Outline

• Part 1: Motivation
• Part 2: Probabilistic Databases
• Part 3: Weighted Model Counting
• Part 4: Lifted Inference for WFOMC
• Part 5: Completeness of Lifted Inference
• Part 6: Query Compilation
• Part 7: Symmetric Lifted Inference Complexity
• Part 8: Open-World Probabilistic Databases
• Part 9: Discussion & Conclusions
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)

\[ D = \]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>Alice</td>
<td>2009</td>
</tr>
<tr>
<td>Alice</td>
<td>2010</td>
</tr>
<tr>
<td>Bob</td>
<td>2009</td>
</tr>
<tr>
<td>Carol</td>
<td>2010</td>
</tr>
</tbody>
</table>

**Smoker**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>Alice</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>Carol</td>
</tr>
<tr>
<td>Bob</td>
<td>Carol</td>
</tr>
<tr>
<td>Carol</td>
<td>Bob</td>
</tr>
</tbody>
</table>
What Everyone Should Know about Databases

Database: relations (= tables)

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>2009</td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>2010</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>2009</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>2010</td>
<td></td>
</tr>
</tbody>
</table>

Smoker

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>Carol</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>Carol</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

Friend

Query: First Order Formula

Q(z) = ∃x (Smoker(x,'2009') ∧ Friend(x,z))

Query answer: Q(D) =

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Carol</td>
</tr>
</tbody>
</table>

Find friends of smokers in 2009

Conjunctive Queries CQ = FO(∃, ∧)

Union of CQs UCQ = FO(∃, ∧, ∨)
What Everyone Should Know about Databases

**Database:** relations (= tables)

<table>
<thead>
<tr>
<th>Smoker</th>
<th>Friend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td><strong>Z</strong></td>
</tr>
<tr>
<td>Alice</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>Carol</td>
</tr>
<tr>
<td>Bob</td>
<td>Carol</td>
</tr>
<tr>
<td>Carol</td>
<td>Bob</td>
</tr>
</tbody>
</table>

**Query:** First Order Formula

Find friends of smokers in 2009

\[
Q(z) = \exists x \ (\text{Smoker}(x,'2009') \land \text{Friend}(x,z))
\]

**Query answer:** \( Q(D) = \)

<table>
<thead>
<tr>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Carol</td>
</tr>
</tbody>
</table>

Conjunctive Queries \( CQ = \text{FO}(\exists, \land) \)

Union of CQs \( \text{UCQ} = \text{FO}(\exists, \land, \lor) \)

**Boolean Query:** FO Sentence

\[
Q = \exists x \ (\text{Smoker}(x,'2009') \land \text{Friend}(x,'Bob'))
\]

**Query answer:** \( Q(D) = \text{TRUE} \)
What Everyone Should Know about Databases

<table>
<thead>
<tr>
<th>Declarative Query</th>
<th>→</th>
<th>Query Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>“what”</td>
<td>→</td>
<td>“how”</td>
</tr>
</tbody>
</table>
What Everyone Should Know about Databases

Declarative Query → Query Plan

“what” → “how”

\[ Q(z) = \exists x \ (\text{Smoker}(x, '2009') \land \text{Friend}(x, z)) \]
What Everyone Should Know about Databases

Declarative Query → Query Plan
“what” → “how”

\[ Q(z) = \exists x \ (\text{Smoker}(x, '2009') \land \text{Friend}(x, z)) \]

**Logical Query Plan**
What Everyone Should Know about Databases

Declarative Query

“what”

→

Query Plan

“how”

Q(z) = \exists x (Smoker(x,’2009’) \land Friend(x,z))

Optimize

Logical Query Plan

Logical Query Plan
What Everyone Should Know about Databases

Declarative Query → Query Plan
“what” → “how”

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

Logical Query Plan

Physical Query Plan

Optimize

Hash-based deduplication

Merge-join

Index-scan

Logical Query Plan

Physical Query Plan
What Every **Researcher** Should Know about Databases

Problem: compute $Q(D)$

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

This talk: query = **blue**, data = **red**
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

This talk: query = blue, data = red
What Every Researcher Should Know about Databases

Problem: compute $Q(D)$

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

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

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

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

This talk: query = blue, data = red
Probabilistic Databases

• A probabilistic database = relational database where each tuple is a random variable

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

• In this talk: tuples are independent events
Example

Probabilistic database $D$:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>$p_1$</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>$p_2$</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>$p_3$</td>
</tr>
</tbody>
</table>
Example

Probabilistic database $D$:  

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>$p_1$</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>$p_2$</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>$p_3$</td>
</tr>
</tbody>
</table>

Possible worlds semantics:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

$p_1 p_2 p_3$
Example

Probabilistic database $D$:  

$$p_1 p_2 p_3$$  

$$(1-p_1)p_2 p_3$$

Possible worlds semantics:

$$\begin{array}{c|c|c|c} 
 x & y & P \\
 A & B & p_1 \\
 A & C & p_2 \\
 B & C & p_3 \\
\end{array}$$  

Friend
Example

Probabilistic database $D$:  

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>$p_1$</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>$p_2$</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>$p_3$</td>
</tr>
</tbody>
</table>

Possible worlds semantics:

$(1-p_1)(1-p_2)(1-p_3)$
Query Semantics

Fix a Boolean query $Q$, probabilistic database $D$:

$$P(Q \mid D) = P_D(Q) = \text{marginal probability of } Q \text{ on possible words of } D$$
An Example

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

\[ P(Q \mid D) = \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>q₁</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>q₂</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>q₃</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>q₄</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>q₅</td>
</tr>
</tbody>
</table>
An Example

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

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

<table>
<thead>
<tr>
<th>Smoker</th>
<th>x</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>p_1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>p_2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>p_3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Friend</th>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>q_1</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>q_2</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>q_3</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>q_4</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>q_5</td>
<td></td>
</tr>
</tbody>
</table>
\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

**An Example**

\[
P(Q | D) = p_1 \left[ 1 - (1 - q_1)(1 - q_2) \right]
\]
Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x, y)

An Example

\[ P(Q \mid D) = \]

\[ p_1 \left[ 1 - (1 - q_1)(1 - q_2) \right] \]

\[ 1 - (1 - q_3)(1 - q_4)(1 - q_5) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>q_1</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>q_2</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>q_3</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>q_4</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>q_5</td>
</tr>
</tbody>
</table>
\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

**An Example**

\[
P(Q \mid D) = p_1 \left[ (1-(1-q_1))(1-q_2) \right] + p_2 \left[ (1-(1-q_3))(1-q_4)(1-q_5) \right]
\]

**Smoker**

<table>
<thead>
<tr>
<th>x</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( p_1 )</td>
</tr>
<tr>
<td>B</td>
<td>( p_2 )</td>
</tr>
<tr>
<td>C</td>
<td>( p_3 )</td>
</tr>
</tbody>
</table>

**Friend**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>( q_3 )</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>( q_4 )</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>( q_5 )</td>
</tr>
</tbody>
</table>
An Example

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

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

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>q_1</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>q_2</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>q_3</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>q_4</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>q_5</td>
</tr>
</tbody>
</table>
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 \( D \)
An Example

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

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

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>q_1</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>q_2</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>q_3</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>q_4</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>q_5</td>
</tr>
</tbody>
</table>

\[ \Pi_{\Phi} \]

\[ \Pi_{x} \]

\text{Smoker}(x) \land \text{Friend}(x,y) \]
An Example

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x,y) \]

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

$Q = \exists x \exists y \text{Smoker}(x) \land \text{Friend}(x, y)$

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

Given: probabilistic database $D$, query $Q$

Compute: $P(Q \mid D)$

Data complexity: fix $Q$, complexity $= f(|D|)$
Approaches to Compute $P(Q \mid D)$

- **Propositional inference:**
  - Ground the query $Q \rightarrow F_{Q,D}$, compute $P(F_{Q,D})$
  - This is **Weighted Model Counting** (later…)
  - 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 $Q$

[Olteanu’08, Jha’13, Dalvi’04, Dalvi’12]
Lifted Inference Rules

Preprocess $Q$ (omitted from this talk; see [Suciu’11]),
then apply these rules (some have preconditions)

\[
P(\neg Q) = 1 - P(Q)
\]

\[
P(Q_1 \land Q_2) = P(Q_1)P(Q_2)
\]

\[
P(Q_1 \lor Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))
\]

\[
P(\exists z \ Q) = 1 - \prod_{A \in \text{Domain}} (1 - P(Q[A/z]))
\]

\[
P(\forall z \ Q) = \prod_{A \in \text{Domain}} P(Q[A/z])
\]

\[
P(Q_1 \land Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \lor Q_2)
\]

\[
P(Q_1 \lor Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \land Q_2)
\]
Example

\[ Q = \forall x \forall y \ (\text{Smoker}(x) \lor \text{Friend}(x,y)) = \forall x \ (\text{Smoker}(x) \lor \forall y \ \text{Friend}(x,y)) \]

\[ P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \lor \forall y \ \text{Friend}(A,y)) \cdot \]

- Check independence:
  - Smoker(Alice) \lor \forall y \ \text{Friend}(Alice,y)
  - Smoker(Bob) \lor \forall y \ \text{Friend}(Bob,y)
Example

\[ Q = \forall x \forall y \ (\text{Smoker}(x) \lor \text{Friend}(x,y)) \]

\[ = \forall x \ (\text{Smoker}(x) \lor \forall y \ \text{Friend}(x,y)) \]

\[ P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \lor \forall y \ \text{Friend}(A,y)) \cdot \]

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

\[ Q = \forall x \forall y \ (\text{Smoker}(x) \lor \text{Friend}(x,y)) \]

\[ = \forall x (\text{Smoker}(x) \lor \forall y \text{Friend}(x,y)) \]

\[
\begin{align*}
P(Q) &= \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \lor \forall y \text{Friend}(A,y)) \\
&= \prod_{A \in \text{Domain}} [1 - (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y \text{Friend}(A,y)))] \\
&= \prod_{A \in \text{Domain}} [1 - (1 - P(\text{Smoker}(A))) \times (1 - \prod_{B \in \text{Domain}} P(\text{Friend}(A,B)))]
\end{align*}
\]

\[ \forall \text{-Rule} \]

\[ \lor \text{-Rule} \]

\[ \text{Check independence:} \]

\[ \text{Smoker}(\text{Alice}) \lor \forall y \text{Friend}(\text{Alice},y) \]

\[ \text{Smoker}(\text{Bob}) \lor \forall y \text{Friend}(\text{Bob},y) \]
Example

Example

\[ Q = \forall x \forall y \ (\text{Smoker}(x) \lor \text{Friend}(x,y)) \]

\[ = \forall x \ (\text{Smoker}(x) \lor \forall y \ \text{Friend}(x,y)) \]

\[ P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \lor \forall y \ \text{Friend}(A,y)) \cdot \]

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

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

Lookup the probabilities in the database
Example

\[ Q = \forall x \forall y (\text{Smoker}(x) \lor \text{Friend}(x,y)) \]

\[ = \forall x (\text{Smoker}(x) \lor \forall y \text{Friend}(x,y)) \]

\[ \text{P}(Q) = \prod_{A \in \text{Domain}} \text{P} (\text{Smoker}(A) \lor \forall y \text{Friend}(A,y)) \cdot \] \[ \text{\forall -Rule} \]

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

\[ \text{\lor -Rule} \]

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

\[ \text{\forall -Rule} \]

Lookup the probabilities in the database

Runtime = \text{O}(n^2).
## Discussion: CNF vs. DNF

<table>
<thead>
<tr>
<th>Databases</th>
<th>FO(∃, ∧)</th>
<th>Positive Clause</th>
<th>FO(∀, ∨)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunctive Queries CQ</td>
<td></td>
<td>Positive Clause</td>
<td>FO(∀, ∨)</td>
</tr>
<tr>
<td>Union of Conjunctive Queries UCQ</td>
<td>FO(∃, ∧, ∨) = ∃ Positive-DNF</td>
<td>Positive FO</td>
<td>FO(∀, ∧, ∨) = ∀ Positive-CNF</td>
</tr>
<tr>
<td>UCQ with “safe negation” UCQ⁻</td>
<td>∃ DNF</td>
<td>First Order CNF</td>
<td>∀ CNF</td>
</tr>
</tbody>
</table>

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

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

Lifted Inference Sometimes Fails.

\[ H_0 = \forall x \forall y (\text{Smoker}(x) \lor \text{Friend}(x,y) \lor \text{Jogger}(y)) \]

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

\[
\begin{align*}
H_0[\text{Alice}/x] &= \forall y (\text{Smoker}(\text{Alice}) \lor \text{Friend}(\text{Alice},y) \lor \text{Jogger}(y)) \\
H_0[\text{Bob}/x] &= \forall y (\text{Smoker}(\text{Bob}) \lor \text{Friend}(\text{Bob},y) \lor \text{Jogger}(y))
\end{align*}
\]

Computing \( P(H_0 \mid D) \) is \#P-hard in \( |D| \)
(Proof: later…)

Dependent
Discussion

Lifted Inference Sometimes Fails.

\[ H_0 = \forall x \forall y \ (\text{Smoker}(x) \lor \text{Friend}(x,y) \lor \text{Jogger}(y)) \]

The \( \forall \)-rule does not apply: \( H_0[Alice/x] \) and \( H_0[Bob/x] \) are dependent:

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

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

(Proof: later…)

Consequence: assuming PTIME \( \neq \#P \), \( H_0 \) is not liftable!
Summary

• Database $D = \text{relations}$
• Query $Q = \text{FO}$
• Query plans, query optimization
• Data complexity: fix $Q$, complexity $f(D)$
• Probabilistic DB’s = independent tuples
• Lifted inference: simple, but fails sometimes