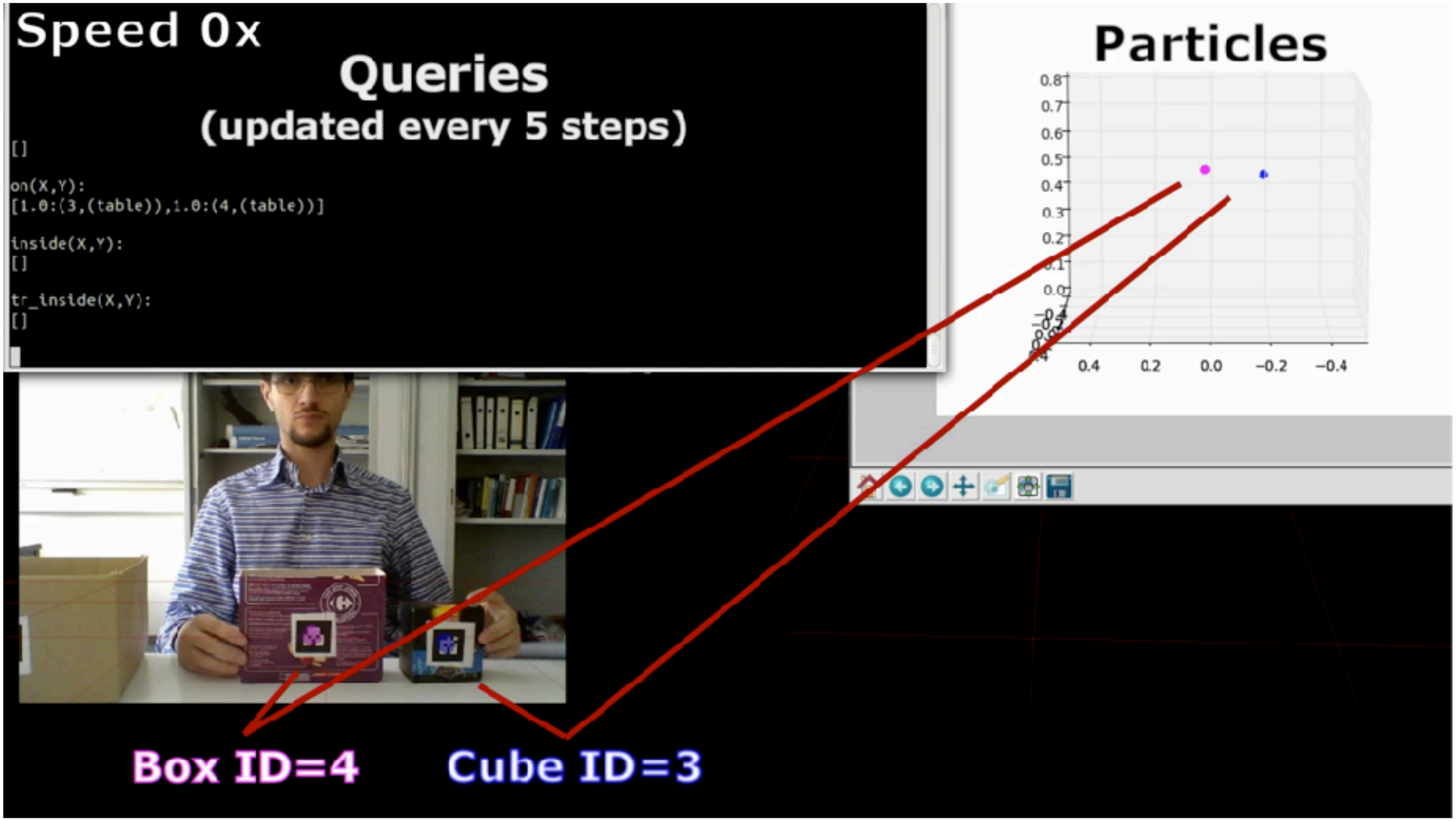
Box scenario

- object tracking even when invisible
- estimate spatial relations

[Nitti et al, IROS 13]


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

IROS 13

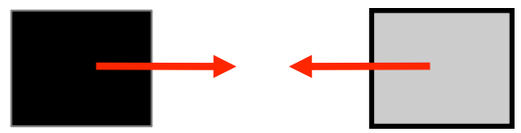

Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

IROS 13

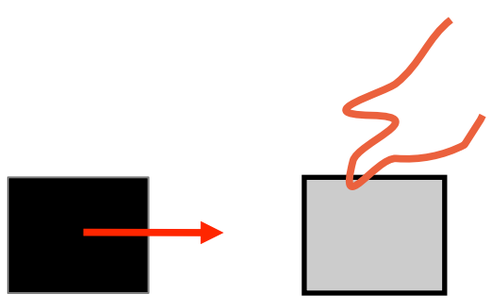
Magnetic scenario

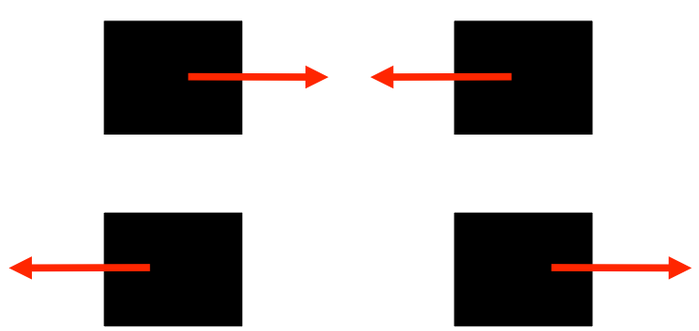
- 3 object types: magnetic, ferromagnetic, nonmagnetic

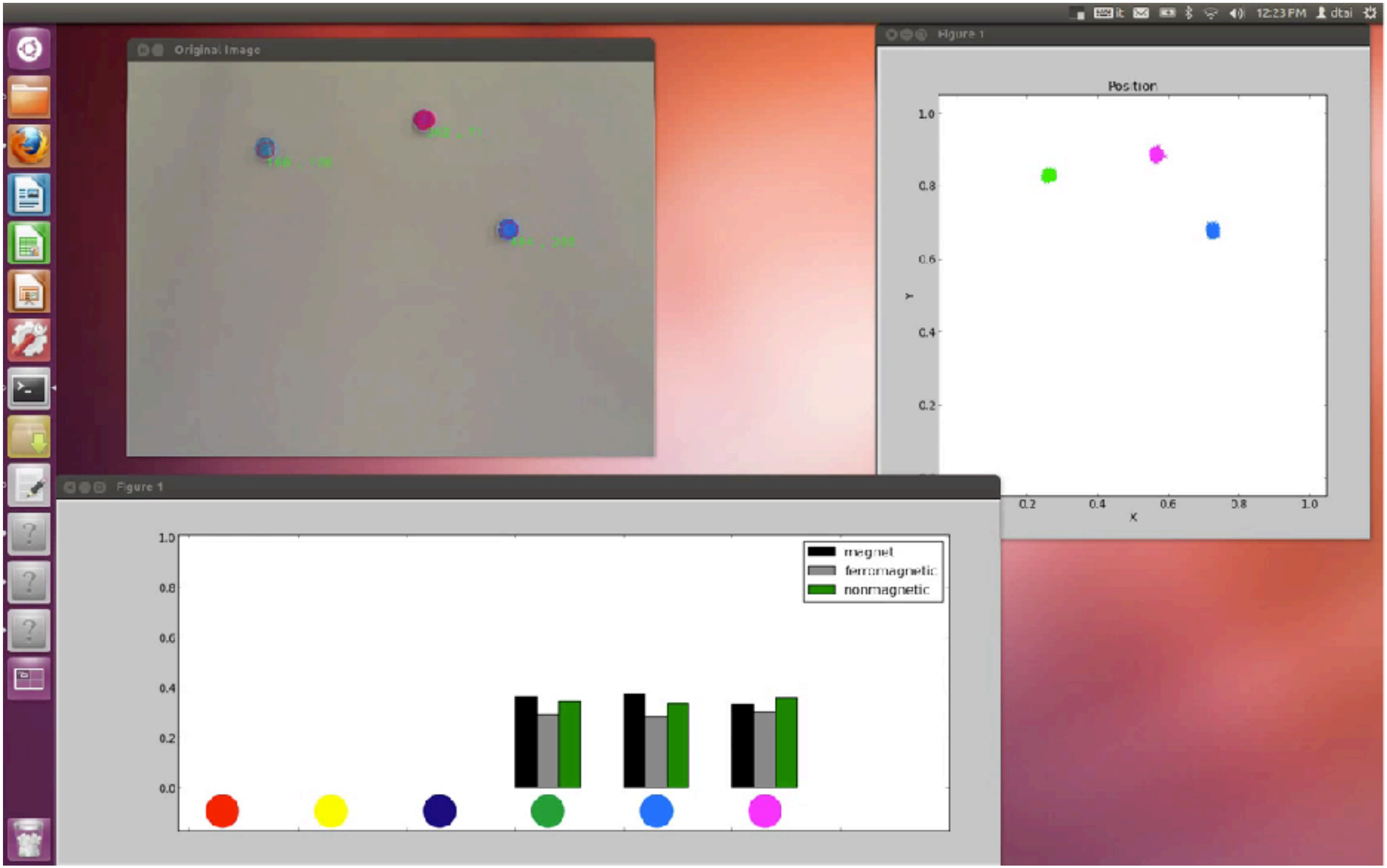
- Nonmagnetic objects do not interact
- A magnet and a ferromagnetic object attract each other

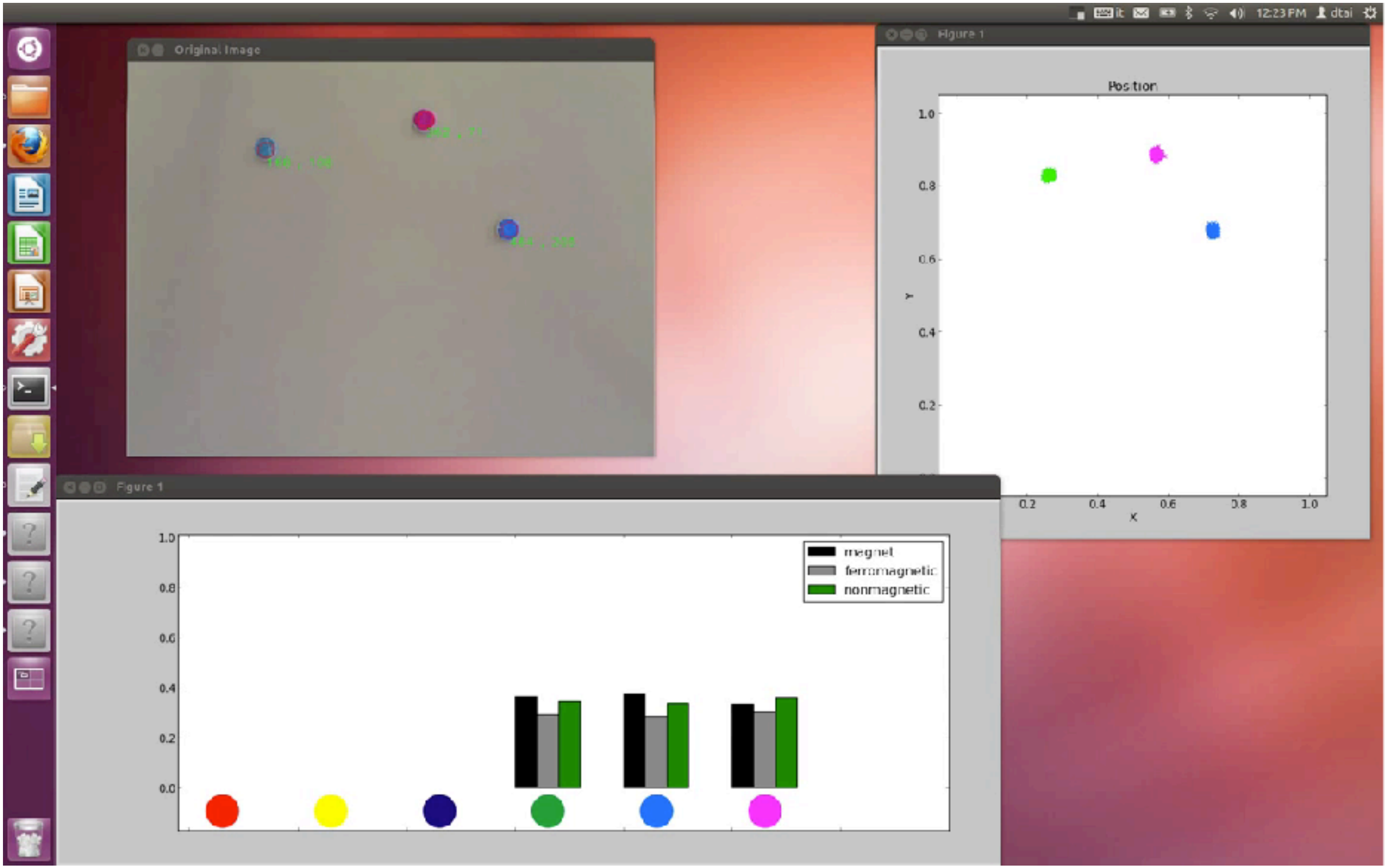


- Magnetic force that depends on the distance
- If an object is held magnetic force is compensated.

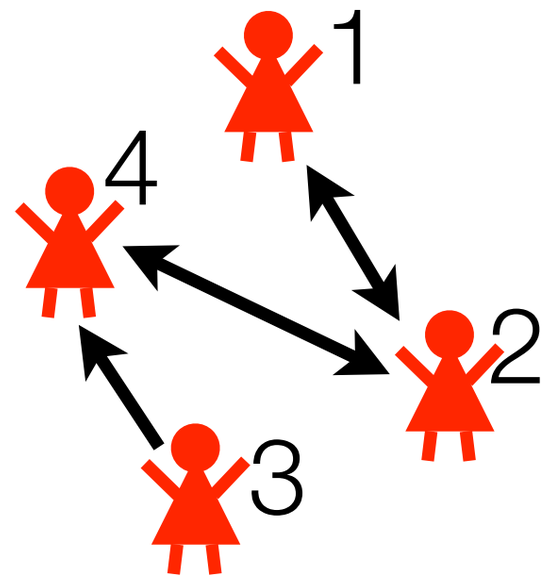





Original Image
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
y
0.2
0.4
0.6
0.8
1.0
x
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
magnet
ferromagnetic
nonmagnetic


Original Image
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
y
0.2
0.4
0.6
0.8
1.0
x
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
magnet
ferromagnetic
nonmagnetic

DTProbLog

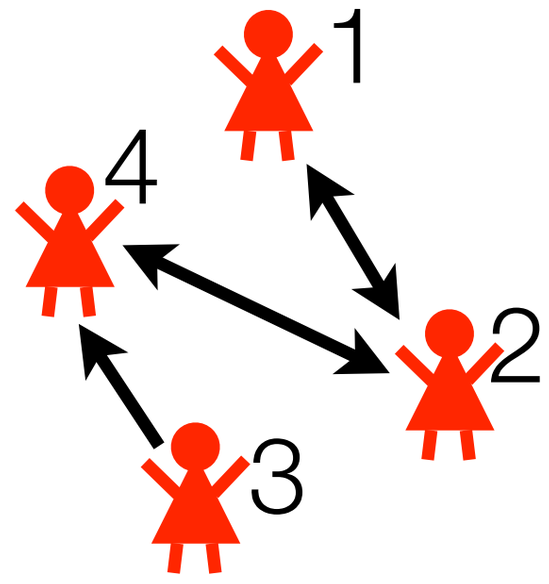



```
person(1).
person(2).
person(3).
person(4).

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

DTProbLog

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

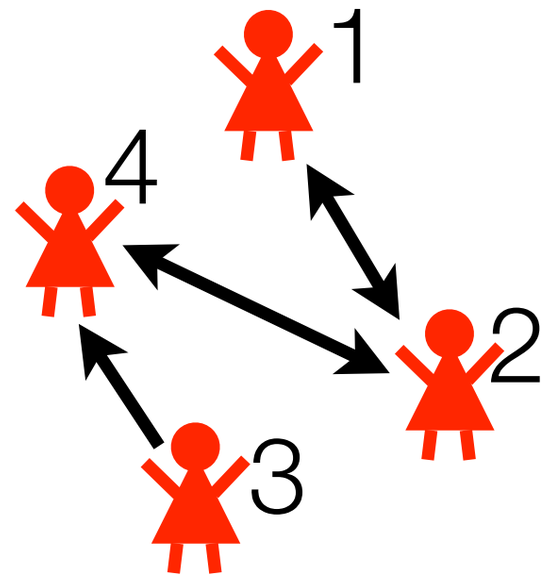
decision fact: true or false?




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

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

DTProbLog




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

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

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

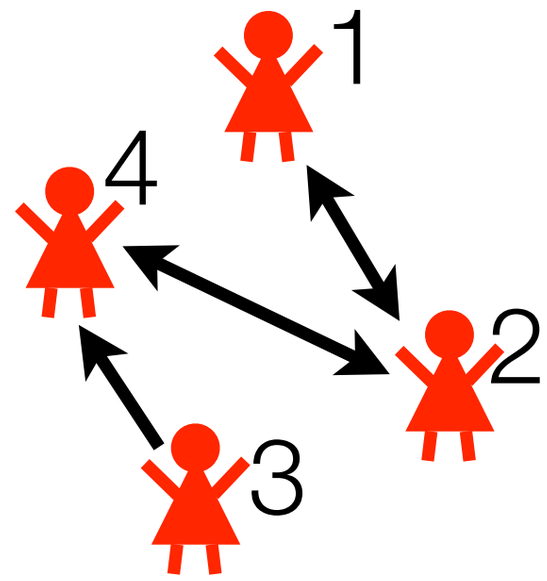
probabilistic facts
+ logical rules

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

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

DTProbLog

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

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

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

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

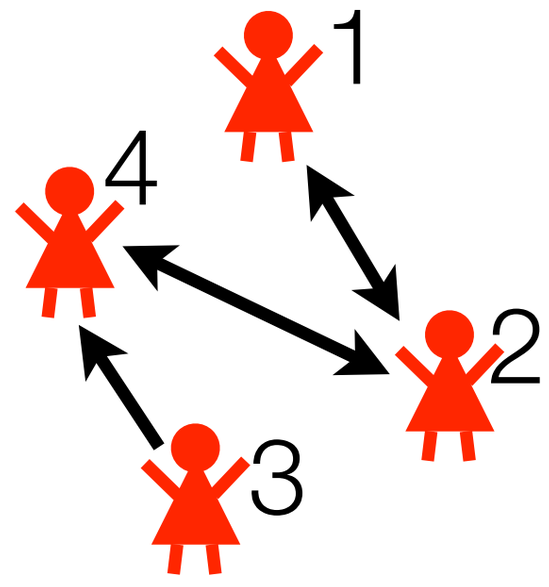
utility facts: cost/reward if true




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

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

DTProbLog

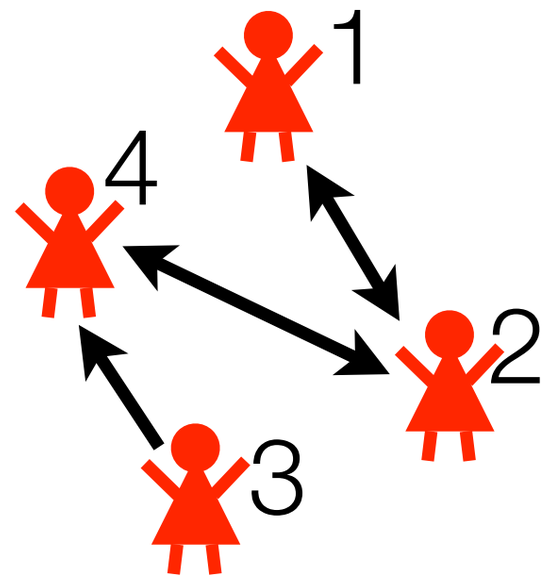
```
? :: marketed(P) :- person(P).

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

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

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




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

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

DTProbLog

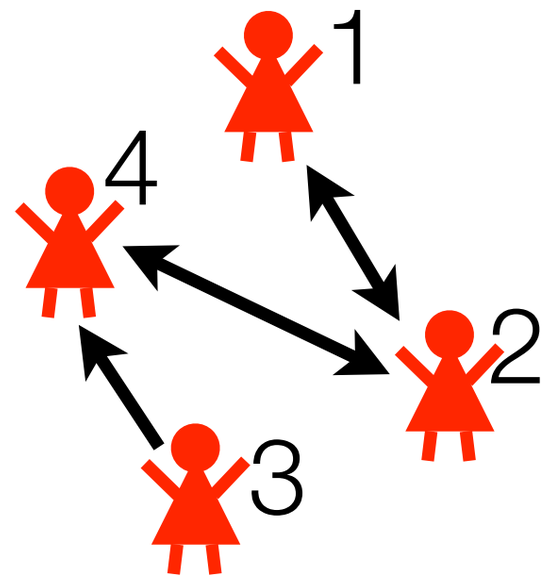
```
? :: marketed(P) :- person(P).

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

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

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




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

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

DTProbLog

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

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

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

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

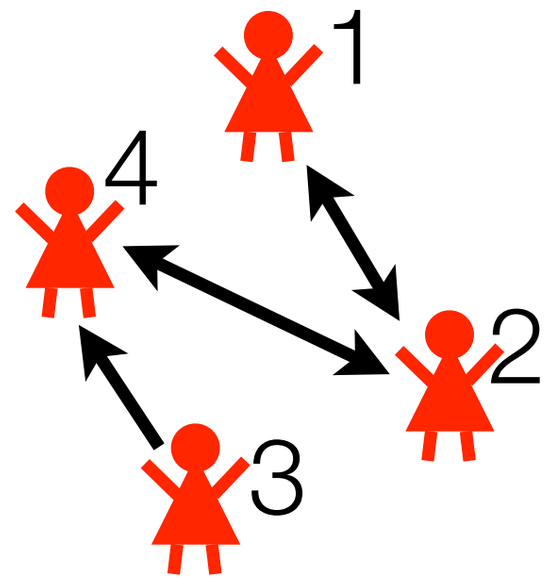



```
person(1).
person(2).
person(3).
person(4).

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

```
marketed(1)        marketed(3)
```

DTProbLog

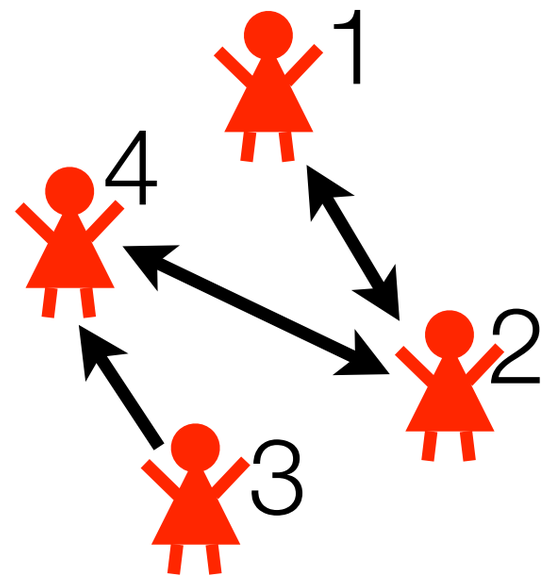
```
? :: marketed(P) :- person(P).

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

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

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




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

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

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)          bm(1)
```

DTProbLog



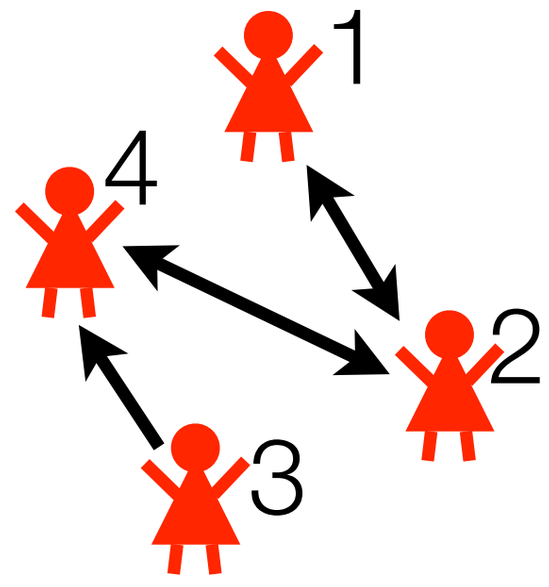

```
? :: marketed(P) :- person(P).

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

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

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

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

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

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)          bm(1)

buys(1)    buys(2)
```

DTProbLog

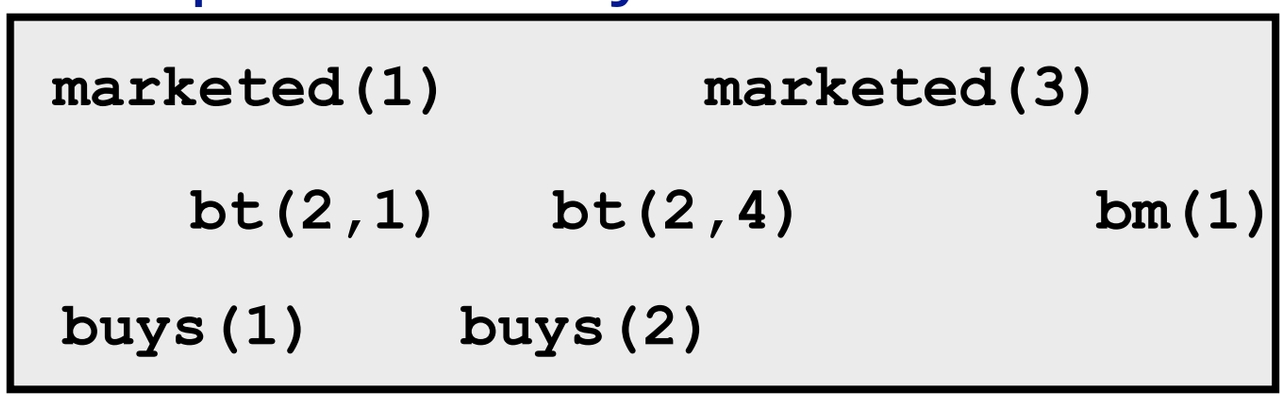
```
? :: marketed(P) :- person(P).

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

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

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




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

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

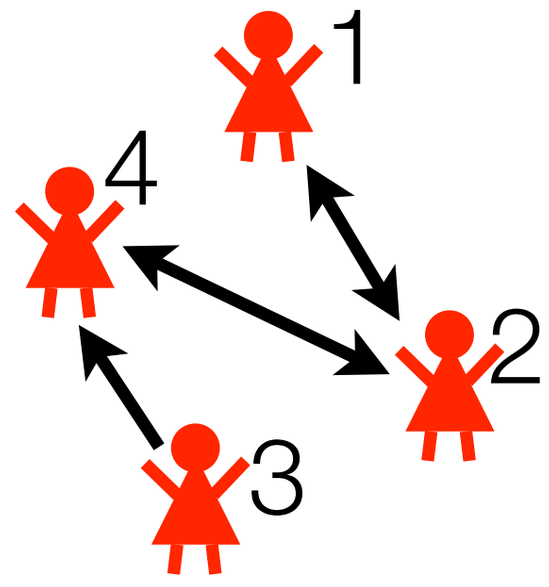
utility = –3 + –3 + 5 + 5 = 4

probability = 0.0032



DTProbLog

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

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

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

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




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

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

utility = –3 + –3 + 5 + 5 = 4

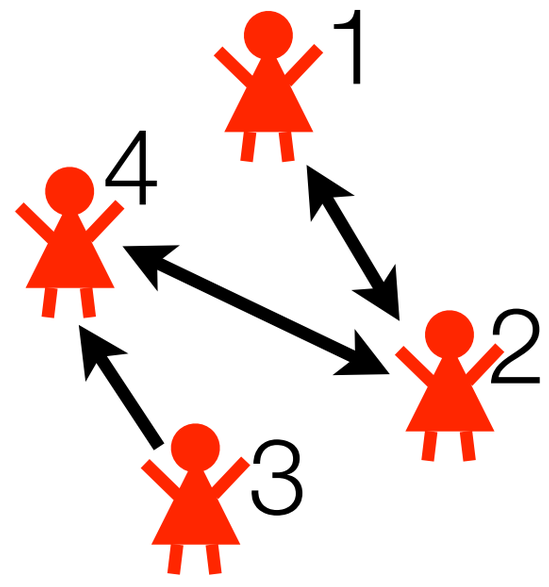
probability = 0.0032

```
marketed(1)        marketed(3)

   bt(2,1)   bt(2,4)          bm(1)

buys(1)    buys(2)
```

world contributes 0.0032×4 to expected utility of strategy

DTProbLog

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

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

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

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




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

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

task: find strategy that maximizes expected utility

solution: using ProbLog technology

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```


[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
```

distribution over **all** possible worlds

```
P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

sbhg e(10)	sb g e(10)	sbh e(10)	sb
s hg e(10)	s g	s h	s
bhg	b g	bh	b
hg	g	h	

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

sbhg e(10)	sb g e(10)	sbh e(10)	sb
s hg e(10)	s g	s h	s
bhg	b g	bh	b
hg	g	h	

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

constraints as first-order logic formulas

sbhg e(10)	sb g e(10)	sbh e(10)	sb
s hg e(10)	s g	s h	s
bhg	b g	bh	b
hg	g	h	

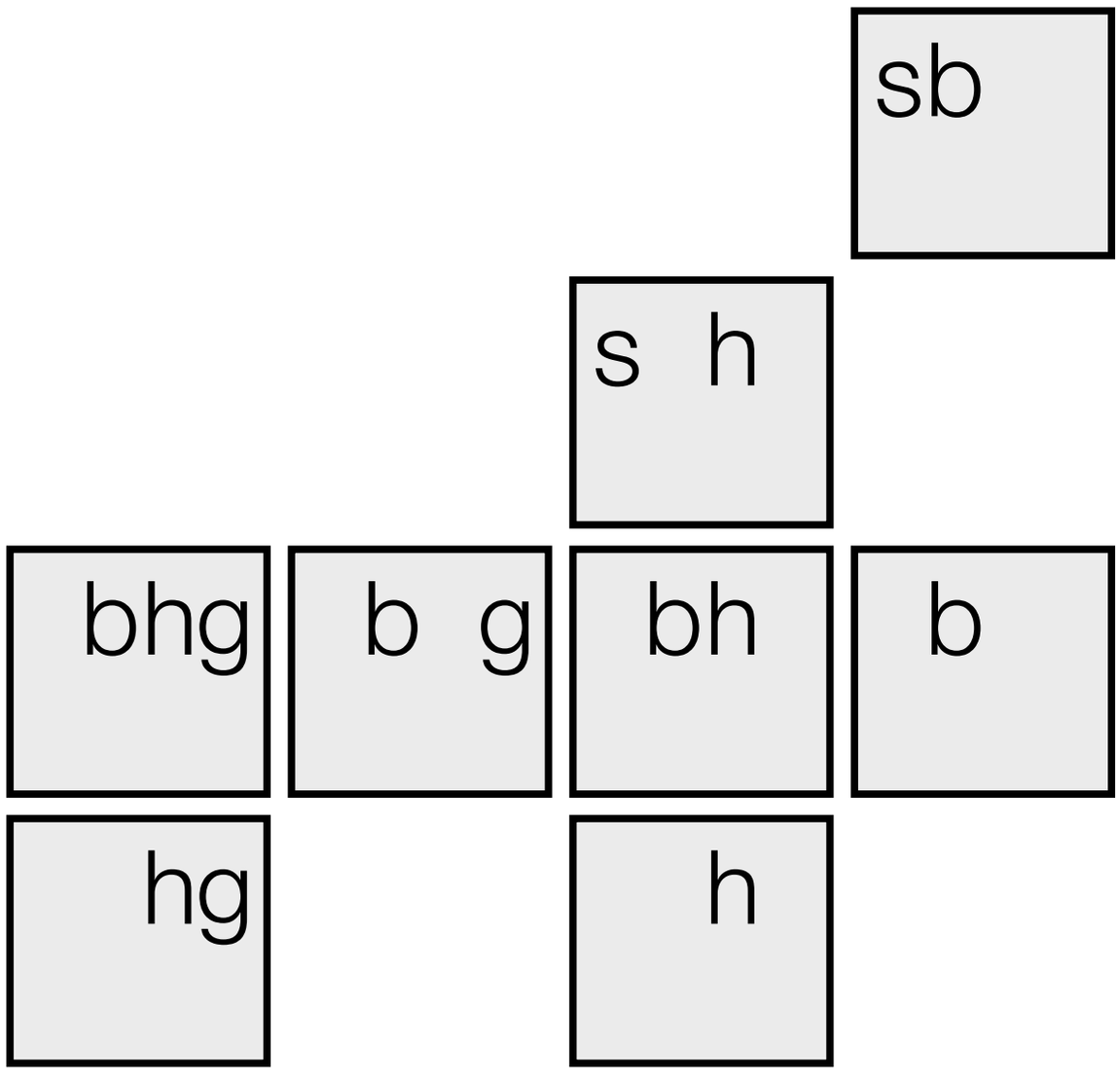
[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...
```

```
not excess(10).
pack(helmet) v pack(boots).
```

constraints as first-order logic formulas



[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

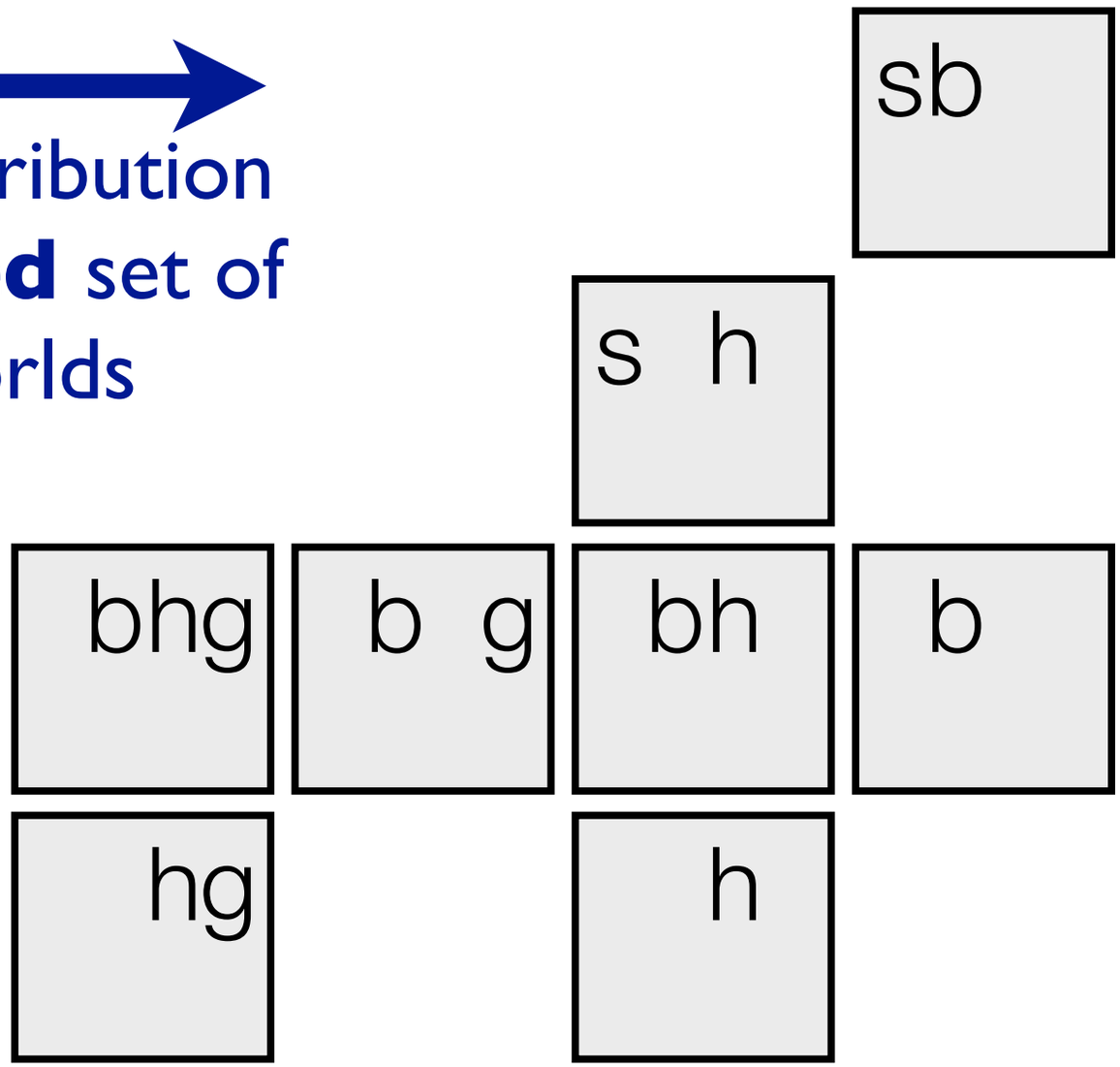
```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
```

```
P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...
```

```
not excess(10).
pack(helmet) v pack(boots).
```

constraints as first-order logic formulas

normalized distribution over **restricted** set of possible worlds



[Fierens et al, PP 12; Shterionov et al]