Representing Large-scale Uncertainty through Probabilistic Databases

Prithviraj Sen

Yahoo! Labs, Bangalore.

International Conference on Management of Data, 2010

(joint work with Profs. Amol Deshpande and Lise Getoor)

Introduction

- Many applications require modeling uncertainty at scale:
  - Information Integration
    - In the absence of primary keys, need to handle potential duplicates.
  - Information Extraction
    - Scraping algorithms often fail.
    - Scale prevents exhaustive manual inspection.
  - Sensor Networks Databases, Mobile Objects Databases
    - Imprecise data, often with confidence bounds.
    - Need to model with statistical models.
  - Social networks, Biological networks.
    - Entity Resolution, Link Prediction etc.
- Need for database systems to model uncertainty for large-scale data.

Motivating Example: Information Integration

Employee DB:

Name	Age	Salary
John Smith	39	$1200
Adam Dole	24	$1250
Maddy Bowen	36	$8700
...	...	...

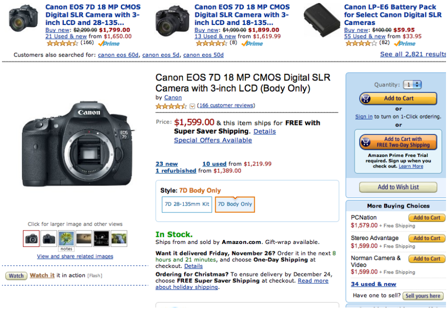
Census DB:

Name	Gender
Johnathan Smith	M
Magdalena Bowen	F
Magda Bowie	F
...	...

↘ ↙

Name	Gender	Age	Salary	
Johnathan Smith	M	39	$1200	0.89
Magdalena Bowen	F	36	$8700	0.95
Magda Bowie	F	36	$8700	0.35
...	...	...	...	...

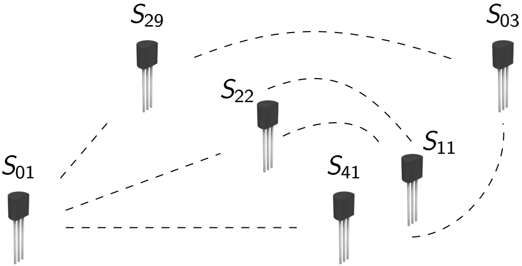
Motivating Example: Information Extraction

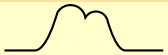
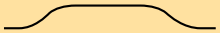


⋮

	Canon EOS7D	$1,507	1/2
		$1,307	1/2
	Canon EF 50	$106.95	2/3
		$7.00	1/3
	Sigma 18-200	$249	1/5
		$200	4/5
	Nikon D5000	$629.95	8/9
		$0.01	1/9

⋮



Sensor	Location	Time	Temperature
S_{11}	32°5′N 67°8′E	11:59pm	
S_{22}	33°8′N 66°6′E	12:06pm	?
S_{29}	34°N 65°8′E	12:10pm	
S_{41}	32°3′N 67°4′E	12:01pm	
⋮	⋮	⋮	⋮

Some History, Why Probabilistic and What's Out There

- Probabilistic databases. Not a recent development.
  - In the 90's, proposals to build databases with IR-style querying.
- Many ways to model uncertainty through databases.
  - Probabilistic databases use probability theory.
  - Because they are powerful enough to represent most applications.
  - While still being (relatively) practical.
- Code is available:
  - SPROUT (from University of Oxford).
  - MystiQ (from University of Washington).
  - Trio (from Stanford).
  - PrDB (soon, from University of Maryland).

Outline

Outline

Semantics of a Probabilistic Database

- A probabilistc database is a distribution over many databases.
- Independent Tuple Uncertain Database
  - Let t denote an uncertain tuple and $pr(t)$ its existence probability.
  - Let $\mathcal{T}$ denote the set of tuples in our probabilistic database.
  - Any $\mathbf{T} \subseteq \mathcal{T}$ is a *possible world*.
  - Probability of possible world $W \in 2^{\mathcal{T}}$ is:

$$Pr(W) \propto \prod_{t \in W} pr(t) \prod_{t \notin W} (1 - pr(t)) \qquad \forall W \in 2^{\mathcal{T}}$$

Example: Semantics of a Probabilistic Database

S

	A	B	
s_1	m	1	0.8
s_2	n	1	0.5

T

	B	C	
t_1	1	p	0.6

possible worlds

instance	probability
$\{s_1, s_2, t_1\}$	0.24
$\{s_1, s_2\}$	0.16*
$\{s_1, t_1\}$	0.24
$\{s_1\}$	0.16
$\{s_2, t_1\}$	0.06
$\{s_2\}$	0.04
$\{t_1\}$	0.06
$\emptyset$	0.04

(Example from Dalvi and Suciu, VLDB'04.)

$^*0.8 \times 0.5 \times (1 - 0.6)$

Query Evaluation

- Every possible world is a “traditional” database.
- Easy to run a query q on W.
- To run query q on a probabilistic database, run q on each W.
- Marginal probability of each result tuple r is:

$$\mu(r) = \sum_{W \in 2^{\mathcal{T}}} pr(W)\delta(r \in q(W))$$

Example: Query Evaluation Semantics

S

	A	B	
s_1	m	1	0.8
s_2	n	1	0.5

T

	B	C	
t_1	1	p	0.6

$$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}) \rightarrow$$

	C
r_1	p

possible worlds		query result
instance	probability	
$\{s_1, s_2, t_1\}$	0.24	$\{r_1\}$
$\{s_1, s_2\}$	0.16	$\emptyset$
$\{s_1, t_1\}$	0.24	$\{r_1\}$
$\{s_1\}$	0.16	$\emptyset$
$\{s_2, t_1\}$	0.06	$\{r_1\}$
$\{s_2\}$	0.04	$\emptyset$
$\{t_1\}$	0.06	$\emptyset$
$\emptyset$	0.04	$\emptyset$

➡ 0.54

Outline

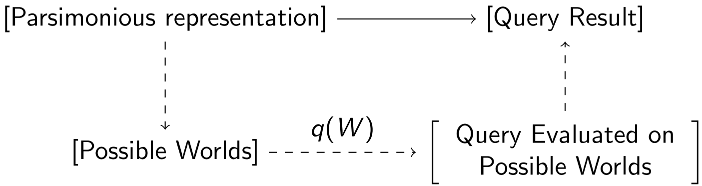
Example: Correlations in a Database

possible worlds	ind.	implies	mutex	nxor	query result
	probability distribution				
$\{s_1, s_2, t_1\}$	0.24	0	0	0.2	$\{r_1\}$
$\{s_1, s_2\}$	0.16	0.33	0.3	0.1	$\emptyset$
$\{s_1, t_1\}$	0.24	0	0	0.2	$\{r_1\}$
$\{s_1\}$	0.16	0.067	0.3	0.1	$\emptyset$
$\{s_2, t_1\}$	0.06	0	0.2	0	$\{r_1\}$
$\{s_2\}$	0.04	0	0	0.2	$\emptyset$
$\{t_1\}$	0.06	0.6	0.2	0	$\emptyset$
$\emptyset$	0.04	0	0	0.2	$\emptyset$
	0.54	0	0.2	0.4	

- *implies*: presence of t_1 implies absence of s_1 and s_2 ($t_1 \Rightarrow \neg s_1 \wedge \neg s_2$).
- *mutual exclusivity* (*mutex*): $t_1 \Rightarrow \neg s_1$ and $s_1 \Rightarrow \neg t_1$.
- *nxor*: high positive correlation between t_1 and s_1, presence (absence) of one almost certainly implies the presence (absence) of the other.

Requirements of a Good Representation

- Should be parsimonious.
  - The set of possible worlds is the power set of a database.
- Independence is not enough, should be able to represent correlations.
- Should be possible to evaluate queries on it.



Outline

Graphical Models and Factored Distributions

- Let X denote a random variable with a fixed-size domain $Dom(X)$.
- Let $pr(X_1, \ldots X_n)$ denote a joint distribution.
- Storing $pr(X_1, \ldots X_n)$ in a table requires $O(|Dom|^n)$ doubles.

Factored Distribution

- Let $\mathbf{X}$ denote a (small) set of random variables.
- Let $f(\mathbf{X})$ denote *factor* such that $0 \leq f(\mathbf{X}) \leq 1$.
- Factored representation:

$$pr(X_1, \ldots X_n) = \frac{1}{Z} \prod_f f(\mathbf{X}_f)$$

where Z denotes the partition function

Example: Linear Chain Bayesian Network

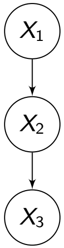
$$pr(X_1 = x_1, X_2 = x_2, X_3 = x_3) = f_1(X_1 = x_1) f_{12}(X_1 = x_1, X_2 = x_2) f_{23}(X_2 = x_2, X_3 = x_3)$$

x_1	f_1
0	0.6
1	0.4

x_1	x_2	f_{12}
0	0	0.9
0	1	0.1
1	0	0.1
1	1	0.9

x_2	x_3	f_{23}
0	0	0.7
0	1	0.3
1	0	0.3
1	1	0.7

x_1	x_2	x_3	Pr
0	0	0	0.378
0	0	1	0.162
0	1	0	0.018
0	1	1	0.042
1	0	0	0.028
1	0	1	0.012
1	1	0	0.108
1	1	1	0.252

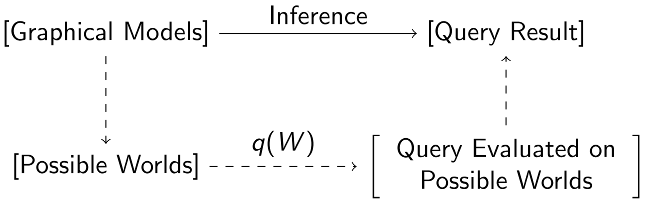


Graphical Models

- Factored representations are parsimonious.
- Graphical representation encodes conditional independencies.
  - e.g., $X_3 \perp X_1 | X_2$ in the previous example.
  - Well known algorithms available (Bayes Ball, D-sep) to read off conditional independence relations from graphical representation.
- Well known flavours: Bayesian networks and Markov networks.
  - Bayesian networks allow directed relationships.
  - Allow non-monotonic reasoning ("explaining away").
  - Factors are called conditional probability tables.
  - Markov networks allow undirected relationships.
  - Factors are called clique potentials.
- More general models include chain graphs and factor graphs.

Benefits of using Graphical Models

- Can represent probabilistic databases parsimoniously.
- Result tuples' probabilities are marginal probability computations.
- Inference algorithms are available.



Outline

Probabilistic Databases and Factors

- Represent correlations with *n*-ary factors.
- For independent tuple databases:
  - Introduce boolean valued random variables for tuples.
  - Use single argument factors to encode tuple probabilities.

$$\forall t : \quad f_t(\mathtt{t}) = pr(t), \quad f_t(\mathtt{f}) = 1 - pr(t)$$

	A	B	
s_1	m	1	0.8

	A	B	
s_2	n	1	0.5

	B	C	
t_1	1	p	0.6

s_1	f_{s_1}
t	0.8
f	0.2

s_2	f_{s_2}
t	0.5
f	0.5

t_1	f_{t_1}
t	0.6
f	0.4

Example: Query Evaluation with Factors

S

	A	B
s_1	m	1
s_2	n	1

f_{s_1}, f_{s_2}

T

	B	C
t_1	1	p

f_{t_1}

$S \bowtie_B T$

$f^{\text{and}}_{i_1,s_1,t_1}, f^{\text{and}}_{i_2,s_2,t_1}$

	A	B	C
i_1	m	1	p
i_2	n	1	p

$\prod_C(S \bowtie_B T)$

	C
r_1	p

$f^{\text{or}}_{r_1,i_1,i_2}$

i_2	s_2	t_1	$f^{\text{and}}_{i_2,s_2,t_1}$
t	t	t	1
t	t	f	0
f	t	f	1
f	t	t	0
⋮	⋮	⋮	⋮

r_1	i_1	i_2	$f^{\text{or}}_{r_1,i_1,i_2}$
t	t	t	1
t	t	f	1
f	t	f	0
f	f	f	1
⋮	⋮	⋮	⋮

Example: Query Evaluation with Factors

S

	A	B
s_1	m	1
s_2	n	1

f_{s_1}, f_{s_2}

T

	B	C
t_1	1	p

f_{t_1}

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

	A	B	C
i_1	m	1	p
i_2	n	1	p

$f^{\text{and}}_{i_1,s_1,t_1}, \; f^{\text{and}}_{i_2,s_2,t_1}$

i_2	s_2	t_1	$f^{\text{and}}_{i_2,s_2,t_1}$
t	t	t	1
t	t	f	0
f	t	f	1
f	t	t	0
⋮	⋮	⋮	⋮

$\Pi_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C
r_1	p

$f^{\text{or}}_{r_1,i_1,i_2}$

r_1	i_1	i_2	$f^{\text{or}}_{r_1,i_1,i_2}$
t	t	t	1
t	t	f	1
f	t	f	0
f	f	f	1
⋮	⋮	⋮	⋮

S

	A	B
s_1	m	1
s_2	n	1

f_{s_1}, f_{s_2}

T

	B	C
t_1	1	p

f_{t_1}

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

$f^{\text{and}}_{i_1,s_1,t_1}$, $f^{\text{and}}_{i_2,s_2,t_1}$

	A	B	C
i_1	m	1	p
i_2	n	1	p

i_1	s_1	t_1	$f^{\text{and}}_{i_1,s_1,t_1}$
t	t	t	1
t	t	f	0
f	t	f	1
f	t	t	0
⋮	⋮	⋮	⋮

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C
r_1	p

$f^{\text{or}}_{r_1,i_1,i_2}$

r_1	i_1	i_2	$f^{\text{or}}_{r_1,i_1,i_2}$
t	t	t	1
t	t	f	1
f	t	f	0
f	f	f	1
⋮	⋮	⋮	⋮

Example: Query Evaluation with Factors

S

	A	B
s_1	m	1
s_2	n	1

f_{s_1}, f_{s_2}

T

	B	C
t_1	1	p

f_{t_1}

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

$f^{\text{and}}_{i_1,s_1,t_1}, \; f^{\text{and}}_{i_2,s_2,t_1}$

	A	B	C
i_1	m	1	p
i_2	n	1	p

i_2	s_2	t_1	$f^{\text{and}}_{i_2,s_2,t_1}$
t	t	t	1
t	t	f	0
f	t	f	1
f	t	t	0
⋮	⋮	⋮	⋮

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C
r_1	p

$f^{\text{or}}_{r_1,i_1,i_2}$

r_1	i_1	i_2	$f^{\text{or}}_{r_1,i_1,i_2}$
t	t	t	1
t	t	f	1
f	t	f	0
f	f	f	1
⋮	⋮	⋮	⋮

Example: Query Evaluation with Factors

S

	A	B
s_1	m	1
s_2	n	1

f_{s_1}, f_{s_2}

T

	B	C
t_1	1	p

f_{t_1}

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$ →

$f^{\text{and}}_{i_1,s_1,t_1}, \ f^{\text{and}}_{i_2,s_2,t_1}$

	A	B	C
i_1	m	1	p
i_2	n	1	p

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$ ↓

	C
r_1	p

$f^{\text{or}}_{r_1,i_1,i_2}$

i_2	s_2	t_1	$f^{\text{and}}_{i_2,s_2,t_1}$
t	t	t	1
t	t	f	0
f	t	f	1
f	t	t	0
⋮	⋮	⋮	⋮

r_1	i_1	i_2	$f^{\text{or}}_{r_1,i_1,i_2}$
t	t	t	1
t	t	f	1
f	t	f	0
f	f	f	1
⋮	⋮	⋮	⋮

Example: Query Evaluation with Factors

S

	A	B
s_1	m	1
s_2	n	1

f_{s_1}, f_{s_2}

T

	B	C
t_1	1	p

f_{t_1}

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$ →

$f^{\text{and}}_{i_1,s_1,t_1}, \; f^{\text{and}}_{i_2,s_2,t_1}$

	A	B	C
i_1	m	1	p
i_2	n	1	p

i_2	s_2	t_1	$f^{\text{and}}_{i_2,s_2,t_1}$
t	t	t	1
t	t	f	0
f	t	f	1
f	t	t	0
⋮	⋮	⋮	⋮

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

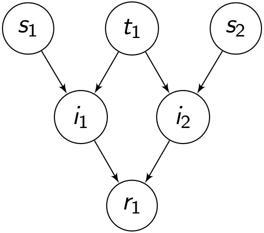
↓

	C
r_1	p

➡ $f^{\text{or}}_{r_1,i_1,i_2}$

r_1	i_1	i_2	$f^{\text{or}}_{r_1,i_1,i_2}$
t	t	t	1
t	t	f	1
f	t	f	0
f	f	f	1
⋮	⋮	⋮	⋮

Inference and Query Evaluation



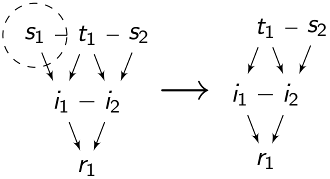
- All factors combined, base and introduced during evaluation, form a graphical model.
- To compute marginal probability of r_1:
  - Multiply all factors.
  - Sum over all random variables except r_1.
- Prior work has used different inference algorithms:
  - variable elimination [SD07]
  - inclusion-exclusion principle [BDHW06, FR97]
  - ordered binary decision diagrams [KO08]
  - Markov Chain Monte Carlo [RDS07, JXWPJH08]
  - ...
- Inference is #P-complete, in general.

Example: Variable Elimination

$$\mu(r_1 = \texttt{t}) = \sum_{i_1,i_2} \sum_{s_1,s_2,t_1} f^{\texttt{or}}_{r_1,i_1,i_2}(r_1 = \texttt{t}, i_1, i_2) f^{\texttt{and}}_{i_2,s_2,t_1}(i_2, s_2, t_1)$$
$$f^{\texttt{and}}_{i_1,s_1,t_1}(i_1, s_1, t_1) f_{t_1}(t_1) f_{s_2}(s_2) f_{s_1}(s_1)$$
$$= \sum_{i_1,i_2} f^{\texttt{or}}_{r_1,i_1,i_2}(r_1 = \texttt{t}, i_1, i_2) \sum_{s_2,t_1} f^{\texttt{and}}_{i_2,s_2,t_1}(i_2, s_2, t_1)$$
$$f_{t_1}(t_1) f_{s_2}(s_2) \underbrace{\sum_{s_1} f^{\texttt{and}}_{i_1,s_1,t_1}(i_1, s_1, t_1) f_{s_1}(s_1)}_{m_{s_1}(i_1,t_1)}$$

$m_{s_1}(i_1, t_1) =$

i_1	t_1	m_{s_1}
f	f	1
t	f	0
f	t	0.2
t	t	0.8



Example: Variable Elimination (contd.)

$$
\begin{aligned}
\mu(r_1 = \mathtt{t}) \\
&= \sum_{i_1,i_2} f^{\mathtt{or}}_{r_1,i_1,i_2}(r_1 = \mathtt{t}, i_1, i_2) \sum_{t_1} m_{s_1}(i_1, t_1) f_{t_1}(t_1) \underbrace{\sum_{s_2} f^{\mathtt{and}}_{i_2,s_2,t_1}(i_2, s_2, t_1) f_{s_2}(s_2)}_{m_{s_2}(i_2,t_1)} \\
&= \sum_{i_1,i_2} f^{\mathtt{or}}_{r_1,i_1,i_2}(r_1 = \mathtt{t}, i_1, i_2) \underbrace{\sum_{t_1} m_{s_1}(i_1, t_1) f_{t_1}(t_1) m_{s_2}(i_2, t_1)}_{m_{t_1}(i_1,i_2)} \\
&= \sum_{i_1} \underbrace{\sum_{i_2} f^{\mathtt{or}}_{r_1,i_1,i_2}(r_1 = \mathtt{t}, i_1, i_2) m_{t_1}(i_1, i_2)}_{m_{i_2}(i_1)} \\
&= \sum_{i_1} m_{i_2}(i_1) \\
&= 0.54
\end{aligned}
$$

Example: Inference with Base Correlations 1

- $(t_1 \Rightarrow \neg s_1 \wedge \neg s_2)$

$$\mu(r_1 = \mathtt{t}) = \sum_{i_1,i_2} f^{\mathtt{or}}_{r_1,i_1,i_2}(r_1 = \mathtt{t}, i_1, i_2) \sum_{s_2,t_1} f^{\mathtt{and}}_{i_2,s_2,t_1}(i_2, s_2, t_1) \sum_{s_1} f^{\mathtt{and}}_{i_1,s_1,t_1}(i_1, s_1, t_1) f^{implies}_{t_1,s_1}(t_1, s_1) f^{implies}_{t_1,s_2}(t_1, s_2) f_{t_1}(t_1)$$

t_1	s_1	$f^{implies}_{t_1,s_1}$
f	f	0
f	t	1
t	f	1
t	t	0

t_1	s_2	$f^{implies}_{t_1,s_2}$
f	f	1/6
f	t	5/6
t	f	1
t	t	0

instance	probability
$\{s_1, s_2, t_1\}$	0
$\{s_1, s_2\}$	0.33
$\{s_1, t_1\}$	0
$\{s_1\}$	0.067
$\{s_2, t_1\}$	0
$\{s_2\}$	0
$\{t_1\}$	0.6
$\emptyset$	0
	0

Example: Inference with Base Correlations 2

- $(t_1 \Rightarrow \neg s_1, s_1 \Rightarrow \neg t_1)$

$$\mu(r_1 = \texttt{t}) = \sum_{i_1,i_2} f^{\texttt{or}}_{r_1,i_1,i_2}(r_1 = \texttt{t}, i_1, i_2) \sum_{s_2,t_1} f^{\texttt{and}}_{i_2,s_2,t_1}(i_2, s_2, t_1) \sum_{s_1} f^{\texttt{and}}_{i_1,s_1,t_1}(i_1, s_1, t_1) f^{mutex}_{t_1,s_1}(t_1, s_1) f_{s_2}(s_2)$$

t_1	s_1	$f^{mutex}_{t_1,s_1}$
f	f	0
f	t	0.6
t	f	0.4
t	t	0

instance	probability
$\{s_1, s_2, t_1\}$	0
$\{s_1, s_2\}$	0.3
$\{s_1, t_1\}$	0
$\{s_1\}$	0.3
$\{s_2, t_1\}$	0.2
$\{s_2\}$	0
$\{t_1\}$	0.2
$\emptyset$	0
	0.2

Example: Inference with Base Correlations 3

- (positive correlation between s_1 and t_1)

$$\mu(r_1 = \mathtt{t}) = \sum_{i_1,i_2} f^{\mathtt{or}}_{r_1,i_1,i_2}(r_1 = \mathtt{t}, i_1, i_2) \sum_{s_2,t_1} f^{\mathtt{and}}_{i_2,s_2,t_1}(i_2, s_2, t_1) \sum_{s_1} f^{\mathtt{and}}_{i_1,s_1,t_1}(i_1, s_1, t_1) f^{nxor}_{t_1,s_1}(t_1, s_1) f_{s_2}(s_2)$$

t_1	s_1	$f^{nxor}_{t_1,s_1}$
f	f	0.4
f	t	0.2
t	f	0
t	t	0.4

instance	probability
$\{s_1, s_2, t_1\}$	0.2
$\{s_1, s_2\}$	0.1
$\{s_1, t_1\}$	0.2
$\{s_1\}$	0.1
$\{s_2, t_1\}$	0
$\{s_2\}$	0.2
$\{t_1\}$	0
$\emptyset$	0.2
	0.4

Outline

Shared Correlations

- Till now, we have been talking about random variables and factors.
- For many applications, this level of detail may be unnecessary.
- Because, uncertainty comes from general statistics, is rarely tuple-specific.

AdID	Make	Color	Price
1	Honda	?	9,000$
2	?	?	6,000$
3	?	Beige	8,000$
⋮	⋮	⋮	⋮

Color	f_{color}
Black	0.75
Beige	0.25

Make	f_{make}
Honda	0.55
Toyota	0.45

Statistical Relational Learning

- Devoted to building large-scale graphical models.
- Use first-order logic (or a suitable subset) to express uncertainty.
- Various approaches: Markov logic networks, probabilistic relational models, Bayesian logic programs, independent choice logic etc.

e.g.: Markov logic networks (`http://alchemy.cs.washington.edu/`)

Friend-of

Name	Friends With
Bob	John
Charlie	Anton
Julie	Cosmo
⋮	⋮

Smokes

Name	Smokes
Bob	?
John	?
Charlie	?
⋮	⋮

$$\forall X, Y, \quad Friend(X, Y) \wedge Smokes(X) \Rightarrow Smokes(Y) \qquad 1.5$$

$$\forall X, \quad Smokes(X) \qquad -1.1$$

Shared Correlations and Query Evaluation

- One approach to inference with shared factors is *propositionalizing*.
- Propositionalizing builds the ground graphical model.
- Flattens out all the shared correlations.
- Second approach is *lifted inference*.
- Attempts to exploit the symmetry in shared correlations.
- Coupled with the fact that shared correlations are introduced during query evaluation too $\Rightarrow$ lifted inference can be much more efficient than propositionalizing.

Outline

Example: Shared Correlations

S	A	B	
s_1	a_1	1	0.8
s_2	a_2	1	0.8
s_3	a_3	1	0.6

T	B	C	
t_1	1	c	0.5

$$S \bowtie_{\mathbf{B}} T$$

Produces 3 result tuples:
$i_j \leftarrow s_j \bowtie t_1, \ \forall j = 1, 2, 3$

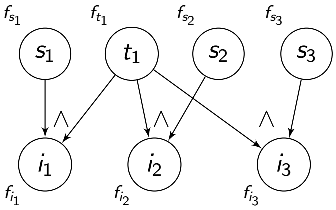
possible world	probability
$\{s_1, s_2, s_3, t_1\}$	0.192
$\{s_1, s_2, s_3\}$	0.192
$\{s_1, s_2, t_1\}$	0.128
$\{s_1, s_2\}$	0.128
$\{s_1, s_3, t_1\}$	0.048
$\{s_1, s_3\}$	0.048
$\{s_1, t_1\}$	0.032
$\{s_1\}$	0.032
$\{s_2, s_3, t_1\}$	0.048
$\{s_2, s_3\}$	0.048
$\{s_2, t_1\}$	0.032
$\{s_2\}$	0.032
$\{s_3, t_1\}$	0.012
$\{s_3\}$	0.012
$\{t_1\}$	0.008
$\emptyset$	0.008

S	A	B	
s_1	a_1	1	0.8
s_2	a_2	1	0.8
s_3	a_3	1	0.6

T	B	C	
t_1	1	c	0.5

$S \bowtie_{\mathbf{B}} T$

$\longrightarrow$



- Inference required:

$$\mu(i_1) = \sum_{s_1, t_1} f_{s_1}(s_1) f_{t_1}(t_1) f_{i_1}^{\text{and}}(i_1, s_1, t_1)$$

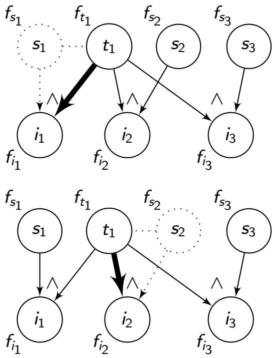
$$\mu(i_2) = \sum_{s_2, t_1} f_{s_2}(s_2) f_{t_1}(t_1) f_{i_2}^{\text{and}}(i_2, s_2, t_1)$$

$$\mu(i_3) = \sum_{s_3, t_1} f_{s_3}(s_3) f_{t_1}(t_1) f_{i_3}^{\text{and}}(i_3, s_3, t_1)$$

Example: Shared Correlations and Inference

$$\mu(i_1) = \sum_{t_1} f_{t_1}(t_1) \underbrace{\sum_{s_1} f_{s_1}(s_1) f_{i_1}^{\text{and}}(i_1, s_1, t_1)}_{m_{s_1}(i_1, t_1)}$$

$$\mu(i_2) = \sum_{t_1} f_{t_1}(t_1) \underbrace{\sum_{s_2} f_{s_2}(s_2) f_{i_2}^{\text{and}}(i_2, s_2, t_1)}_{m_{s_2}(i_2, t_1)}$$



- Two factors f_1 and f_2 are *shared* (or $f_1 \cong f_2$) if they consist of the same input-output mappings.

f	f	1
f	t	0.2
t	f	0
t	t	0.8

S

	A	B	
s_1	m	1	0.8
s_2	n	1	0.8
s_3	o	1	0.6

T

	B	C	
t_1	1	p	0.5

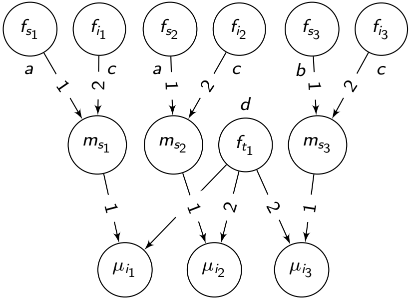
$S \bowtie_{\mathbf{B}} T$

$$\mu_{i_1} = \sum_{t_1} f_{t_1}(t_1) \sum_{s_1} f_{s_1}(s_1) f^{\text{and}}_{i_1}(i_1, s_1, t_1)$$

$$\mu_{i_2} = \sum_{t_1} f_{t_1}(t_1) \sum_{s_2} f_{s_2}(s_2) f^{\text{and}}_{i_2}(i_2, s_2, t_1)$$

$$\mu_{i_3} = \sum_{t_1} f_{t_1}(t_1) \sum_{s_3} f_{s_3}(s_3) f^{\text{and}}_{i_3}(i_3, s_3, t_1)$$

RV-Elim Graph



Shared Factors

- $f_{s_1}(s_1) \cong f_{s_2}(s_2) \ncong f_{s_3}(s_3)$:

s_1	f_{s_1}
t	0.8
f	0.2

s_2	f_{s_2}
t	0.8
f	0.2

s_3	f_{s_3}
t	0.6
f	0.4

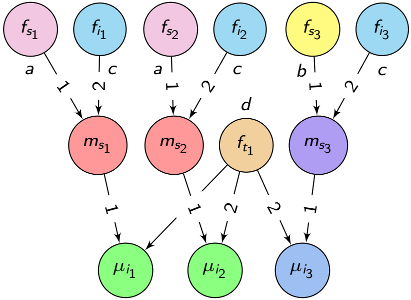
- $m_{s_1}(i_1, t_1) \cong m_{s_2}(i_2, t_1)$:

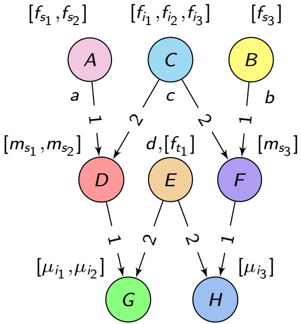
i_1	t_1	m_{s_1}
t	t	0.8
t	f	0
f	t	0.2
f	f	1

i_2	t_1	m_{s_2}
t	t	0.8
t	f	0
f	t	0.2
f	f	1

Compressing RV-Elim Graphs

- $f_1 \cong f_2$ if parents are shared, and labels match.

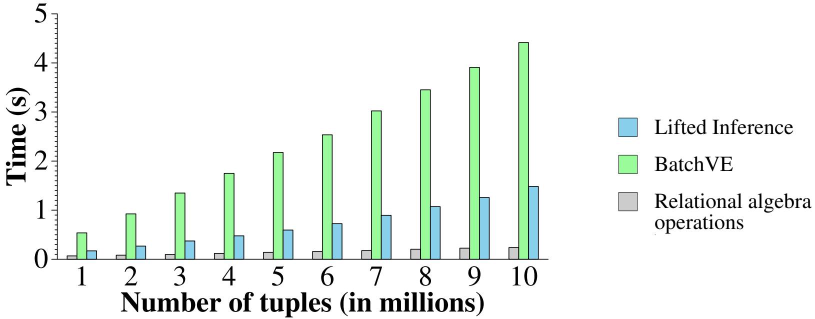




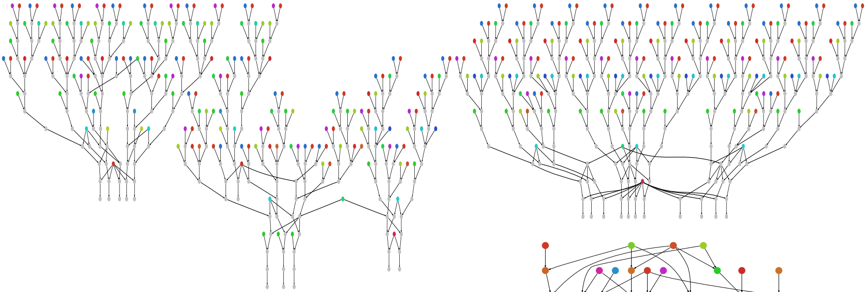
Details

- Final inference algorithm is a three-stage approach:
  1. Detect shared factors in the rv-elim graph.
  2. Run inference on the compressed rv-elim graph.
  3. Retrieve relevant marginals.
- Computing "$\cong$" is closely related to *bisimulation* [KS83].
- RV-Elim graphs are DAGs.
- Fast bisimulation algorithms available for DAGs [DPP01].
- Our algorithm runs in $O(|E| \log D + |V|)$ time.

Lifted Inference: Scalability



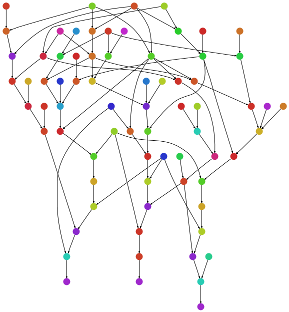
Sample RV-Elim graphs



↑

Original rv-elim graph

Compressed rv-elim graph →



Outline

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$S \bowtie_B T$

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_C (S \bowtie_B T)$

	C	
r_1	p	$s_1 t_1 + s_2 t_1$

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_{C}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C
r_1	p

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_{C}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C
r_1	p

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$ →

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$ ↓

	C	
r_1	p	$s_1 t_1 + s_2 t_1$

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C	
r_1	p	$s_1 t_1 + s_2 t_1$

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$ →

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$ ↓

	C	
r_1	p	$s_1 t_1 + s_2 t_1$

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Example: Boolean Formulas

S

	A	B	
s_1	m	1	s_1
s_2	n	1	s_2

T

	B	C	
t_1	1	p	t_1

$\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T}$

	A	B	C	
i_1	m	1	p	$s_1 t_1$
i_2	n	1	p	$s_2 t_1$

$\prod_{\mathbf{C}}(\mathbf{S} \bowtie_{\mathbf{B}} \mathbf{T})$

	C	
r_1	p	$s_1 t_1 + s_2 t_1$

- Boolean formulas are restricted graphical models.
- For querying independent tuples, boolean formulas suffice.

Hierarchical Queries

- r_1's boolean formula has a special property:

$$s_1 t_1 + s_2 t_1 = t_1(s_1 + s_2)$$

- Easy to compute marginal probabilities from factorized formulas.
- *Hierarchical queries* [DS04] always give factorized formulas.
- Form a well defined subclass of relational algebra.

Definition of Hierarchical Queries

- Let subgoals of an attribute denote the relations it is present in.

$$q(\mathbf{C}) :- \mathbf{S}(\mathbf{A}, \mathbf{B}), \mathbf{T}(\mathbf{B}, \mathbf{C})$$
$$\text{sg}(\mathbf{A}) = \{\mathbf{S}\}$$
$$\text{sg}(\mathbf{B}) = \{\mathbf{S}, \mathbf{T}\}$$

- Hierarchical query: For any two attributes a, b
  - $\text{sg}(a) \subseteq \text{sg}(b)$ or
  - $\text{sg}(a) \supseteq \text{sg}(b)$ or
  - $\text{sg}(a) \cap \text{sg}(b) = \emptyset$

- In the previous example: $\text{sg}(\mathbf{A}) = \{\mathbf{S}\} \subset \{\mathbf{S}, \mathbf{T}\} = \text{sg}(\mathbf{B})$

A non-hierarchical query

- Non-hierarchical query:

$$q() :- \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

- Because:

$$\text{sg}(\mathbf{X}) = \{\mathcal{X}, \mathcal{Z}\}$$
$$\text{sg}(\mathbf{Y}) = \{\mathcal{Z}, \mathcal{Y}\}$$

- Therefore:

$$\text{sg}(\mathbf{X}) \not\subseteq \not\supseteq \text{sg}(\mathbf{Y})$$
$$\text{sg}(\mathbf{Y}) \cap \text{sg}(\mathbf{X}) = \{\mathcal{Y}\}$$

- Well known hard query, can be used to count satisfying assignments of any 2-DNF [DS04].

Drawbacks of Hierarchical Queries

- Does not consider the database.
- Originally defined for conjunctive queries, no self-joins.
- Original formulation was strictly meant for equality predicates only.
- Later, extensions for inequality predicates [OH08, OH09], self-joins [DSS10].

Example(s)

$$q() :- \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

Example(s)

$q() :- \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$

$\mathcal{X}$:

	X
x_1	x_1
x_2	x_2

$\mathcal{Z}$:

	X	**Y**
z_1	x_1	y_1
z_2	x_1	y_2
z_3	x_2	y_3
z_4	x_2	y_4

$\mathcal{Y}$:

	Y
y_1	y_1
y_2	y_2
y_3	y_3
y_4	y_4

$$r = x_1 z_1 y_1 + x_1 z_2 y_2 + x_2 z_3 y_3 + x_2 z_4 y_4$$

$$= x_1(z_1 y_1 + z_2 y_2) + x_2(z_3 y_3 + z_4 y_4)$$

Example(s)

$$q() \coloneq \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

$\mathcal{X}$:

	X
x_1	x_1
x_2	x_2

$\mathcal{Z}$:

	X	Y
z_1	x_1	y_1
z_2	x_1	y_2
z_3	x_2	y_3
z_4	x_2	y_4

$\mathcal{Y}$:

	Y
y_1	y_1
y_2	y_2
y_3	y_3
y_4	y_4

$$r = x_1 z_1 y_1 + x_1 z_2 y_2 + x_2 z_3 y_3 + x_2 z_4 y_4$$

$$= x_1(z_1 y_1 + z_2 y_2) + x_2(z_3 y_3 + z_4 y_4)$$

Example(s)

$$q() :- \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

$\mathcal{X}$:

	X
x_1	x_1
x_2	x_2

$\mathcal{Z}$:

	X	**Y**
z_1	x_1	y_1
z_2	x_1	y_2
z_3	x_2	y_2

$\mathcal{Y}$:

	Y
y_1	y_1
y_2	y_2

$$r = x_1 z_1 y_1 + x_1 z_2 y_2 + x_2 z_3 y_2$$

$=$ Not factorizable

Example(s)

$$q() :- \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

$\mathcal{X}$:

	X
x_1	x_1
x_2	x_2

$\mathcal{Z}$:

	X	Y
z_1	x_1	y_1
z_2	x_1	y_2
z_3	x_2	y_2

$\mathcal{Y}$:

	Y
y_1	y_1
y_2	y_2

$$r = x_1 z_1 y_1 + x_1 z_2 y_2 + x_2 z_3 y_2$$

$$= \text{Not factorizable}$$

Example(s)

$\mathcal{X}$:

	$\mathbf{X}$
x_1	x_1
x_2	x_2
x_3	x_3

$\mathcal{Z}$:

	$\mathbf{X}$	$\mathbf{Y}$
z_1	x_1	y_1
z_2	x_1	y_2
z_3	x_2	y_3
z_4	x_3	y_3

$\mathcal{Y}$:

	$\mathbf{Y}$
y_1	y_1
y_2	y_2
y_3	y_3

$$q() \coloneq \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

$$r = x_1 z_1 y_1 + x_1 z_2 y_2 + x_2 z_3 y_3 + x_3 z_4 y_3$$

$$= x_1(z_1 y_1 + z_2 y_2) + y_3(x_2 z_3 + x_3 z_4)$$

Example(s)

$$q() \coloneq \mathcal{X}(\mathbf{X}), \mathcal{Z}(\mathbf{X}, \mathbf{Y}), \mathcal{Y}(\mathbf{Y})$$

$\mathcal{X}$:

	X
x_1	x_1
x_2	x_2
x_3	x_3

$\mathcal{Z}$:

	X	**Y**
z_1	x_1	y_1
z_2	x_1	y_2
z_3	x_2	y_3
z_4	x_3	y_3

$\mathcal{Y}$:

	Y
y_1	y_1
y_2	y_2
y_3	y_3

$$r = x_1 z_1 y_1 + x_1 z_2 y_2 + x_2 z_3 y_3 + x_3 z_4 y_3$$

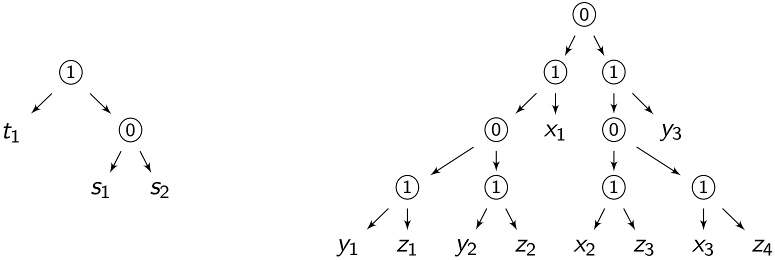
$$= x_1(z_1 y_1 + z_2 y_2) + y_3(x_2 z_3 + x_3 z_4)$$

Query Evaluation with Factorized Formulas

- Hierarchical queries are great.
- Even better: involve the database while deciding tractability.
- One step further: query evaluation with factorized formulas.
- Algorithms to determine factorizability are available.
- However, these are expensive.
- Possible to factorize faster for conjunctive queries without self-joins.
  - No restrictions on join predicates.

Read-once functions [GMR06]

- Factorized form: Each variable appears at most once.
- Factorizable boolean formulas are also known as *read-once functions*.
- The factorized form of a formula, is called its *read-once expression*.
- Read-once expressions are traditionally represented using *co-trees*.



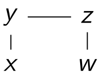
$$Pr(v) = \begin{cases} \prod_{c \in ch(v)} Pr(c) & \text{if } v \text{ is } ① \\ 1 - \prod_{c \in ch(v)} (1 - Pr(c)) & \text{if } v \text{ is } ⓪ \\ pr(v) & \text{if } v \in R \end{cases}$$

Three Properties of Read-Once Functions

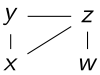
- [Unateness] No variable appears in both positive and negated forms

xy is unate — $\bar{x}y + \bar{x}z$ is unate — $\bar{x}y + xz$ is **not** unate

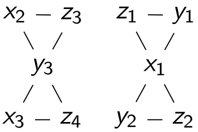
- [P_4-free] Co-occurrence graph should be P_4-free



$xy + yz + zw$ has a P_4

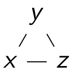


$z(xy + w)$ is P_4-free



- [Normality] Each clique should be contained in some clause

xyz is normal — $xy + yz + xz$ is **not** normal



Limitations of factorization algorithms [GMR06]

- Given ϕ, let $G_\phi = (V, E)$ denote its co-occurrence graph

$$\begin{aligned}\text{Time complexity} &= \text{Unateness} + P_4\text{-free} + \text{Normality} \\ &= O(|\phi|) + O(|V| + |E|) + O(|\phi||V|)\end{aligned}$$

- Normality check is expensive
- P_4-check requires DNF or co-occurrence graph
- Conversion to DNF may require $O(n^k)$ operations, where n is #tuples and k is #joins.

Our goals:

- **Avoid** performing expensive checks
- **Avoid** building co-occurrence graph or the DNF

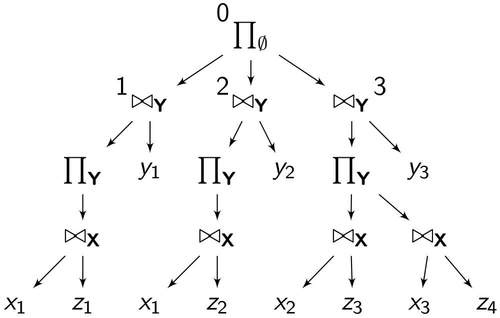
Is possible for conjunctive queries without self-joins.

P_4-checking without DNFs

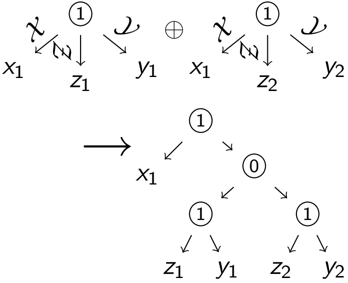
2-phase approach to factorizing:

- 1^{st} phase builds lineage-trees for result tuples.
- 2^{nd} phase recursively builds factorized expression from lineage-tree.
  - 2^{nd} phase uses a tree alignment operator $\oplus$.
  - Conceptually, $T_1 \oplus T_2$ computes $\phi(T_1) \vee \phi(T_2)$.

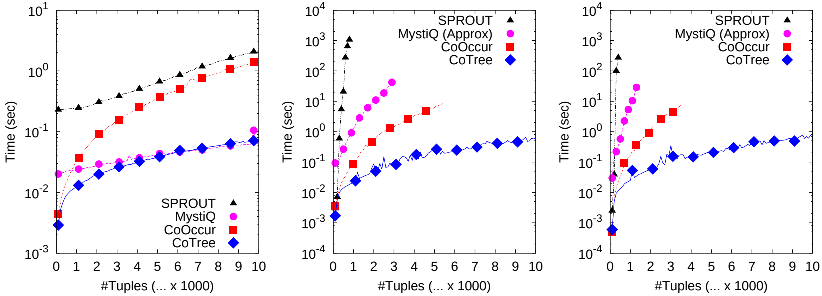
Example: Building Co-Trees



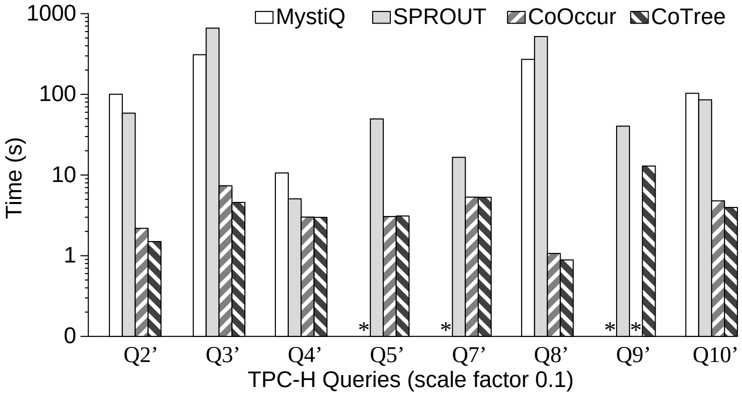
$$\begin{aligned} T_0 &= T_1 \oplus T_2 \oplus T_3 \\ T_3 &= T[\bowtie(\pi(\bowtie(x_2, z_3), \\ &\qquad \bowtie(x_3, z_4)), y_3)] \\ &= ①(⓪(①(x_2, z_3), \\ &\qquad ①(x_3, z_4)), y_3) \end{aligned}$$



Experiments: Synthetic data



Experiments: TPC-H



Outline

Summary

- Lots of people have done lots of very diverse work in this field.
- Alternate representations:
  - x-tuples (Trio)
  - world set decomposition (SPROUT/MayBMS)
  - block independent disjoint (MystiQ)
  - conditional random fields (BayesStore)
  - And/Or trees
  - more?
- Query evaluation:
  - Inequality Predicates
  - Queries with Self-Joins
  - Approximate Query Evaluation
  - Inference based on Improved Sampling
  - Indexing for large Junction Trees
- Each has its own pros and cons.
- Lots of open questions.

Topics Not Discussed

- Ranking Queries.
- Continuous-valued Attributes.
- Ranking over Continuous-valued Attributes.
- Time-varying attributes.
- Query Languages based on Secord-order Logic.
- Mobile Object Databases.
- Privacy and Security.
- Improving the Quality of a Probabilistic Database.
- ...

Outline

References

[BDHW06] Omar Benjelloun, Anish Das Sarma, Alon Halevy and Jennifer Widom.
ULDBs: Databases with Uncertainty and Lineage.
In *VLDB*, 2006.

[DSS10] Nilesh N. Dalvi, Karl Schnaitter and Dan Suciu
Computing query probability with incidence algebras.
In *PODS*, 2010.

[DS04] Nilesh Dalvi and Dan Suciu.
Efficient Query Evaluation on Probabilistic Databases.
In *VLDB*, 2004.

[DGS08] Amol Deshpande, Lise Getoor and Prithviraj Sen.
Graphical Models for Uncertain Data.
In *Managing and Mining Uncertain Data*, Charu Aggarwal (ed.), Springer, 2008.

[DPP01] Agostino Dovier, Carla Piazza and Alberto Policriti.
A Fast Bisimulation Algorithm.
In *International Conference on Computer Aided Verification*, 2001.

References

[FR97] Norbert Fuhr and Thomas Rolleke.
A probabilistic relational algebra for the integration of information retrieval and database systems.
In *Transactions on Information Systems*, 1997.

[GFKT02] L. Getoor, N. Friedman, D. Koller and B. Taskar.
Learning probabilistic models with link uncertainty.
In *JMLR*, 2002.

[GMR06] M. Golumbic, A. Mintz and U. Rotics.
Factoring and Recognition of Read-Once Functions Using Cographs and Normality and the Readability of Functions Associated with Partial k-Trees.
In *Discrete Applied Mathematics*, 2006.

[JXWPJH08] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher M. Jermaine and Peter J. Haas.
MCDB: A monte carlo approach to managing uncertain data.
In *SIGMOD*, 2008.

[KS83] Paris Kanellakis and Scott Smolka.
CCS expressions, finite state processes, and three problems of equivalence.
In *PODC*, 1983.

References

[KO08] Christoph Koch and Dan Olteanu.
Conditioning probabilistic databases.
In *VLDB*, 2008.

[OH09] Dan Olteanu and Jiewen Huang.
Secondary-storage confidence computation for conjunctive queries with inequalities.
In *SIGMOD*, 2009.

[OH08] Dan Olteanu and Jiewen Huang.
Using OBDDs for Efficient Query Evaluation on Probabilistic Databases.
In *SUM*, 2008.

[Poole03] David Poole.
First-order probabilistic inference.
In *IJCAI*, 2003.

[RDS07] Christopher Re, Nilesh Dalvi and Dan Suciu.
Efficient Top-k Query Evaluation on Probabilistic Data.
In *ICDE*, 2007.

[RD06] Matthew Richardson and Pedro Domingos.
Markov Logic Networks.
In *Machine Learning*, 2006.

References

[SD07] Prithviraj Sen and Amol Deshpande.
Representing and Querying Correlated Tuples in Probabilistic Databases.
In *ICDE*, 2007.

[SDG08] Prithviraj Sen, Amol Deshpande and Lise Getoor.
Exploiting Shared Correlations in Probabilistic Databases.
In *VLDB*, 2008.

[SDG09] Prithviraj Sen, Amol Deshpande and Lise Getoor.
Bisimulation-based Approximate Lifted Inference.
In *UAI*, 2009

[SDG09] Prithviraj Sen, Amol Deshpande and Lise Getoor.
PrDB: Managing and Exploiting Rich Correlations in Probabilistic Databases.
In VLDB Journal, 2009.

[SDG10] Prithviraj Sen, Amol Deshpande and Lise Getoor.
Read-Once Functions and Query Evaluation in Probabilistic Databases.
In *VLDB*, 2009.

[SDG07] Prithviraj Sen, Amol Deshpande and Lise Getoor.
Representing Tuple and Attribute Uncertainty in Probabilistic Databases.
In *DUNE (ICDM)*, 2007.

Thank you.