Probabilistic Databases and Reasoning

Thomas Lukasiewicz and Dan Olteanu
University of Oxford
Probabilistic Databases and Reasoning

This 3-hour tutorial has two main parts:

1. Dan Olteanu: Probabilistic Databases
   Now: 8.30am - 10am.

2. Thomas Lukasiewicz: Probabilistic Reasoning
   Next: 10am - 10.30am, then a break, then 11am - 12pm.

Further 1-hour lectures on advanced topics in probabilistic databases:

1. DL invited talk today at 12.10pm
   Dan Suciu: Lifted Inference in Probabilistic Database

2. KR invited lecture tomorrow at 9.30am
   Dan Suciu: Query compilation: the View from the Database Side

KR features several more papers on probabilistic data and knowledge bases!
Probabilistic Databases

For the purpose of the first half of this tutorial:

Probabilistic data =

- Relational data
  + Probabilities that measure the degree of uncertainty in the data.

Long-term key challenges:

- Models for probabilistic data to capture data and its uncertainty.

- Query evaluation = Probabilistic inference
  Query answers are annotated with output probabilities.
Outline

Why Probabilistic Databases?

Probabilistic Data Models

The Query Evaluation Problem

Dichotomies for Query Evaluation

- The Hard Queries
- The Tractable Queries

Ranking Queries

Next Steps

References
We can unify logic and probability by defining distributions over possible worlds that are first-order model structures (objects and relations). Gaifman’64

Early work (80s and 90s):

- Basic data models and query processing
  
  Wong’82, Shoshani’82, Cavallo & Pittarelli’87, Barbará’92, Lakshmanan’97,’01, Fuhr & Röllke’97, Zimányi’97, ..

Recent wave (2004 - now):

- Computational complexity of query evaluation
- Probabilistic database systems

  Stanford (Trio), UW (MystiQ), Cornell & Oxford (MayBMS/SPROUT), IBM Almaden & Rice (MCDB), LogicBlox & Technion & Oxford (PPDL), Florida, Maryland, Purdue, Waterloo, Wisconsin, ..
Why This Interest in Probabilistic Databases?

Probabilistic relational data is commonplace. It accommodates several possible interpretations of the data weighted by probabilities.

- Information extraction: Probabilistic data inferred from unstructured data (e.g., web) text using statistical models
  - Google Knowledge Vault, DeepDive, NELL

- Manually entered data
  - Represent several possible readings with MayBMS [Antova’07]
  - Infer missing data with meta-rule semi-lattices [Stoyanovich’11]
  - Manage OCR data with Staccato/Google OCRopus [Kumar’11]

- Data cleaning
  - Represent several possible data repairs [Beskales’09]

- Data integration
  - Google Squared and SPROUT\(^2\) [Fink’11]

- Risk management (Decision support queries, hypothetical queries); ...
Possible segmentations of unstructured text

52-A Goregaon West Mumbai 400 076

<table>
<thead>
<tr>
<th>ID</th>
<th>HouseNo</th>
<th>Area</th>
<th>City</th>
<th>PinCode</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52</td>
<td>Goregaon West</td>
<td>Mumbai</td>
<td>400 062</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>52-A</td>
<td>Goregaon</td>
<td>West Mumbai</td>
<td>400 062</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>52-A</td>
<td>Goregaon West</td>
<td>Mumbai</td>
<td>400 062</td>
<td>0.4</td>
</tr>
<tr>
<td>1</td>
<td>52</td>
<td>Goregaon</td>
<td>West Mumbai</td>
<td>400 062</td>
<td>0.2</td>
</tr>
</tbody>
</table>

- Probabilities obtained using probabilistic extraction models (e.g., CRF)
  - The probabilities correlate with the precision of the extraction.
- The output is a ranked list of possible extractions
- Several segmentations are required to cover most of the probability mass and improve recall
  - Avoid empty answer to queries such as *Find areas in 'West Mumbai'*
Continuously Improving Information Extraction

Never-Ending Language Learner (NELL) database [Mitchell'15]

<table>
<thead>
<tr>
<th>Instance</th>
<th>Iteration</th>
<th>Date</th>
<th>Learned</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>biscutte swift is an animal</td>
<td>211</td>
<td>18-feb-2011</td>
<td></td>
<td>100.0</td>
</tr>
<tr>
<td>pedigree animals is a mammal</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>99.5</td>
</tr>
<tr>
<td>poppy seed holiday bread is a baked good</td>
<td>212</td>
<td>20-feb-2011</td>
<td></td>
<td>100.0</td>
</tr>
<tr>
<td>manuel criado de val is a South American person</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>99.5</td>
</tr>
<tr>
<td>dillon county airport is an airport</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>93.8</td>
</tr>
<tr>
<td>the sports team toronto blue jays was the winner of n1993 world series</td>
<td>212</td>
<td>20-feb-2011</td>
<td></td>
<td>96.9</td>
</tr>
<tr>
<td>mozart is a person who died at the age of 35</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>96.9</td>
</tr>
<tr>
<td>peoria and arizona are proxies for eachother</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>99.9</td>
</tr>
<tr>
<td>wuty tv is a TV affiliate of the network fox</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>96.9</td>
</tr>
<tr>
<td>white stripes collaborates with jack white</td>
<td>210</td>
<td>17-feb-2011</td>
<td></td>
<td>93.8</td>
</tr>
</tbody>
</table>
Manually entered census data

MayBMS manages $10^{10^6}$ possible readings of census data. [Antova’07]

<table>
<thead>
<tr>
<th>Social Security Number:</th>
<th>185</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Smith</td>
</tr>
<tr>
<td>Marital Status:</td>
<td>(2) married</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Social Security Number:</th>
<th>185</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Brown</td>
</tr>
<tr>
<td>Marital Status:</td>
<td>(4) widowed</td>
</tr>
</tbody>
</table>

We want to enter the information from forms like these into a database.

- What is the marital status of the first resp. the second person?
- What are the social security numbers? 185? 186? 785?
Much of the available information cannot be represented and is lost, e.g.

- Smith’s SSN is either 185 or 785; Brown’s SSN is either 185 or 186.
- Data cleaning: No two distinct persons can have the same SSN.
Staccato [Kumar’12]

- Stochastic automaton constructed from text using Google OCRopus.
- String *Ford* has the highest probability (0.21).
- String *Ford* has lower probability (0.12).

Staccato accommodates several possible readings of the text to increase recall.
Web Data Integration with Google Squared

- Tables instead of page links as answers to Google queries [Fink’11]
- Integration of data sources with contradicting information or different schemas, degrees of trust, and degrees of completion
- Confidence values mapped to [0,1]
Outline

Why Probabilistic Databases?

Probabilistic Data Models

The Query Evaluation Problem

Dichotomies for Query Evaluation

• The Hard Queries

• The Tractable Queries

Ranking Queries

Next Steps

References
NULL values are too uninformative.
We could instead incorporate all available possibilities:

- Smith’s SSN is either 185 or 785; Brown’s SSN is either 185 or 186.
- Smith’s M is either 1 or 2; Brown’s M is either 1, 2, 3, or 4.
Revisiting the Census Data Example

There are $2 \times 2 \times 2 \times 4 = 32$ possible readings of our two census entries.
**Incomplete Databases**

An **Incomplete Database** is a finite set of database instances \( W = (W_1, \ldots, W_n) \).

<table>
<thead>
<tr>
<th>( W_1 )</th>
<th>( W_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
</tr>
<tr>
<td>185</td>
<td>Brown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( W_3 )</th>
<th>( W_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
</tr>
<tr>
<td>185</td>
<td>Brown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( W_5 )</th>
<th>( W_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
</tr>
<tr>
<td>186</td>
<td>Brown</td>
</tr>
</tbody>
</table>

...  

Each \( W_i \) is a **possible world**.
An **Incomplete Database** is a finite set of database instances \( W = (W_1, \ldots, W_n) \).

<table>
<thead>
<tr>
<th></th>
<th>( W_1 )</th>
<th>( W_2 )</th>
<th>( W_3 )</th>
<th>( W_4 )</th>
<th>( W_5 )</th>
<th>( W_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>N</td>
<td>M</td>
<td>SSN</td>
<td>N</td>
<td>M</td>
<td>SSN</td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
<td>1</td>
<td>185</td>
<td>Smith</td>
<td>1</td>
<td>185</td>
</tr>
<tr>
<td>185</td>
<td>Brown</td>
<td>1</td>
<td>185</td>
<td>Brown</td>
<td>2</td>
<td>185</td>
</tr>
<tr>
<td>186</td>
<td>Brown</td>
<td>1</td>
<td>186</td>
<td>Brown</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Each \( W_i \) is a **possible world**.

Typical scenario: 200M people (2/3 US census), 50 questions, 1 in 10000 ambiguous (2 options)
- \( 2^{10^6} \) possible worlds
- A world is a table with 50 columns and 200M rows!

[Antova’07]

→ Key challenge: How to succinctly represent incomplete databases?
A **Probabilistic Database** is \((W, P)\), where \(W\) is an incomplete database and \(P : W \rightarrow [0, 1]\) is a probability distribution: \(\sum_{W_i \in W} P(W_i) = 1\).

<table>
<thead>
<tr>
<th>(W_1) : (P(W_1) = 0.1)</th>
<th>(W_2) : (P(W_2) = 0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
</tr>
<tr>
<td>185</td>
<td>Brown</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(W_3) : (P(W_3) = 0.1)</th>
<th>(W_4) : (P(W_4) = 0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
</tr>
<tr>
<td>185</td>
<td>Brown</td>
</tr>
<tr>
<td>185</td>
<td>Brown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(W_5) : (P(W_5) = 0.3)</th>
<th>(W_6) : (P(W_6) = 0.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>N</strong></td>
</tr>
<tr>
<td>185</td>
<td>Smith</td>
</tr>
<tr>
<td>186</td>
<td>Brown</td>
</tr>
</tbody>
</table>

For \(W = \{W_1, \ldots, W_6\}\),
\[\sum_{W_i \in W} P(W_i) = 1.\]
Succinct Representations of Incomplete/Probabilistic Data

Succinct or-set representation: [Imielinski’91]

<table>
<thead>
<tr>
<th>SSN</th>
<th>N</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>185,785</td>
<td>Smith</td>
<td>{ 1,2 }</td>
</tr>
<tr>
<td>185,186</td>
<td>Brown</td>
<td>{ 1,2,3,4 }</td>
</tr>
</tbody>
</table>

It exploits independence of possible values for different fields:

- Choice for Smith’s SSN independent of choice of for Brown’s SSN.
- Likewise, the probability distributions associated with these choices are independent (not shown).
BID: Alternative Representation of Our Or-Set

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>185</td>
<td>0.7</td>
</tr>
<tr>
<td>$t_1$</td>
<td>785</td>
<td>0.3</td>
</tr>
<tr>
<td>$t_2$</td>
<td>185</td>
<td>0.8</td>
</tr>
<tr>
<td>$t_2$</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>$t_1$</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>$t_2$</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>$t_2$</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>$t_2$</td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Interpretation:
The tuples within each block with the same key RID are disjoint. Each world contains one tuple per block, so the tuples within a block are mutually exclusive. Blocks are independent of each other. The choices of tuples within different blocks are independent. The aggregated probability of the worlds taking the first tuple of the first block in each relation is $0.7 \times 1 \times 0.9 = 0.63$. These block-independent disjoint (BID) relations are sometimes called x-relations or x-tables. Google squares are prime examples.
**BID: Alternative Representation of Our Or-Set**

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>185</td>
<td>0.7</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>785</td>
<td>0.3</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>185</td>
<td>0.8</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>Smith</td>
<td>1</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>Brown</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

**Interpretation:**
- The tuples within each block with the same key RID are *disjoint*. Each world contains one tuple per block, so the tuples within a block are mutually exclusive.
### BID: Alternative Representation of Our Or-Set

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>185</td>
<td>0.7</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>785</td>
<td>0.3</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>185</td>
<td>0.8</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>Smith</td>
<td>1</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>Brown</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

**Interpretation:**

- The tuples within each block with the same key RID are **disjoint**
  - Each world contains one tuple per block, so the tuples within a block are mutually exclusive.
- Blocks are **independent** of each other.
  - The choices of tuples within different blocks are independent.
  - The aggregated probability of the worlds taking the first tuple of the first block in each relation is \(0.7 \times 1 \times 0.9 = 0.63\).

These **block-independent disjoint** (BID) relations are sometimes called x-relations or x-tables. Google squares are prime examples.
More on BID Databases

BIDs also allow blocks with probabilities less than 1:

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>185</td>
<td>0.6</td>
</tr>
<tr>
<td>t₁</td>
<td>785</td>
<td>0.3</td>
</tr>
<tr>
<td>t₂</td>
<td>185</td>
<td>0.8</td>
</tr>
<tr>
<td>t₂</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>t₁</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>t₂</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>t₂</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>t₂</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>t₂</td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>Smith</td>
<td>0.9</td>
</tr>
<tr>
<td>t₂</td>
<td>Brown</td>
<td>1</td>
</tr>
</tbody>
</table>

Interpretation:

- There are worlds where the first block of each of the three relations is empty, e.g., the following world:

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₂</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>

The probability of this world is

\[
0.2 \times 1 \times 0.25 \times (1 - 0.6 - 0.3) \times (1 - 0.9) \times (1 - 0.8 - 0.1) = 5 \times 10^{-5}.
\]
Clarification notes to come with the previous slide and to answer questions posed during the tutorial:

* The two BIDs from the previous two slides are not equivalent since they do not represent the same probabilistic database! Furthermore, by allowing groups with empty instances, some tuples are only partially defined in the column-oriented representation.

* See [Antova’08] for column-oriented representation of relations with attribute-level uncertainty.
**TI: Tuple-Independent Databases**

*TI databases* are BID databases where each block has exactly one tuple.

TI databases are the simplest and most common probabilistic data model.

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>185</td>
<td>0.7</td>
</tr>
<tr>
<td>$t_2$</td>
<td>185</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>Smith</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>Brown</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>$t_2$</td>
<td>2</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Interpretation:

- Each tuple $t$ is in a random world with its probability $p(t)$.
- A relation with $n$ tuples, whose probabilities are less than 1, has $2^n$ possible worlds, since each tuple may be in or out.
- Our TI example has $2^4$ worlds: Any subset of the first and third relation and the entire second relation.
Are BID Databases Enough?

BIDs (and TIs) are good at capturing independence and local choice. What about correlations across blocks?

- Enforce the key dependency on SSN in each world.
  That is: Discard the worlds where both $t_1$ and $t_2$ have SSN = 185.

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>185</td>
<td>0.6</td>
</tr>
<tr>
<td>$t_1$</td>
<td>785</td>
<td>0.3</td>
</tr>
<tr>
<td>$t_2$</td>
<td>185</td>
<td>0.8</td>
</tr>
<tr>
<td>$t_2$</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Are BID Databases Enough?

BIDs (and TIs) are good at capturing independence and local choice. What about correlations across blocks?

- Enforce the key dependency on SSN in each world.
  - That is: Discard the worlds where both $t_1$ and $t_2$ have SSN = 185.

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>185</td>
<td>0.6</td>
</tr>
<tr>
<td>$t_1$</td>
<td>785</td>
<td>0.3</td>
</tr>
<tr>
<td>$t_2$</td>
<td>185</td>
<td>0.8</td>
</tr>
<tr>
<td>$t_2$</td>
<td>186</td>
<td>0.2</td>
</tr>
</tbody>
</table>

⇒

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>$\Phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>185</td>
<td>$X = 1$</td>
</tr>
<tr>
<td>$t_1$</td>
<td>785</td>
<td>$X = 2$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>185</td>
<td>$Y = 1 \land X \neq 1$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>186</td>
<td>$Y = 2$</td>
</tr>
</tbody>
</table>

This constraint is supported by a probabilistic version of conditional databases. [Imielinski’84]

Idea: Use random variables to encode correlations between tuples.

- Exclude the world where $t_1$ and $t_2$ have the same SSN 185 by using contradicting assignments for variable $X$.
- Transfer probabilities of tuples to probability distributions of variables.
A **PC database** is \((D, X, \Phi)\), where \(D\) is a relational database, \(X\) is a set of independent random variables, and \(\Phi\) is a function mapping each tuple in \(D\) to a propositional formula over \(X\).

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>(\Phi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>185</td>
<td>(X = 1)</td>
</tr>
<tr>
<td>(t_1)</td>
<td>785</td>
<td>(X = 2)</td>
</tr>
<tr>
<td>(t_2)</td>
<td>185</td>
<td>(Y = 1 \land X \neq 1)</td>
</tr>
<tr>
<td>(t_2)</td>
<td>186</td>
<td>(Y = 2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VAR</th>
<th>Dom</th>
<th>(P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X)</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>(X)</td>
<td>2</td>
<td>0.3</td>
</tr>
<tr>
<td>(Y)</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>(Y)</td>
<td>2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**Interpretation:**

- The **world table** (right) lists the probability distribution for each independent random variable in \(X\).
- Each total valuation of variables in \(X\) defines a world whose probability is the product of probabilities of the variable assignments.
- Each tuple \(t\) is **conditional** on the satisfiability of the formula \(\Phi(t)\) and is contained in those worlds defined by valuations that satisfy \(\Phi(t)\).
Clarification notes to come with the previous slide and to answer questions posed during the tutorial:

* The PC table from the previous slide is not equivalent to the BID table from two slides ago: While the PC table captures the key dependency on SSN, the BID table does not.

* However, the PC table is not the BID table where the key dependency is enforced: This is because we did not adjust the probabilities of the remaining worlds that satisfy the key dependency.

* The mechanism for this adjustment is called conditioning, see [Koch’08].
TIs and BIDs are Special Cases of PCs

Recall our previous TI database example:

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>185</td>
<td>0.7</td>
</tr>
<tr>
<td>t₂</td>
<td>185</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Here is a PC encoding of the above TI database:

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>Φ</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>185</td>
<td>s₁</td>
<td>0.7</td>
</tr>
<tr>
<td>t₂</td>
<td>185</td>
<td>s₂</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>Smith</td>
<td>1</td>
</tr>
<tr>
<td>t₂</td>
<td>Brown</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>t₂</td>
<td>2</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Idea:

- Consider a set of Boolean random variables
- Associate each tuple in the TI database with exactly one of them
- For instance, $s_1$ annotates $(t_1, 185)$ and $P(s_1) = 0.7$
- World table with variable assignments may be stored explicitly
Takeaways

Various representations for probabilistic databases of increasing expressiveness.

- **Most complex:** probabilistic conditioned databases. [Imielinski’84]
  - Trio’s ULDBs [Benjelloun’06] and MayBMS’ U-relations [Antova’07].
  - Completeness: They can represent any probabilistic database.

- **Mid-level:** block-independent disjoint databases. [Barbará’92]
  - MystiQ, Trio, MayBMS, SPROUT².
  - Prime examples of BIDs: Google squares.
  - Not complete, but achieve completeness via conjunctive queries over BIDs. [Poole’93]

- **Simplest:** tuple-independent databases.
  - The norm in real-world repositories like Google’s, DeepDive, and NELL.
  - Most theoretical work on complexity of query evaluation done for them.
  - Not complete even via unions of conjunctive queries.
  - However, inference in Markov Logic Networks is captured by relational queries on TI databases! See Dan Suciu’s invited DL’16 talk. Also work by Guy van den Broeck. [Jha’12]
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem
Dichotomies for Query Evaluation
- The Hard Queries
- The Tractable Queries
Ranking Queries
Next Steps
References
Possible Worlds Semantics

The underlying semantics of query evaluation in probabilistic databases:

**Possible worlds semantics**: Given a database $W = \{W_1, \ldots, W_n\}$ and a query $Q$, the query answer is $Q(W) = \{Q(W_1), \ldots, Q(W_n)\}$. 
Possible Worlds Semantics

The underlying semantics of query evaluation in probabilistic databases:

Possible worlds semantics: Given a database \( W = \{ W_1, \ldots, W_n \} \) and a query \( Q \), the query answer is \( Q(W) = \{ Q(W_1), \ldots, Q(W_n) \} \).

Investigations so far followed three main directions:

1. Possible and certain query answers for incomplete databases.

2. Probabilities of query answers for probabilistic databases.

3. Succinct representation of \( Q(W) \) for query languages and data models.

Approaches 1 & 2 close the possible worlds semantics: They compute one relation with answer tuples and possibly their probabilities.
Queries on Incomplete Databases

Given query $Q$ and incomplete database $W$:

- An answer $t$ is **certain**, if $\forall W_i \in W, t \in Q(W_i)$
- An answer $t$ is **possible** if $\exists W_i \in W, t \in Q(W_i)$

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>$W_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>SSN</strong></td>
</tr>
<tr>
<td>185</td>
<td>185</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$W_3$</th>
<th>$W_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>SSN</strong></td>
</tr>
<tr>
<td>185</td>
<td>185</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$W_5$</th>
<th>$W_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSN</strong></td>
<td><strong>SSN</strong></td>
</tr>
<tr>
<td>185</td>
<td>185</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Queries on Incomplete Databases

Given query $Q$ and incomplete database $W$:

- An answer $t$ is **certain**, if $\forall: W_i \in W, t \in Q(W_i)$
- An answer $t$ is **possible** if $\exists W_i \in W, t \in Q(W_i)$

<table>
<thead>
<tr>
<th>$W_1$</th>
<th>$W_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>SSN</td>
</tr>
<tr>
<td>185</td>
<td>185</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$W_3$</th>
<th>$W_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>SSN</td>
</tr>
<tr>
<td>185</td>
<td>185</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Let $W = \{ W_1, \ldots, W_6 \}$.

- Query $\exists N \exists M \text{Census}(S, N, M)$ has certain answer (185) and possible answers (185) and (186).

- Query $\exists S \exists M \text{Census}(S, N, M)$ has the same possible and certain answers (Smith) and (Brown).
Queries on Incomplete Databases

Several studies on this started back in the 90s for various models, in particular conditional databases.

[Abiteboul’91, O.’08a]

Hard tasks already for positive relational algebra:

- Tuple possibility is NP-complete
- Tuple certainty is coNP-complete

We next focus on probabilistic databases.
Given query $Q$ and probabilistic database $(W, P)$: The Marginal Probability of an answer $t$ is: $P(t) = \sum\{P(W_i) \mid W_i \in W, t \in Q(W_i)\}$.

- $W_1 : P(W_1) = 0.1$
  - SSN: 185, N: Smith, M: 1
  - SSN: 185, N: Brown, M: 1

- $W_2 : P(W_2) = 0.1$
  - SSN: 185, N: Smith, M: 1
  - SSN: 185, N: Brown, M: 2

- $W_3 : P(W_3) = 0.1$
  - SSN: 185, N: Smith, M: 1
  - SSN: 185, N: Brown, M: 3

- $W_4 : P(W_4) = 0.1$
  - SSN: 185, N: Smith, M: 1
  - SSN: 185, N: Brown, M: 4

- $W_5 : P(W_5) = 0.3$
  - SSN: 185, N: Smith, M: 1
  - SSN: 186, N: Brown, M: 1

- $W_6 : P(W_6) = 0.3$
  - SSN: 185, N: Smith, M: 1
  - SSN: 186, N: Brown, M: 2
Given query \( Q \) and probabilistic database \((W, P)\): The Marginal Probability of an answer \( t \) is: 
\[
P(t) = \sum \{P(W_i) \mid W_i \in W, t \in Q(W_i)\}.
\]

Let \( W = \{W_1, \ldots, W_6\} \).

- \( \exists N \exists M \text{Census}(S, N, M) \):
  
  \[
P(185) = 1 \text{ and } P(186) = 0.6.
  \]

- \( \exists S \exists M \text{Census}(S, N, M) \):
  
  \[
P(\text{Smith}) = P(\text{Brown}) = 1.
  \]

These are trivial queries!

Computing the marginal probability is hard in general!
Given query $Q$ and probabilistic database $(\mathcal{W}, P)$: The Marginal Probability of an answer $t$ is: 
$$P(t) = \sum\{P(W_i) \mid W_i \in \mathcal{W}, t \in Q(W_i)\}.$$ 

Let $\mathcal{W} = \{W_1, \ldots, W_6\}$. 

- $\exists N \exists M \text{Census}(S, N, M)$: 
  - $P(185) = 1$ and 
  - $P(186) = 0.6$. 

- $\exists S \exists M \text{Census}(S, N, M)$: 
  - $P(\text{Smith}) = P(\text{Brown}) = 1$. 

These are trivial queries! Computing the marginal probability is hard in general!

$\rightarrow$ Key challenge: Which queries admit efficient (polynomial time) computation of marginal probabilities for their answers?
For a given query language $Q$ and data model $W$:
For any query $Q \in Q$ and database $W \in W$, is there $\overline{Q} \in Q$ such that $\overline{Q}(W) = \{ Q(W_i) \mid W_i \in W \}$ and can be represented in $W$?

\[ W \xrightarrow{Q} \overline{Q}(W) \]

\[ \{W_1, \ldots, W_n\} \xrightarrow{Q} \{ Q(W_1), \ldots, Q(W_n) \} \]

- This holds for relational algebra and PC databases: [Imielinski’84]
  $\overline{Q}(T)$ is an extension of $Q$ to also compute the query lineage.

- This does not hold for BIDs and TIs, but query lineage still useful for computing marginal probabilities of query answers on BIDs and TIs.

- This idea is also used by Trio and MayBMS. [Das Sarma’06, Antova’08]
Query Lineage by Example

<table>
<thead>
<tr>
<th>Customer</th>
<th>Orders</th>
<th>Lineitem</th>
</tr>
</thead>
<tbody>
<tr>
<td>ckey</td>
<td>name</td>
<td>Φ</td>
</tr>
<tr>
<td>1</td>
<td>Joe</td>
<td>x₁</td>
</tr>
<tr>
<td>2</td>
<td>Dan</td>
<td>x₂</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1994-11-11</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>2</td>
</tr>
</tbody>
</table>

Query asking for the dates of discounted orders shipped to customer 'Joe':

\[ \exists C \exists O \exists D \text{Customer}(C, \text{Joe}), \text{Orders}(O, C, D), \text{Lineitem}(O, S, C), S > 0 \]

<table>
<thead>
<tr>
<th>odate</th>
<th>Φ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995-01-10</td>
<td>x₁y₁z₁ + x₁y₁z₂</td>
</tr>
</tbody>
</table>

\[ Q \] does \[ Q \] and propagates the input conditions \[ Φ \] to the answers:

- join of tuples leads to conjunction of their conditions
- union/disjunction of tuples leads to disjunction of their conditions.

Query lineage traces the computation of an answer back to its input.
Marginal Probabilities via Query Lineage

The marginal probability of a query answer is the probability of its lineage.

How to compute the lineage probability?

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$y_1$</th>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$x_1 y_1 z_1 + x_1 y_1 z_2$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$P(x_1) \cdot P(y_1) \cdot (1 - P(z_1)) \cdot P(z_2)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$P(x_1) \cdot P(y_1) \cdot P(z_1) \cdot (1 - P(z_2))$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$P(x_1) \cdot P(y_1) \cdot P(z_1) \cdot P(z_2)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$$P(x_1 y_1 z_1 + x_1 y_1 z_2) = P(x_1) \cdot P(y_1) \cdot [1 - (1 - P(z_1)(1 - P(z_2)))].$$

- Going over its truth table is exponential in the number of variables.

Two ideas:  

- Read-once lineage factorization  
  $$x_1 y_1 z_1 + x_1 y_1 z_2 = x_1 y_1 (z_1 + z_2)$$

- Lineage compilation into polysize decision diagrams.
Where Are We Now?

- We know how to compute the query answers using a simple query extension that also computes the query lineage.

- We do not know yet how to compute the marginal probabilities of query answers efficiently.

Next part of the tutorial:

- Analyze the complexity of computing marginal probabilities as a function of database size and query structure.
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem

Dichotomies for Query Evaluation

• The Hard Queries
• The Tractable Queries

Ranking Queries
Next Steps
References
Short Recap on Complexity Class $\#P$ (Sharp P)

$\#P = \text{Class of functions } f(x) \text{ for which there exists a PTIME non-deterministic Turing machine } M \text{ such that } f(x) = \text{number of accepting computations of } M \text{ on input } x.$  

Class of **counting problems** associated with decision problems in NP:

- SAT (given formula $\phi$, is $\phi$ satisfiable?) is NP-complete
- $\#$SAT (given formula $\phi$, count # of satisfying assignments) is $\#P$-complete

A PTIME machine with a $\#P$ oracle can solve any problem in polynomial hierarchy with one $\#P$ query.  

$\#$SAT is $\#P$-complete already for bipartite positive DNFs!  

- .. yet SAT is trivially PTIME for DNFs.
The following property has been observed for several classes \( \mathcal{Q} \) of relational queries on TI databases:

The data complexity of every query in \( \mathcal{Q} \) is either **polynomial time** or **\#P-hard**.
The following property has been observed for several classes $Q$ of relational queries on TI databases:

The data complexity of every query in $Q$ is either polynomial time or $\#P$-hard.

Examples of such classes $Q$ of relational queries:

- NCQ: non-repeating conjunctive queries [Dalvi’07]
- NCQs under functional dependencies [O.’09]
- Quantified queries (division, set comparisons) [Fink’11]
- UCQ: unions of conjunctive queries [Dalvi’12]
- RNCQ: ranking NCQ [O.’12]
- 1RA: NCQ’s relational algebra counterpart extended with negation [Fink’16]
Syntactic Characterizations of Tractable Queries

The tractable queries in (R)NCQ and 1RA\(^-\) admit an efficient syntactic characterization via the *hierarchical* property.

A (Boolean) NCQ or 1RA\(^-\) query \(Q\) is *hierarchical* if:

For every pair of distinct variables \(A\) and \(B\) in \(Q\),
there is no triple of relation symbols \(R\), \(S\), and \(T\) in \(Q\) such that:

- \(R^{A\rightarrow B}\) has query variable \(A\) and not \(B\),
- \(S^{AB}\) has both query variables \(A\) and \(B\), and
- \(T^{\neg AB}\) has query variable \(B\) and not in \(A\).
Examples

Non-hierarchical queries:

- $\exists_A \exists_B [R(A) \land S(A, B) \land T(B)]$
- $\exists_B [\exists_A (R(A) \land S(A, B)) \land \neg T(B)]$
- $\exists_B [T(B) \land \neg \exists_A (R(A) \land S(A, B))]$
Examples

Non-hierarchical queries:

- $\exists_A \exists_B [R(A) \land S(A, B) \land T(B)]$
- $\exists_B [\exists_A (R(A) \land S(A, B)) \land \neg T(B)]$
- $\exists_B [T(B) \land \neg \exists_A (R(A) \land S(A, B))]$
Examples

Hierarchical queries:

- $\exists A \exists B \left[ (R(A) \land S(A, B)) \land \neg T(A, B) \right]$
- $\exists A \exists B \left[ (R(A) \land T(B)) \land \neg (U(A) \land V(B)) \right]$
- $\exists A \exists B \left[ (M(A) \land N(B)) \land \neg \left[ (R(A) \land T(B)) \land \neg (U(A) \land V(B)) \right] \right]$
Examples

Hierarchical queries:

- $\exists_A \exists_B \left[ (R(A) \land S(A, B)) \land \neg T(A, B) \right]

- $\exists_A \exists_B \left[ (R(A) \land T(B)) \land \neg (U(A) \land V(B)) \right]

- $\exists_A \exists_B \left[ (M(A) \land N(B)) \land \neg \left[ (R(A) \land T(B)) \land \neg (U(A) \land V(B)) \right] \right]$
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem
Dichotomies for Query Evaluation

- The Hard Queries
- The Tractable Queries

Ranking Queries
Next Steps
References
Hardness Proof Idea

Reduction from \( \#P \)-hard model counting problem for positive bipartite DNF:

- Given a non-hierarchical 1RA\(^-\) query \( Q \) and

- Any positive bipartite DNF formula \( \Psi \) over disjoint sets \( X \) and \( Y \) of random variables.

- \( \#\Psi \) can be computed using linearly (in most cases constantly) many calls to an oracle for \( P(Q) \), where \( Q \) is evaluated on tuple-independent databases with sizes polynomial in the size of \( \Psi \).
Simplest Example of Hardness Reduction

[Grädel’98, Dalvi’07]

Input formula and query:

- $\Psi = x_1 y_1 \lor x_1 y_2 \lor x_2 y_1$ over sets $X = \{x_1, x_2\}, Y = \{y_1, y_2\}$
- $Q = \exists_A \exists_B [R(A) \land S(A, B) \land T(B)]$

Construct a TI database $D$ such that $\Psi$ annotates $Q(D)$:

- Column $\Phi$ holds random variables in $\Psi$.
  - Notation: $\top$ (true)
- Variables also used as constants for $A$ and $B$.
- $S(x_i, y_j, \top)$: $x_i y_j$ is a clause in $\Psi$.
- $R(x_i, x_i)$ and $T(y_j, y_j)$: $x_i$ is a variable in $X$ and $y_j$ is a variable in $Y$.

<table>
<thead>
<tr>
<th>$R$</th>
<th>$T$</th>
<th>$S$</th>
<th>$R \land S \land T$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\Phi$</td>
<td>$B$</td>
<td>$\Phi$</td>
<td>$\top$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_1$</td>
<td>$y_1$</td>
<td>$y_1$</td>
<td>$x_1 y_1$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_2$</td>
<td>$y_2$</td>
<td>$y_2$</td>
<td>$x_2 y_2$</td>
</tr>
</tbody>
</table>

Query $Q$ is the only minimal hard pattern in case of queries without negation!
Simplest Example of Hardness Reduction

[Grädel’98, Dalvi’07]

Input formula and query:

- $\Psi = x_1y_1 \lor x_1y_2 \lor x_2y_1$ over sets $X = \{x_1, x_2\}, Y = \{y_1, y_2\}$
- $Q = \exists_A \exists_B \left[ R(A) \land S(A, B) \land T(B) \right]$.

Construct a TI database $D$ such that $\Psi$ annotates $Q(D)$:

- Column $\Phi$ holds random variables in $\Psi$.
  - Notation: $\top$ (true)
- Variables also used as constants for $A$ and $B$.
- $S(x_i, y_j, \top)$: $x_iy_j$ is a clause in $\Psi$.
- $R(x_i, x_i)$ and $T(y_j, y_j)$: $x_i$ is a variable in $X$ and $y_j$ is a variable in $Y$.

<table>
<thead>
<tr>
<th>$R$</th>
<th>$T$</th>
<th>$S$</th>
<th>$R \land S \land T$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\Phi$</td>
<td>$B$</td>
<td>$\Phi$</td>
<td>$A$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_1$</td>
<td>$y_1$</td>
<td>$y_1$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_2$</td>
<td>$y_2$</td>
<td>$y_2$</td>
<td>$x_2$</td>
</tr>
</tbody>
</table>

Query $Q$ is the only minimal hard pattern in case of queries without negation!
A Surprising Example of Hardness Reduction

Input formula and query: \[\text{[Fink'16]}\]

- \(\Psi = x_1y_1 \lor x_1y_2\) over sets \(X = \{x_1\}, Y = \{y_1, y_2\}\)
- \(Q = \exists_A [R(A) \land \lnot \exists_B (T(B) \land S(A, B))]\)

Construct a TI database \(D\) such that \(\Psi\) annotates \(Q(D)\):

- \(S(i, b, \top)\): Clause \(i\) in \(\Psi\) has variable \(b\).
- \(R(i, \top)\) and \(T(b, \lnot b)\): \(i\) is a clause and \(b\) is a variable in \(\Psi\).

<table>
<thead>
<tr>
<th>(R)</th>
<th>(A \Phi)</th>
<th>(T)</th>
<th>(B \Phi)</th>
<th>(S)</th>
<th>(A B \Phi)</th>
<th>(T \land S)</th>
<th>(A B \Phi)</th>
<th>(\exists_B (T \land S))</th>
<th>(A \Phi)</th>
<th>(R \land \lnot \exists_B (T \land S))</th>
<th>(A \Phi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (\top)</td>
<td>(x_1 \lnot x_1)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td></td>
</tr>
<tr>
<td>2 (\top)</td>
<td>(y_1 \lnot y_1)</td>
<td>(1 \top)</td>
<td>(1 \top)</td>
<td>(2 \top)</td>
<td>(1 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td></td>
</tr>
<tr>
<td>(y_2 \lnot y_2)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td>(2 \top)</td>
<td></td>
</tr>
</tbody>
</table>

Query \(Q\) is already hard when \(T\) is the only uncertain input relation!
A Surprising Example of Hardness Reduction

Input formula and query: [Fink’16]

- $\Psi = x_1y_1 \lor x_1y_2$ over sets $X = \{x_1\}$, $Y = \{y_1, y_2\}$
- $Q = \exists_A[R(A) \land \neg\exists_B(T(B) \land S(A, B))]$

Construct a TI database $D$ such that $\Psi$ annotates $Q(D)$:

- $S(i, b, \top)$: Clause $i$ in $\Psi$ has variable $b$.
- $R(i, \top)$ and $T(b, \neg b)$: $i$ is a clause and $b$ is a variable in $\Psi$.

<table>
<thead>
<tr>
<th>$R$</th>
<th>$T$</th>
<th>$S$</th>
<th>$T \land S$</th>
<th>$\exists_B(T \land S)$</th>
<th>$R \land \neg\exists_B(T \land S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \phi$</td>
<td>$B \phi$</td>
<td>$A B \phi$</td>
<td>$A B \phi$</td>
<td>$A \phi$</td>
<td>$A \phi$</td>
</tr>
<tr>
<td>1 $\top$</td>
<td>$x_1 \neg x_1$</td>
<td>1 $x_1 \top$</td>
<td>1 $x_1 \neg x_1$</td>
<td>1 $\neg x_1 \lor \neg y_1$</td>
<td>1 $x_1 y_1$</td>
</tr>
<tr>
<td>2 $\top$</td>
<td>$y_1 \neg y_1$</td>
<td>1 $y_1 \top$</td>
<td>1 $y_1 \neg y_1$</td>
<td>2 $\neg x_1 \lor \neg y_2$</td>
<td>2 $x_1 y_2$</td>
</tr>
<tr>
<td></td>
<td>$y_2 \neg y_2$</td>
<td>2 $x_1 \top$</td>
<td>2 $x_1 \neg x_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 $y_2 \top$</td>
<td>2 $y_2 \neg y_2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query $Q$ is already hard when $T$ is the only uncertain input relation!
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem
Dichotomies for Query Evaluation
  • The Hard Queries
  • The Tractable Queries
Ranking Queries
Next Steps
References
Evaluation of Hierarchical 1RA⁻ Queries

Approach based on knowledge compilation

- For any TI database \( D \), the probability \( P_{Q(D)} \) of a 1RA⁻ query \( Q \) is the probability \( P_\Psi \) of the query lineage \( \Psi \).
- Compile \( \Psi \) into poly-size OBDD(\( \Psi \)).
- Compute probability of OBDD(\( \Psi \)) in time linear in its size.
Evaluation of Hierarchical $1\text{RA}^-$ Queries

Approach based on knowledge compilation

- For any TI database $D$, the probability $P_{Q(D)}$ of a $1\text{RA}^-$ query $Q$ is the probability $P_{\Psi}$ of the query lineage $\Psi$.
- Compile $\Psi$ into poly-size OBDD($\Psi$).
- Compute probability of OBDD($\Psi$) in time linear in its size.

Lineage of tractable $1\text{RA}^-$ queries:

- **Read-once** for queries without negation (so NCQ) [O.’08b]
  It admits linear-size OBBDs.

- **Not** read-once for queries with negation [Fink’16]
  - It admits OBBDs of size linear in the database size
    but exponential in the query size.
The Inner Workings

From hierarchical 1RA$^-$ to RC-hierarchical $\exists$-consistent RC$^\exists$:

- Translate query $Q$ into an equivalent disjunction of disjunction-free existential relational calculus queries $Q_1 \lor \cdots \lor Q_k$.

- **RC-hierarchical:**
  For each $\exists_X(Q')$, every relation symbol in $Q'$ has variable $X$.
  - Each of the disjuncts gives rise to a poly-size OBDD.

- **$\exists$-consistent:**
  The nesting order of the quantifiers is the same in $Q_1, \cdots, Q_k$.
  - All OBDDs have compatible variable orders and their disjunction is a poly-size OBDD.

- The OBDD width grows exponentially with $k$, its height stays linear in the size of the database.
  - Width = maximum number of edges crossing the section between any two consecutive levels.

Similar ideas used for the evaluation of inversion-free UCQs. [Jha’13]
Consider the following query and TI database:

\[ \exists_A \exists_B \left[ (R(A) \land T(B)) \land \neg(U(A) \land V(B)) \right] \]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(R)</td>
<td>(T)</td>
<td>(U)</td>
<td>(V)</td>
<td>(R \land T)</td>
</tr>
<tr>
<td>(A \ \Phi)</td>
<td>(\Phi)</td>
<td>(B \ \Phi)</td>
<td>(\Phi)</td>
<td>(\Phi)</td>
<td>(\Phi)</td>
</tr>
<tr>
<td>1 r₁</td>
<td>1 t₁</td>
<td>1 u₁</td>
<td>1 v₁</td>
<td>1 1 r₁ t₁</td>
<td>1 1 r₁ t₁ \neg(u₁ v₁)</td>
</tr>
<tr>
<td>2 r₂</td>
<td>2 t₂</td>
<td>2 u₂</td>
<td>2 v₂</td>
<td>1 2 r₁ t₂</td>
<td>1 2 r₁ t₂ \neg(u₁ v₂)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 1 r₂ t₁</td>
<td>2 1 r₂ t₁ \neg(u₂ v₁)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 2 r₂ t₂</td>
<td>2 2 r₂ t₂ \neg(u₂ v₂)</td>
</tr>
</tbody>
</table>
Consider the following query and TI database:

\[ Q = \exists_A \exists_B \left[ (R(A) \land T(B)) \land \neg (U(A) \land V(B)) \right] \]

<table>
<thead>
<tr>
<th>R</th>
<th>T</th>
<th>U</th>
<th>V</th>
<th>R \land T</th>
<th>R \land T \land \neg(U \land V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ⦬</td>
<td>B ⦬</td>
<td>A ⦬</td>
<td>B ⦬</td>
<td>A B ⦬</td>
<td>1 1 r1t1 ¬(u1v1)</td>
</tr>
<tr>
<td>1 r1</td>
<td>1 t1</td>
<td>1 u1</td>
<td>1 v1</td>
<td>1 1 r1t1</td>
<td>1 2 r1t2 ¬(u1v2)</td>
</tr>
<tr>
<td>2 r2</td>
<td>2 t2</td>
<td>2 u2</td>
<td>2 v2</td>
<td>2 1 r2t1</td>
<td>2 1 r2t1 ¬(u2v1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 2 r2t2</td>
<td>2 2 r2t2 ¬(u2v2)</td>
</tr>
</tbody>
</table>

The lineage of \( Q \) is:

\[ \Psi = r_1 [t_1 (\neg u_1 \lor \neg v_1) \lor t_2 (\neg u_1 \lor \neg v_2)] \lor r_2 [t_1 (\neg u_2 \lor \neg v_1) \lor t_2 (\neg u_2 \lor \neg v_2)]. \]

- Variables entangle in \( \Psi \) beyond read-once factorization.
- This is the pivotal intricacy introduced by negation.
Query Evaluation Example (2/3)

Translate $Q = \exists_A \exists_B \left[ (R(A) \land T(B)) \land \neg(U(A) \land V(B)) \right]$ into RC $^3$:

$$Q_{RC} = \exists_A (R(A) \land \neg U(A)) \land \exists_B T(B) \lor \exists_A R(A) \land \exists_B (T(B) \land \neg V(B)) .$$

- Both $Q_1$ and $Q_2$ are RC-hierarchical.
- $Q_1 \lor Q_2$ is $3$-consistent: Same order $\exists_A \exists_B$ for $Q_1$ and $Q_2$.

Query annotation:

$$\Psi = (r_1 \lor r_2) \lor (t_1 \lor t_2) \lor (r_1 \lor r_2) \lor (t_1 \lor t_2) .$$

- Both $\Psi_1$ and $\Psi_2$ admit linear-size OBDDs.
- The two OBDDs have compatible orders and their disjunction is an OBDD whose width is the product of the widths of the two OBDDs.
Query Evaluation Example (3/3)

Compile query annotation into OBDD:

\[
\Psi = (r_1 \lnot u_1 \lor r_2 \lnot u_2) \land (t_1 \lor t_2) \lor (r_1 \lor r_2) \land (t_1 \lnot v_1 \lor t_2 \lnot v_2).
\]

Diagram:

- \(r_1\) -> \(\lnot u_1\) -> \(t_1\) -> \(\top\)
- \(r_2\) -> \(\lnot u_2\) -> \(t_1\) -> \(\top\)
- \(r_2\) -> \(\lnot u_2\) -> \(t_2\) -> \(\top\)
- \(r_1\) -> \(\lnot v_1\) -> \(t_1\) -> \(\top\)
- \(r_2\) -> \(\lnot v_2\) -> \(t_2\) -> \(\top\)

\(\Psi_1\)  \lor  \(\Psi_2\)  =  \(\Psi\)
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem
Dichotomies for Query Evaluation
- The Hard Queries
- The Tractable Queries
Ranking Queries
Next Steps
References
Given a NCQ query $Q$, a TI database $D$, and any two answers $t_1, t_2 \in Q(D)$, does $P(t_1) \leq P(t_2)$ hold?

Motivation:

- Probabilities are mere degrees of uncertainty in the data and are not otherwise meaningful to the user.

- Users mostly care about the ranking of answers in decreasing order of their probabilities or about a few most likely answers.
Two complementary observations

1. Probability computation for distinct answers **may share a common factor**
   - That can be computed only once
     - Save computation time for both query evaluation and ranking!
   - Or that can be uniformly ignored for all answers.
     - For ranking purposes, we may ignore computationally hard tasks!

Ranking is computationally easier than query evaluation.
Two complementary observations

1. Probability computation for distinct answers may share a common factor
   ▶ That can be computed only once
     ▶ Save computation time for both query evaluation and ranking!
   ▶ Or that can be uniformly ignored for all answers.
     ▶ For ranking purposes, we may ignore computationally hard tasks!

   Ranking is computationally easier than query evaluation.

2. To compute the exact ranking of query answers, approximate probabilities of the individual answers may suffice.
   ▶ Compute lower and upper bounds on these probabilities.
   ▶ Incrementally refine the bounds to the extent needed to rank the answers.
Share Query Plans and Anytime Approximation

Approach with two main ingredients [O.’12]

1. Share query plans to detect factors common to query answers
   ▶ Static analysis on the query structure to identify subqueries whose computation can be shared across distinct query answers.
   ▶ Equivalently, they identify factors shared by lineage of query answers.

2. Ranking based on anytime deterministic approximate inference
   ▶ Incremental compilation of lineage with shared factors into BDDs
   ▶ Each compilation step refines lower and upper bounds on lineage probabilities
Share Query Plans and Anytime Approximation

Approach with two main ingredients [O.’12]

1. Share query plans to detect factors common to query answers
   - Static analysis on the query structure to identify subqueries whose computation can be shared across distinct query answers.
   - Equivalently, they identify factors shared by lineage of query answers.

2. Ranking based on anytime deterministic approximate inference
   - Incremental compilation of lineage with shared factors into BDDs
   - Each compilation step refines lower and upper bounds on lineage probabilities

Alternative approach using FPRAS-based Monte Carlo [Ré’07]

- Ranking with probabilistic guarantee only
- Not truly incremental
- Black box approach, structure and common factors of query lineage not exploited.
Example

List topics posted by users who have mentioned their followers:

\[ Q(X) = \exists Y \exists Z \exists U \text{Trends}(X, Y), \text{Follows}(Y, Z), \text{Mentions}(U, Y, Z), \text{Tweets}(U, Y). \]

(User Y contributed to trendy topic X, user Y follows user Z, user Y mentions user Z in tweet U, tweet U of user Y.)

A share plan for Q is as follows

```
      Y
     / \  
   Trends(X, *)  Follows(*, Z), Mentions(U, *, Z), Tweets(U, *)
```

and corresponds to the following rewriting:

\[ Q(X) = \text{Trends}(X, Y), Q'(Y) \]
\[ Q'(Y) = \text{Follows}(Y, Z), \text{Mentions}(U, Y, Z), \text{Tweets}(U, Y) \]

- Several answers (X-values) can be paired with the same value y of the variable Y and thus share the lineage Q'(y).
- For any value y, the query Q'(y) is non-hierarchical and thus #P-hard!
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem
Dichotomies for Query Evaluation
  ● The Hard Queries
  ● The Tractable Queries
Ranking Queries
Next Steps
References
Next Steps

- **Declarative Probabilistic Programming with Datalog**: Probabilities become first-class citizens in the query language.
  - PPDL, semantics given by a notion of probabilistic chase [Bárány’16]
  - Incorporating ontologies
  - Vast literature (including MLNs) but missing the declarativity aspect!
Next Steps

- **Declarative Probabilistic Programming with Datalog**: Probabilities become first-class citizens in the query language.
  - PPDL, semantics given by a notion of probabilistic chase [Bárány’16]
  - Incorporating ontologies
  - Vast literature (including MLNs) but missing the declarativity aspect!

- Further push the barrier on complexity
  - See Dan Suciu’s advanced lecture on lifted inference!
  - Understand tractability for probabilistic programs at large
Next Steps

- **Declarative Probabilistic Programming with Datalog:** Probabilities become first-class citizens in the query language.
  - PPDL, semantics given by a notion of probabilistic chase [Bárány’16]
  - Incorporating ontologies
  - Vast literature (including MLNs) but missing the declarativity aspect!

- Further push the barrier on complexity
  - See Dan Suciu’s advanced lecture on lifted inference!
  - Understand tractability for probabilistic programs at large

- Tractability not sufficient in practice, develop approximations with predictable performance
Next Steps

- **Declarative** Probabilistic Programming with Datalog: Probabilities become first-class citizens in the query language.
  - PPDL, semantics given by a notion of probabilistic chase [Bárány’16]
  - Incorporating ontologies
  - Vast literature (including MLNs) but missing the declarativity aspect!

- Further push the barrier on complexity
  - See Dan Suciu’s advanced lecture on lifted inference!
  - Understand tractability for probabilistic programs at large

- Tractability not sufficient in practice, develop approximations with predictable performance

- This tutorial assumed CWA. What about OWA, e.g., Google squares? See Guy van den Broeck et al’s talk on open-world probabilistic databases!
Next Steps

- **Declarative** Probabilistic Programming with Datalog: Probabilities become first-class citizens in the query language.
  - PPDL, semantics given by a notion of probabilistic chase [Bárány’16]
  - Incorporating ontologies
  - Vast literature (including MLNs) but missing the declarativity aspect!

- Further push the barrier on complexity
  - See Dan Suciu’s advanced lecture on lifted inference!
  - Understand tractability for probabilistic programs at large

- Tractability not sufficient in practice, develop approximations with predictable performance

- This tutorial assumed CWA. What about OWA, e.g., Google squares?
  - See Guy van den Broeck et al’s talk on open-world probabilistic databases!

- Build open-source systems, provide benchmarks
Outline

Why Probabilistic Databases?
Probabilistic Data Models
The Query Evaluation Problem
Dichotomies for Query Evaluation
  • The Hard Queries
  • The Tractable Queries
Ranking Queries
Next Steps
References
References: Some Probabilistic Database Systems (1/3)

- **Stanford: Trio**

- **UW: MystiQ**

- **Cornell & Oxford: MayBMS & SPROUT**
  Jiewen Huang, Lyublena Antova, Christoph Koch, Dan Olteanu. *MayBMS: a probabilistic database management system*. SIGMOD 2009: 1071-1074

IBM Almaden & Rice: MCDB  

LogicBlox & Technion & Oxford: PPDL  

Maryland: PrDB  
UC Berkeley: Bayestore

Purdue: Orion

Waterloo: Probabilistic ranking

many, many others..
References: Applications (1/2)

- **Google Knowledge Vault**
  Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, Wei Zhang: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. KDD 2014: 601-610

- **DeepDive**

- **NELL**

- **Census data**
  [Antova’07](#) Lyublena Antova, Christoph Koch, Dan Olteanu. 10^{10^6} Worlds and Beyond: Efficient Representation and Processing of Incomplete Information. ICDE 2007: 606-615
References: Applications (2/2)

- Creating probabilistic databases
  


- OCR with Staccato
  

- Possible data repairs
  

- Querying open-world Google squares
  
Or-sets


World-set decompositions


Block-independent disjoint, x-relations, x-tables


References: Data Models (1/2)

- Conditional databases (c-tables)
  

- Trio's ULDBs
  
  \[\textbf{Benjelloun'06}\]: Omar Benjelloun, Anish Das Sarma, Alon Halevy, and Jennifer Widom. *ULDBs: databases with uncertainty and lineage*. In Proc. 32nd Int. Conf. on Very large Data Bases, pages 953-964, 2006

- MayBMS’ U-relations
  
  \[\textbf{Antova'08}\] Lyublena Antova, Thomas Jansen, Christoph Koch, Dan Olteanu. *Fast and Simple Relational Processing of Uncertain Data*. ICDE 2008: 983-992

- Markov Logic Networks via queries over TI databases
  
References: Query Evaluation (1/2)


- Possible/certain answers in conditional databases (c-tables)
  


- Representability of query answers, query lineage
  


  [Antova’08] Lyublena Antova, Thomas Jansen, Christoph Koch, Dan Olteanu. *Fast and Simple Relational Processing of Uncertain Data*. ICDE 2008: 983-992
Factorization of query lineage


Ranking query answers


References: Complexity of Probabilistic Query Evaluation

- Complexity class $\#P$
  


- Hardness results
  

References: Dichotomies for Probabilistic Query Evaluation

- Non-repeating CQ


- Union of conjunctive queries (UCQs)

- Non-repeating conjunctive relational algebra (with negation)
References: Futher References

- Conditioning probabilistic databases
  

- Probabilistic XML
  


- Beyond relational queries
  
Thank you!