Tractability in Probabilistic Databases

Dan Suciu
University of Washington

Motivation

Lots of uncertain data:

• *External data* in business intelligence: blogs, emails, twitter, social network data

• Data generated by large *information extraction* systems

• Data integration in the presence of uncertainty

• Specialized data (RFID, scientific data)
Probabilistic Databases

• In traditional databases: data is *certain*
• In new applications: data is *uncertain*

• **Probabilistic Databases**: model uncertainty using probabilities

• **Theoretical foundations**: logic + probability
Landscape of PDB Research

Incomplete list:
• Representation
  – [Widom’05, Antova’07, Sen’07, Benjelloun’08]
• Query evaluation
  – [Dalvi’04, Antova’08, Olteanu’09, Koch’08, Kimelfeld’09, Deutsch’10]
• Data integration, data exchange
  – [Dong’07, Agrawal’10, Fagin’10]
• Applications
  – [Nierman’02, Keulen’05, Gupta’06, Hassanzadeh’09, Stoyanovich’10]
• Ranking
  – [Re’07, Soliman’07, Zhang’08, Li’09]
• Indexing
  – [Letchner’09, Kanagal’09, Kimura’10]
• Pushing ML in the db engine
  – [Wang’08, Wick’10]
Landscape of PDB Systems

• Some research prototypes:
  – MaybeMS (Oxford&Cornell), Trio (Stanford), MystiQ (UW), ProbDB (Maryland), Orion (Purdue)
  – They do not scale to large datasets

• Commercial systems: NONE!
  – Why? Because to date we do not know how to build scalable probabilistic database systems

Main challenge: Query Evaluation
This Talk: Query Evaluation

- I will describe recent progress: *tractable queries*
- I will discuss where we are stuck: *intractable queries*

This talk is based on:

- Dalvi, S. *Efficient query evaluation on probabilistic databases.* VLDB’04
- Dalvi, S. *The Dichotomy of Conjunctive Queries on Probabilistic Structures*, PODS’07
- Dalvi, Schnaitter, S.: *Computing query probability with incidence algebras.* PODS’10
- Gatterbauer, Jha, S., *Dissociation and Propagation for Efficient Query Evaluation over Probabilistic Databases.* MUD'2010

Outline

• Problem Statement

• Intractable queries

• Tractable queries

• Summary and Open Problems
Every tuple $t$ in $D = \text{random variable (present/absent)}$  
Possible worlds semantics:  
$D = (W, p)$, where $p : W \rightarrow [0,1]$ s.t. $\sum_{W \in W} p(W) = 1$  

$$
\begin{array}{c|c|c|c|}
\text{A} & \text{B} & \text{S(A,B)} \\
\hline
a_1 & b_1 & .7 \\
\hline
a_2 & b_2 & .4 \\
\hline
a_2 & b_3 & .2 \\
\hline
a_2 & b_4 & .2 \\
\hline
a_3 & b_5 & .5 \\
\end{array}
$$

$R(A)$  
$a_1$  
$\text{.4}$  
$a_2$  
$\text{.2}$  
$a_3$  
$\text{.5}$  

$8$ tuples $\rightarrow 2^8$ possible worlds  
$W = \{W_1, W_2, \ldots, W_{256}\}$  

Need correlations? Normalize database!  

In this talk = all tuples are independent
Queries = UCQ

Unions of Conjunctive Queries:

\[ Q := R(x_1, \ldots, x_k) \mid \exists x. Q \mid Q_1 \land Q_2 \mid Q_1 \lor Q_2 \]

In this talk, I consider only Boolean queries:

\[ \exists x. \exists y. R(x) \land S(x, y) \equiv R(x), S(x, y) \]

Traditional database: \( D \models Q \) (true/false)

Probabilistic database: \( P(Q) = \sum_{W \in \mathbf{w} : W \models Q} p(W) \) (in \([0,1]\))
Lineage

Query Q + Database D = Lineage $F_Q$

- Every tuple $t$ in $D$ = Boolean variable $X_t$
- The *lineage* $F_Q$ = a propositional formula
  - Definition: next slide
  - Intuitively: it says when Q is true on D
- Terminology alert: what we call here *lineage* corresponds to the *PosBool* semiring [Tannen, ICDT’10]
Definition of Lineage $F_Q$

**Def.**

\[
\begin{align*}
F_t & = X_t \\
F_{\exists x. Q} & = F_{Q[a_1/x]} \lor \ldots \lor F_{Q[a_n/x]} \\
F_{Q_1 \land Q_2} & = F_{Q_1} \land F_{Q_2} \\
F_{Q_1 \lor Q_2} & = F_{Q_1} \lor F_{Q_2}
\end{align*}
\]

($t = \text{ground tuple } t$)

(Active dom. = \{a_1, \ldots, a_n\})

Example

\[
\begin{array}{c|c|c}
R(A) & \multicolumn{2}{|c|}{S(A,B)} \\
\hline
A & a_1 & b_1 \\
& a_2 & b_2 \\
& a_3 & b_3 \\
& a_2 & b_4 \\
& a_2 & b_5 \\
\end{array}
\]

For any fixed Q, $F_Q$ has a DNF of size polynomial in size(D)

$F_Q = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \lor X_2 Y_4 \lor X_2 Y_5$

$Q = R(x), S(x,y)$
Problem Statement

**Problem.** Fix Q. Given D, compute \( P(Q) \); Equivalently: compute \( P(F_Q) \).

This talk I discuss tractable and intractable queries

**Dichotomy Theorem.** For any, fixed UCQ Q:
- either \( P(Q) \) is in PTIME -- *tractable*
- or \( P(Q) \) is hard for FP\(^{\#P} \) -- *intractable*

This is a form of probabilistic inference (next)
Discussion: Probabilistic Inference

Graphical models:

**Bayesian Network**

\[
P(X,Y,Z,U) = P(X) P(Y) P(Z|X,Y) P(U|Y)
\]

**Markov Network**

\[
\Phi(X,Y,Z,U) = \frac{1}{Z} \Phi_1(X,Y,Z) \Phi_2(Y,U)
\]

Propositional formula = *very* special case

\[
F = X Y \lor Y Z \lor X Z
\]

[Pearl’88, Koller&Friedman’09, Darwiche’09]
## Discussion: Probabilistic Inference

<table>
<thead>
<tr>
<th>Model</th>
<th>Graphical Models</th>
<th>Prob DBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex</td>
<td>$P(X) = 1/Z \prod_i \Phi_i(X_i)$</td>
<td>Simple Independent tuples</td>
</tr>
<tr>
<td>Simple</td>
<td>$Q = P(X,Y \mid Z,U,V)$</td>
<td>Complex</td>
</tr>
</tbody>
</table>

- **Network**
  - Static
  - Bayesian/Markov Network
  - Dynamic
    - $F_Q = Q + D$

- **Complexity**
  - $f(\text{tree-width}, \text{network})$
  - $f(Q, D)$

Inference in GM’s is exponential in $tw$; ill suited for PDB. Need new, query-based approach: *data complexity*
Outline

• Problem Statement

• Intractable queries

• Tractable queries

• Summary and Open Problems
Intractable Queries

$H_0 = R(x), S(x, y), T(y)$  
$H_1 = R(x_0), S(x_0, y_0) \lor S(x_1, y_1), T(y_1)$  
$H_2 = R(x_0), S_1(x_0, y_0) \lor S_1(x_1, y_1), S_2(x_1, y_1) \lor S_2(x_2, y_2), T(y_2)$  
$H_3 = R(x_0), S_1(x_0, y_0) \lor S_1(x_1, y_1), S_2(x_1, y_1) \lor S_2(x_2, y_2), S_3(x_2, y_2) \lor S_3(x_3, y_3), T(y_3)$  

. . .

Theorem. Data complexity of $H_k$ is hard for FP$^\#P$

Will give the proof for $H_0$ and $H_1$ next
Model Counting

**Definition** A propositional formula $F$ is a *Positive, Partite, 2DNF* if $F = \bigvee_{i,j} X_i Y_j$

**Example:**
$$F = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \lor X_2 Y_4 \lor X_2 Y_5$$

**Theorem** The problem: "given a PP2DNF $F$, count the number of satisfying assignments $\#F$" is \#P-hard.

[Provan&Ball’83]

Holds also for PP2CNF
Proof: \( H_0 \) is hard for \( \text{FP}^{\#\text{P}} \)

Reduction from \( \#\text{PP2DNF} \)
- Given \( F = X_{i_1} Y_{j_1} \lor X_{i_2} Y_{j_2} \lor \ldots \)

construct \( D = \)

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
X_1 & Y_{j_1} \\
\hline
X_2 & Y_{j_2} \\
\hline
\ldots & \ldots \\
\hline
\end{array}
\]

- Assignment \( \theta \leftrightarrow \) possible world \( R^W, T^W \)
- \( \theta \) satisfies \( F \leftrightarrow (R^W,S,T^W) \models H_0 \)
- Therefore \( P(H_0) = \#F / 2^n \), where \( n = \text{number of variables} \)

We can compute \( \#F \) using an oracle for \( P(H_0) \); hence \( H_0 \) is hard for \( \text{FP}^{\#\text{P}} \)
Proof: $H_1$ is hard for $\text{FP}^{\#P}$

$H_1 = R(x_0), S(x_0, y_0) \lor S(x_1, y_1), T(y_1)$

By reduction from $\#\text{PP2CNF}$:

- $F = (X_{i_1} \lor Y_{j_1}) \land (X_{i_2} \lor Y_{j_2}) \land \ldots \land (X_{i_m} \lor Y_{j_m})$

Construct $D = [\text{Table}]$

- Assignment $\theta \leftrightarrow$ possible world $R^W, T^W$ (no $S$!)

- $P(\neg H_1 \mid \theta) = z^{\#\text{cnt}(\theta)}$, where $\#\text{cnt}(\theta) = |\{k \mid \theta(X_{i_k} \lor Y_{j_k}) = \text{true}\}|$

- $P(\neg H_1) = \sum_{\theta} P(\neg H_1 \mid \theta) P(\theta) = \frac{1}{2^n} \sum_{k=0}^{m} a_k z^k = f(z)$
  where $a_k = |\{\theta \mid \#\text{cnt}(\theta) = k\}|$

- Compute $f(z)$ at $m+1$ distinct values $z$, using oracle for $P(H_1)$. Obtain all coefficients $a_0, a_1, \ldots, a_m$

- Return $\#F = a_m$

We can compute $\#F$ using an oracle for $P(H_1)$; hence $H_1$ is hard for $\text{FP}^{\#P}$.
Discussion of Intractable Queries

• $H_0$ corresponds to a Restricted Boltzmann Machine
  
  [Salakhutdinov’10]

• The hardness proof for $H_k$, $k \geq 2$, and other queries, requires significant extensions

• There are other intractable queries: will describe all of them shortly

Open problems: Is the model counting problem for an intractable query #P hard? How do we evaluate/approximate them?
Outline

• Problem Statement

• Intractable queries

• Tractable queries
  – Safe plans
  – An algorithm for UCQ queries

• Summary and Open Problems
Overview

• Traditional query processing is done with query plans using *relational operators*

• Query processing on probabilistic databases is done with query plans using simple *extensions of the relational operators*
Example

\[ Q = \exists x. \exists y. R(x), S(x, y) \]

\[ P(Q) = 1 - \{1 - p_1 \ast [1 - (1 - q_1) \ast (1 - q_2)]\} \ast \{1 - p_2 \ast [1 - (1 - q_3) \ast (1 - q_4) \ast (1 - q_5)]\} \]

\[ F_Q = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \lor X_2 Y_4 \lor X_2 Y_5 = X_1(Y_1 \lor Y_2) \lor X_2(Y_3 \lor Y_4 \lor Y_5) \]

\[
\begin{array}{|c|c|c|}
\hline
A & B & P \\
\hline
a_1 & b_1 & q_1 \\
\hline
a_1 & b_2 & q_2 \\
\hline
a_2 & b_3 & q_3 \\
\hline
a_2 & b_4 & q_4 \\
\hline
a_2 & b_5 & q_5 \\
\hline
\end{array}
\]

*S See Sudeepa Roy’s talk on Tuesday*
Operators

Independent join operator

Independent projection operator

... other operators for self-joins and for unions

Operators...

other operators
\[
Q = \exists x. \exists y. R(x), S(x, y)
\]

\[
1 - (1 - p_1 q_1)(1 - p_1 q_2)(1 - p_2 q_3)(1 - p_2 q_4)(1 - p_2 q_5)
\]

Wrong

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

Right

[Dalvi&S.'04]
Discussion

• An extended operator manipulates probabilities explicitly: join, projection, etc

• A safe plan is a plan that computes the query probability correctly

• Next: I will give an algorithm for computing all tractable UCQ queries. Each tractable query can be computed using a safe plan, using a small set of extended operators, but I will not discuss safe plans in the rest of the talk
Outline

• Problem Statement

• Intractable queries

• Tractable queries
  – Safe plans
  – An algorithm for UCQ queries

• Summary and Open Problems
CQ with Self-Joins

\[ Q_J = q_1, \ q_2 = R(x_1),S(x_1,y_1), \ T(x_2),S(x_2,y_2) \]

\[ F_J = [X_1(Y_1 \lor Y_2) \lor X_2 (Y_3 \lor Y_4 \lor Y_5)] \land [Z_1(Y_1 \lor Y_2) \lor Z_2 (Y_3 \lor Y_4 \lor Y_5)] \]

Not read-once

\[ Q_U = q_1 \lor q_2 = R(x_1),S(x_1,y_1) \lor T(x_2),S(x_2,y_2) \]

\[ F_U = (X_1 \lor Z_1) (Y_1 \lor Y_2) \lor (X_2 \lor Z_2) (Y_3 \lor Y_4 \lor Y_5) \]

Read-once

\[ P(Q_J) = P(q_1, \ q_2) = P(q_1) + P(q_2) - P(q_1 \lor q_2) \]

[Dalvi, Schaiter, S.’10]
Discussion

• In order to handle self-joins, we needed \( V \)

• CQ = not a natural class to study

• UCQ = the natural class to study

• Will give the algorithm next as five rules
Five Rules for Query Evaluation

Rule 1: Inclusion/Exclusion Formula
\[ P(Q_1 \land Q_2 \land Q_3) = P(Q_1) + P(Q_2) + P(Q_3) - P(Q_1 \lor Q_2) - P(Q_1 \lor Q_3) - P(Q_2 \lor Q_3) + P(Q_1 \lor Q_2 \lor Q_3) \]

Note 1: this is the dual of the more popular:
\[ P(Q_1 \lor Q_2 \lor Q_3) = \ldots \]

Note 2: this rule is not used for inference on GM or on propositional formulae; new in PDBs
Five Rules for Query Evaluation

**Definition.** z is called *separator variable* if:
- z occurs in every atom A in Q (z is a *root variable*)
- If two atoms A₁, A₂ in Q are unifiable, then z occurs on a common position in A₁, A₂.

**Rule 2: Independent Project** If z is separator variable, 
\[ P(\exists z.Q) = 1 - (1 - P(Q[a₁/z])) \times (1 - P(Q[a₂/z])) \times \ldots \]

Where active domain = \{a₁, a₂, a₃, ... aₙ\}

**Example:**  
\[ Q_U = R(x₁), S(x₁,y₁) \lor T(x₂), S(x₂,y₂) \]
\[ = \exists z.(\exists y₁.R(z), S(z,y₁) \lor \exists y₂.T(z), S(z,y₂)) \]

z is “separator variable”
Rule 3: Independent Join
\[ P(Q_1 \land Q_2) = P(Q_1) \times P(Q_2) \]

Rule 4: Independent Union
\[ P(Q_1 \lor Q_2) = 1 - (1 - P(Q_1)) \times (1 - P(Q_2)) \]

If \( Q_1, Q_2 \) are independent (no common symbols)
Five Rules for Query Evaluation

Rule 5a: Ranking attribute-constant
Given attribute $R(...A...)$ and constant ‘a’, replace $R$ in $Q$ with $R = R_1 \cup R_2$, where $R_1 = \sigma_{A='a'}(R)$, $R_2 = \sigma_{A\neq'a'}(R)$

Rule 5b: Ranking attribute-attribute
Given two attributes $R(...A...B...)$ replace $R$ in $Q$ with $R = R_1 \cup R_2 \cup R_3$, where $R_1 = \sigma_{A<B}(R)$, $R_2 = \sigma_{A>B}(R)$, $R_3 = \sigma_{A=B}(R)$

Example: $q = R(x,y), R(y,x)$ rewrite to $q = R_1(x,y), R_2(y,x) \lor R_3(z,z)$

Note: ranking is applied before all other rules
Summary

• Five rules:
  1. Inclusion/exclusion
  2. Existential quantifier elimination (separator !)
  3. Independent AND
  4. Independent OR
  5. Ranking

• Each rule reduces $P(Q)$ to a simpler $P(Q')$
  – If we succeed, then $P(Q)$ in PTIME
  – If we fail, then $P(Q)$ is hard for FP$^{\#P}$
An Example

\[ Q_V = R(x_1), S(x_1, y_1) \lor S(x_2, y_2), T(y_2) \lor R(x_3), T(y_3) \]

\[ = H_1 \text{ (hard !)} \]

[Dalvi, Schaitter, S.’10]
An Example

$Q_V = R(x_1), S(x_1, y_1) \lor S(x_2, y_2), T(y_2) \lor R(x_3), T(y_3)$

Disconnected query

$= H_1$ (hard !)

[Dalvi,Schaitter,S.'10]
An Example

Disconnected query

DNF

$$Q_V = R(x_1), S(x_1,y_1) \lor S(x_2,y_2), T(y_2) \lor R(x_3), T(y_3)$$

CNF

$$Q_V = [S(x_2,y_2), T(y_2) \lor R(x_3)] \land [R(x_1), S(x_1,y_1) \lor T(y_3)]$$

$$= H_1 \text{ (hard !)}$$

[Dalvi, Schaiter, S.’10]
An Example

\[ Q_V = R(x_1), S(x_1, y_1) \lor S(x_2, y_2), T(y_2) \lor R(x_3), T(y_3) \]

Disconnected query

\[ = H_1 \text{ (hard !)} \]

DNF

CNF

\[ Q_V = [S(x_2, y_2), T(y_2) \lor R(x_3)] \land [R(x_1), S(x_1, y_1) \lor T(y_3)] \]

Inclusion/exclusion:

\[ P(Q_V) = P(q_1 \land q_2) = P(q_1) + P(q_2) - P(q_1 \lor q_2) \]

\[ = R(x_3) \lor T(y_3) \]

\[ Q_V \text{ has a subquery } H_1 \text{ that is hard, yet it is in PTIME !} \]

[Dalvi, Schaitter, S.’10]
Another Example

\[ Q_w = [R(x_0), S_1(x_0, y_0) \lor S_2(x_2, y_2), S_3(x_2, y_2)] \land /* Q1 */ \]
\[ [R(x_0), S_1(x_0, y_0) \lor S_3(x_3, y_3), T(y_3)] \land /* Q2 */ \]
\[ [S_1(x_1, y_1), S_2(x_1, y_1) \lor S_3(x_3, y_3), T(y_3)] /* Q3 */ \]

[Dalvi, Schaitter, S.’10]
Another Example

\[ Q_W = [R(x_0), S_1(x_0, y_0) \lor S_2(x_2, y_2), S_3(x_2, y_2)] \land /* Q1 */ \\
[ R(x_0), S_1(x_0, y_0) \lor S_3(x_3, y_3), T(y_3)] \land /* Q2 */ \\
[S_1(x_1, y_1), S_2(x_1, y_1) \lor S_3(x_3, y_3), T(y_3)] /* Q3 */ \]

P(\(Q_W\)) = P(\(Q_1\)) + P(\(Q_2\)) + P(\(Q_3\)) + \\
- P(\(Q_1 \lor Q_2\)) - P(\(Q_2 \lor Q_3\)) - P(\(Q_1 \lor Q_3\)) \\
+ P(\(Q_1 \lor Q_2 \lor Q_3\)) \\
Also = H_3

\[= H_3 \text{ (hard !)} \]
Another Example

\[ Q_W = [R(x_0), S_1(x_0,y_0) \lor S_2(x_2,y_2), S_3(x_2,y_2)] \land /* Q1 */ \\
[ R(x_0), S_1(x_0,y_0) \lor S_3(x_3,y_3), T(y_3)] \land /* Q2 */ \\
[ S_1(x_1,y_1), S_2(x_1,y_1) \lor S_3(x_3,y_3), T(y_3)] /* Q3 */ \]

\[
P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) + \\
- P(Q_1 \lor Q_2) - P(Q_2 \lor Q_3) - P(Q_1 \lor Q_3) \\
+ P(Q_1 \lor Q_2 \lor Q_3) \\
= H_3 \text{ (hard !)}
\]

Also = H_3

How do we need to detect cancellations in the inclusion/exclusion rule?
August Ferdinand Möbius
1790-1868

- Möbius strip
- Möbius function $\mu$ in number theory
- Generalized to lattices [Stanley’97, Rota’09]
- And now to queries!
From Query Q to Lattice L(Q)

• Write Q in CNF:
  \[ Q = Q_1 \land Q_2 \land \ldots \land Q_m. \]
• For \( s \subseteq [m] \), denote \( Q_s = \bigvee_{i \in s} Q_i \)

**Def.** The \textbf{CNF lattice} of Q is \( L(Q) = (L, \leq) \) where:
• \( L = \{ Q_s \mid s \subseteq [m] \} \) (up to logical equivalence)
• \( Q_{s_1} \leq Q_{s_2} \) if the logical implication \( Q_{s_2} \rightarrow Q_{s_1} \) holds

[Dalvi,Schaitter,S.’10]
Example

\[ Q_W = [R(x_0), S_1(x_0, y_0) \lor S_2(x_2, y_2), S_3(x_2, y_2)] \land \quad /* Q1 */ \\
[R(x_0), S_1(x_0, y_0) \lor S_3(x_3, y_3), T(y_3)] \land \quad /* Q2 */ \\
[S_1(x_1, y_1), S_2(x_1, y_1) \lor S_3(x_3, y_3), T(y_3)] \quad /* Q3 */ \]

\[ L(Q_W) = \hat{1} = \max(L) \]

Blue nodes are in PTIME, Red nodes are \#P hard.
The Möbius’ Function

**Def.** The Möbius function:
\[ \mu(1, 1) = 1 \]
\[ \mu(u, 1) = - \sum_{u < v \leq 1} \mu(v, 1) \]

**Möbius’ Inversion Formula:**
\[ P(Q) = - \sum_{Q_i < 1} \mu(Q_i, 1) P(Q_i) \]

**Rule 1 (revised)**
Inclusion/Exclusion → Möbius’ Inversion Formula
The Dichotomy

Dichotomy Theorem  Fix a UCQ query Q.
1. Algorithm terminates, then P(Q) is in PTIME
2. Algorithm fails, then P(Q) is hard for FP#P

Note 1: dichotomy into PTIME/FP#P based on “syntax” where “syntax” includes the Möbius function!

Note 2: the query complexity is open.

Note 3: we cannot avoid the Möbius function (next).

[Dalvi, Schaitter, S. ’10]
**Representation Theorem**

**THEOREM** For every lattice L, there exists Q s.t. L(Q) ≅ L and:
1. The query at \( \hat{0} (= \min(L)) \) is hard for FP\(^\#P\)
2. All other queries are in PTIME

**Proof**
- Let \( u_0, u_1, \ldots, u_k \) be the “join irreducibles” of L
- Associate them to the \( k+1 \) components of \( H_k = h_{k0} \lor h_{k1} \lor \ldots \lor h_{kk} \)
- For every \( v \in L \), define \( Q_v = \lor \{h_{ki} | u_i \not\leq v\} \)
- Define \( Q = \land \{Q_v | v = \text{co-atom in } L\} \)

\(Q\) is in PTIME iff \( \mu(\hat{0}, \hat{1}) = 0 \)!
Summary of Tractable Queries

Five *simple* rules

• Each rule has operator (not shown) → *safe plan*
• Open problem: engineering challenges of safe plans

Other approaches to compute $P(F)$ include: read-once formulae, OBDDs, FBDDs, d-DNNF’s

• For probabilistic databases, the Möbius Rule rules them all (see Abhay Jha’s talk Tuesday)
Preview of Abhay Jha’s talk on Tuesday
Outline

• Problem Statement

• Intractable queries

• Tractable queries

• Summary and Open Problems
Summary

• Why we care:
  – Strong demand for managing uncertain data

• What is difficult:
  – Computational complexity of Logic + Probabilities

• What we have seen today:
  – Tractable queries: the five rules are simple
  – Intractable queries: hardness proofs are difficult
  – Dichotomy into PTIME/FP^#P based on syntax
Open Problems

- Correlations using views:
  - Markov Logic \[ \text{[Richardson&Domingos]} \] \( \rightarrow \) Markov DBs?
- Model counting for \( H_0 \) (and others) \#P hard?
- Efficient approximation of \( H_0 \) (and others)?
- Query complexity for checking hardness?
- Data (and query) complexity for:
  - FO (\( \neg, \forall \)), interpreted predicates (\( \neq, < \))?
- Characterize UCQ(FBDD), UCQ(d-DNNF)
- Prove UCQ(d-DNNF) \( \neq \) UCQ(PTIME)
The 4x Grand Challenge

**Challenge**: make probabilistic dbs run at most 4x slower than deterministic dbs, by using approximations

**Suggested approach:**
- A rich class of tractable UCQ queries are known
- Given any Q, find all tractable Q’, s.t. P(Q) \( \approx \) P(Q’):
  - Model theoretic (Robert Fink’s talk on Tuesday)
  - Dissociation [Gatterbauer’10] [http://LaPushDB.com](http://LaPushDB.com)

This is an **engineering challenge**: search space for Q’; optimize and execute Q’
Thank You!

Questions?