A Dichotomy

for Queries with **Negation**

in **Probabilistic** Databases

---

**Dan Olteanu**

Joint work with Robert Fink

*Uncertainty in Computation*

Simons Institute for the Theory of Computing

Berkeley  Oct 5, 2016
Outline

Probabilistic Databases 101

The Dichotomy

The Interesting but Hard Queries

The Easy Queries

Leftovers
Tuple-Independent Probabilistic Databases

Tuple-independent database of $n$ tuples $\(t_i\)_{i \in [n]}$:

- Each tuple $t_i$ associated with an independent Boolean random variable $x_i$.
- $P(x_i = \text{true})$ gives the probability that $t_i$ exists in the database.

Possible-worlds semantics:

- Each possible world defined by an assignment $\theta$ of the variables $\(x_i\)_{i \in [n]}$:
  - It consists of all tuples $t_i$ for which $\theta(x_i) = \text{true}$.
  - It has probability $P(\theta) = \prod_{i \in [n]} P(x_i = \theta(x_i))$.

- A tuple-independent database with $n$ tuples has $2^n$ possible worlds.
Relational Algebra

Popular database query language since Codd times.

- Algebra carrier: set of all finite relations
- Algebra operations: $\pi$ (projection), $\times$ (Cartesian product), $-$ (set difference), $\Join$ (join), $\sigma$ (selection), $\cup$ (set union), $\delta$ (renaming)
- As expressive as domain relational calculus (RC)

In this talk: Relational algebra fragment 1RA$^-$

- Included: Equality joins, selections, projections, difference
- Excluded: Repeating relation symbols, unions
Relational Algebra

Popular database query language since Codd times.

- Algebra carrier: set of all finite relations
- Algebra operations: $\pi$ (projection), $\times$ (Cartesian product), $\neg$ (set difference), $\Join$ (join), $\sigma$ (selection), $\cup$ (set union), $\delta$ (renaming)
- As expressive as domain relational calculus (RC)

In this talk: Relational algebra fragment 1RA$^-$

- Included: Equality joins, selections, projections, difference
- Excluded: Repeating relation symbols, unions

Examples of (Boolean) 1RA$^-$ queries:

- Are there combinations of tuples in $(R, T)$ that are not in $(U, V)$?

  $\pi_\emptyset [(R(A) \times T(B)) \neg (U(A) \times V(B))]$
  $\exists_A \exists_B [(R(A) \land T(B)) \land \neg (U(A) \land V(B))]$ (in RC)

- Does relation $S$ “hold hands” with both $R$ and $T$?

  $\pi_\emptyset [R(A) \Join S(A,B) \Join T(B)]$
  $\exists_A \exists_B [R(A) \land S(A,B) \land T(B)]$ (in RC)
The Query Evaluation Problem

For any Boolean 1RA− query $Q$ and tuple-independent database $D$:

Compute the probability that $Q$ is true in a random world of $D$.

The case of non-Boolean queries can be reduced to the Boolean case.

We are interested in the data complexity of this problem.

- Fix the query $Q$ and take the database $D$ as input.
Outline

Probabilistic Databases 101

The Dichotomy

The Interesting but Hard Queries

The Easy Queries

Leftovers
Data complexity of any 1RA- query $Q$ in tuple-independent databases:

- Polynomial time if $Q$ is hierarchical and \#P-hard otherwise.
Hierarchical 1RA\(^{-}\) Queries

(Boolean) 1RA\(^{-}\) query \(Q\) is *hierarchical* if

- For every pair of distinct query variables \(A\) and \(B\) in \(Q\),
- there is no triple of relation symbols \(R\), \(S\), and \(T\) in \(Q\) such that:
- \(R\) has \(A\) but not \(B\), \(S\) has both \(A\) and \(B\), and \(T\) has \(B\) but not \(A\).
Hierarchical \(1\text{RA}^-\) Queries

(Boolean) \(1\text{RA}^-\) query \(Q\) is \textit{hierarchical} if

- For every pair of distinct query variables \(A\) and \(B\) in \(Q\),
- there is no triple of relation symbols \(R, S,\) and \(T\) in \(Q\) such that:
- \(R\) has \(A\) but not \(B\), \(S\) has both \(A\) and \(B\), and \(T\) has \(B\) but not \(A\).

\[
\begin{align*}
R(A) & \quad S(A, B) & \quad T(B)
\end{align*}
\]
(Boolean) $\text{1RA}^-$ query $Q$ is *hierarchical* if

- For every pair of distinct query variables $A$ and $B$ in $Q$,
- there is no triple of relation symbols $R$, $S$, and $T$ in $Q$ such that:
- $R$ has $A$ but not $B$, $S$ has both $A$ and $B$, and $T$ has $B$ but not $A$. 
Hierarchical $1\text{RA}^-$ Queries

(Boolean) $1\text{RA}^-$ query $Q$ is \textit{hierarchical} if

- For every pair of distinct query variables $A$ and $B$ in $Q$,
- there is no triple of relation symbols $R$, $S$, and $T$ in $Q$ such that:
  - $R$ has $A$ but not $B$,
  - $S$ has both $A$ and $B$,
  - $T$ has $B$ but not $A$. 
Examples

Hierarchical queries:

- $\pi_\emptyset[(R(A) \boxtimes S(A, B)) - T(A, B)]$
- $\pi_\emptyset[(R(A) \times T(B)) - (U(A) \times V(B)))]$
- $\pi_\emptyset[(M(A) \times N(B)) - [(R(A) \times T(B)) - (U(A) \times V(B))]]$
- $\pi_\emptyset[\pi_A[M(A) \times N(B)] - \pi_A[(R(A) \times T(B)) - (U(A) \times V(B))]]$
Examples

Hierarchical queries:

- $\pi_\emptyset[(R(A) \times S(A, B)) - T(A, B)]$
- $\pi_\emptyset[(R(A) \times T(B)) - (U(A) \times V(B))]$
- $\pi_\emptyset[(M(A) \times N(B)) - [(R(A) \times T(B)) - (U(A) \times V(B))]]$
- $\pi_\emptyset[\pi_A[M(A) \times N(B)] - \pi_A[(R(A) \times T(B)) - (U(A) \times V(B))]]$

Non-hierarchical queries:

- $\pi_\emptyset[R(A) \times S(A, B) \times T(B)]$
- $\pi_\emptyset[\pi_B(R(A) \times S(A, B)) - T(B)]$
- $\pi_\emptyset[T(B) - \pi_B(R(A) \times S(A, B))]$
- $\pi_\emptyset[X(A) \times [R(A) - \pi_A(T(B) \times S(A, B))]]$
Outline

- Probabilistic Databases 101
- The Dichotomy
- The Interesting but Hard Queries
- The Easy Queries
- Leftovers
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 many calls to an oracle for \( P(Q) \), where \( Q \) is evaluated on tuple-independent databases of sizes linear in the size of \( \Psi \).
A Simple Case

Input formula and query:

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

Construct a database $D$ such that $\Psi$ becomes the grounding of $Q$ wrt $D$:

- Column $\Phi$ holds formulas over random variables.
  - We use $\top$ for $true$ and $\bot$ for $false$.
  - 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 \Join S \Join T )</th>
<th>$\pi_\emptyset [R \Join S \Join T]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ $\Phi$</td>
<td>$B$ $\Phi$</td>
<td>$A$ $B$ $\Phi$</td>
<td>$A$ $B$ $\Phi$</td>
<td>$\emptyset$ $\Phi$</td>
</tr>
<tr>
<td>$x_1$ $x_1$</td>
<td>$y_1$ $y_1$</td>
<td>$x_1$ $y_1$ $\top$</td>
<td>$x_1$ $y_1$ $x_1 y_1$</td>
<td>( \emptyset ) $\Psi$</td>
</tr>
<tr>
<td>$x_2$ $x_2$</td>
<td>$y_2$ $y_2$</td>
<td>$x_1$ $y_2$ $\top$</td>
<td>$x_1$ $y_2$ $x_1 y_2$</td>
<td>( \emptyset ) $\Psi$</td>
</tr>
</tbody>
</table>
A Simple Case

Input formula and query:
- \( \Psi = x_1 y_1 \lor x_1 y_2 \lor x_2 y_1 \) over sets \( X = \{x_1, x_2\} \), \( Y = \{y_1, y_2\} \)
- \( Q = \pi_\emptyset \left[ R(A) \Join S(A, B) \Join T(B) \right] \)

Construct a database \( D \) such that \( \Psi \) becomes the grounding of \( Q \) wrt \( D \):
- Column \( \Phi \) holds formulas over random variables.
  ▶ We use \( \top \) for true and \( \bot \) for false.
- 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></th>
<th>( R )</th>
<th></th>
<th>( T )</th>
<th></th>
<th>( S )</th>
<th></th>
<th>( R \Join S \Join T )</th>
<th>( \pi_\emptyset \left[ R \Join S \Join T \right] )</th>
<th>( \Phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( \Phi )</td>
<td>( B )</td>
<td>( \Phi )</td>
<td>( A )</td>
<td>( B )</td>
<td>( \Phi )</td>
<td>( A )</td>
<td>( B )</td>
<td>( \Phi )</td>
</tr>
<tr>
<td>( x_1 )</td>
<td>( x_1 )</td>
<td>( y_1 )</td>
<td>( y_1 )</td>
<td>( x_1 )</td>
<td>( y_1 )</td>
<td>( \top )</td>
<td>( x_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_1 )</td>
<td>( y_2 )</td>
<td>( \top )</td>
<td>( x_2 )</td>
<td>( y_1 )</td>
<td>( x_2 y_1 )</td>
</tr>
</tbody>
</table>

This is the only minimal hard pattern for \textit{positive} \( 1\text{RA}^- \) queries!
A Surprising Case

Input formula and query:

- \( \Psi = x_1y_1 \lor x_1y_2 \) over sets \( X = \{x_1\}, Y = \{y_1, y_2\} \)
- \( Q = \pi_\emptyset \left[ R(A) - \pi_A \left( T(B) \ni S(A, B) \right) \right] \)

Construct a database \( D \) such that \( \Psi \) becomes the grounding of \( Q \) wrt \( D \):

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

<table>
<thead>
<tr>
<th>( R )</th>
<th>( T )</th>
<th>( S )</th>
<th>( T \ni S )</th>
<th>( \pi_A(T \ni S) )</th>
<th>( R - \pi_A(T \ni 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_1y_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_1y_2 )</td>
</tr>
<tr>
<td>( y_2 \ \neg y_2 )</td>
<td>2 ( x_1 \ \top )</td>
<td>2 ( x_1 \ \neg x_1 )</td>
<td>2 ( y_2 \ \neg y_2 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Surprising Case

Input formula and query:

- $\Psi = x_1 y_1 \lor x_1 y_2$ over sets $X = \{x_1\}, Y = \{y_1, y_2\}$
- $Q = \pi_{\emptyset} \left[ R(A) - \pi_A (T(B) \Join S(A, B)) \right]$

Construct a database $D$ such that $\Psi$ becomes the grounding of $Q$ wrt $D$:

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

This query is already hard when $T$ is the only probabilistic input relation!
A More Involved Case

Input formula and query:

- \( \Psi = x_1y_1 \lor x_1y_2 \lor x_2y_1 \) over sets \( X = \{x_1, x_2\}, Y = \{y_1, y_2\} \)
- \( Q = \pi_\emptyset \left[ S(A, B) - R(A) \times T(B) \right] \)

We need a different reduction gadget:

- Use additional random variables \( Z = \{z_1, \ldots, z_{|E|}\} \), one per clause in \( \Psi = \psi_1 \lor \cdots \lor \psi_{|E|} \).
- Construct a database \( D \) such that the grounding of \( Q \) wrt \( D \) is \( \neg \Upsilon = \neg \left[ \bigvee_{i=1}^{|E|} \neg z_i \neg \psi_i \right] = \bigwedge_{i=1}^{|E|} (z_i \lor \psi_i) \).

<table>
<thead>
<tr>
<th>( R )</th>
<th>( T )</th>
<th>( S )</th>
<th>( S - R \times T )</th>
<th>( \pi_\emptyset [S - R \times T] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \Phi )</td>
<td>( B \Phi )</td>
<td>( A \Phi )</td>
<td>( A \Phi )</td>
<td>( \Phi )</td>
</tr>
<tr>
<td>( x_1 \ x_1 )</td>
<td>( y_1 \ x_1 )</td>
<td>( x_1 \ y_1 \ \neg z_1 )</td>
<td>( x_1 \ y_1 \ \neg z_2 )</td>
<td>( x_1 \ y_1 \ \neg z_3 )</td>
</tr>
<tr>
<td>( x_2 \ x_2 )</td>
<td>( y_2 \ y_2 )</td>
<td>( x_1 \ y_2 \ \neg z_2 )</td>
<td>( x_1 \ y_2 \ \neg z_3 )</td>
<td>( x_1 \ y_2 \ \neg z_1 )</td>
</tr>
</tbody>
</table>

- Compute \( \#\Psi \) using linearly many calls to the oracle for \( P_Q = 1 - P(\Upsilon) \).
The Small Print (1/2)

- \( \Psi = \bigvee_{(i,j) \in E} x_i y_j = \psi_1 \lor \cdots \lor \psi_{|E|} \) over disjoint variable sets \( X \) and \( Y \)

- Let \( \Theta \) be the set of assignments of variables \( X \cup Y \) that satisfy \( \Psi \):

\[
\#\Psi = \sum_{\theta \in \Theta : \theta \models \Psi} 1.
\]

- Partition \( \Theta \) into disjoint sets \( \Theta_0 \cup \cdots \cup \Theta_{|E|} \), such that \( \theta \in \Theta_i \) if and only if \( \theta \) satisfies exactly \( i \) clauses of \( \Psi \):

\[
\#\Psi = \sum_{\theta \in \Theta_1 : \theta \models \Psi} 1 + \cdots + \sum_{\theta \in \Theta_{|E|} : \theta \models \Psi} 1 = |\Theta_1| + \cdots + |\Theta_{|E|}|.
\]

- \( |\Theta_1|, \ldots, |\Theta_{|E|}| \) can be computed using an oracle for \( P_\Upsilon \):

\[
\Upsilon = \bigvee_{i=1}^{|E|} \neg z_i \land \neg \psi_i \quad \text{or, equivalently} \quad \neg \Upsilon = \bigwedge_{i=1}^{|E|} (z_i \lor \psi_i)
\]
Express the probability of \( -\Upsilon = \bigwedge_{i=1}^{|E|}(z_i \lor \psi_i) \) as a function of \(|\Theta_1|, \ldots, |\Theta_{|E|}|\):

- Fix the probabilities of variables in \( X \cup Y \) to \(1/2\) and of variables in \( Z \) to \(p_z\). Then:

\[
P_{-\Upsilon} = \sum_{k=0}^{|E|} P\left( -\Upsilon \bigg| \text{exactly } k \text{ clauses of } \Psi \text{ are satisfied} \right) \cdot P\left( \text{exactly } k \text{ clauses of } \Psi \text{ are satisfied} \right)
\]

\[
= \frac{1}{2}^{|X|+|Y|} \sum_{k=0}^{|E|} p_z^{|E|-k} |\Theta_k|
\]

- This is a polynomial in \(p_z\) of degree \(|E|\), with coefficients \(|\Theta_0|, \ldots, |\Theta_{|E|}|\).

- The coefficients can be derived from \(|E| + 1\) pairs \((p_z, P_{\Upsilon})\) using Lagrange’s polynomial interpolation formula.

- \(|E| + 1\) oracle calls to \(P_{\Upsilon}\) suffice to determine \(\#\Psi = \sum_{i=0}^{|E|} |\Theta_i|\).
Hard Query Patterns

There are 48 (!) minimal non-hierarchical query patterns.

- Binary trees with leaves $A$, $AB$, and $B$ and inner nodes $\otimes$ or $\neg$.
  - Some are symmetric and need not be considered separately:
    $A$ and $B$ can be exchanged, joins are commutative and associative.
  - Still, many cases left to consider due to the difference operator.

- There is a database construction scheme for each pattern.
- Each non-hierarchical query $Q$ matches a pattern $P_{x,y}$. 

\[
\begin{array}{cccc}
P_{1.1} & \otimes & P_{1.2} & \otimes \\
\otimes & AB & \neg & AB \\
A & B & A & B \\
\end{array}
\]

\[
\begin{array}{cccc}
P_{1.3} & \otimes & P_{1.4} & \neg \\
\neg & AB & \neg & AB \\
A & B & A & B \\
\end{array}
\]

\[
\begin{array}{cccc}
P_{5.1} & \otimes & P_{5.2} & \otimes \\
\otimes & AB & \neg & AB \\
B & AB & B & AB \\
\end{array}
\]

\[
\begin{array}{cccc}
P_{5.3} & \neg & P_{5.4} & \neg \\
\neg & AB & \neg & AB \\
B & AB & B & AB \\
\end{array}
\]

\[
\begin{array}{cccc}
\ldots & \ldots & \ldots & \ldots \\
\end{array}
\]
There are 48 (!) minimal non-hierarchical query patterns.

- Binary trees with leaves $A$, $AB$, and $B$ and inner nodes $\otimes$ or $\ominus$.
  - Some are symmetric and need not be considered separately: $A$ and $B$ can be exchanged, joins are commutative and associative.
  - Still, many cases left to consider due to the difference operator.

There is a database construction scheme for each pattern.

Each non-hierarchical query $Q$ matches a pattern $P_{x,y}$.

In the absence of negation, $P_{1.1}$ is the only hard pattern to consider!
Non-hierarchical Queries Match Minimal Hard Patterns

Each non-hierarchical query $Q$ matches a pattern $P_{x,y}$:

- There is a total mapping from $P_{x,y}$ to $Q$'s parse tree that
  - is identity on inner nodes $\land$ and $\rightarrow$,
  - preserves ancestor-descendant relationships,
  - maps leaves to relations: $A$ to $R(A)$; $AB$ to $S(A, B)$; and $B$ to $T(B)$.

- The match “preserves” the grounding of the query pattern:
  $Q$ and $P_{x,y}$ have the same grounding for any database using our construction scheme.
Outline

Probabilistic Databases 101

The Dichotomy

The Interesting but Hard Queries

The Easy Queries

Leftovers
Evaluation of Hierarchical 1RA\(^-\) Queries

Approach based on knowledge compilation

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

Approach based on knowledge compilation

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

Distinction from existing tractability results [O. & Huang 2008]:

- $1\text{RA}^-$ without negation: Grounding formulas are read-once.
  - Read-once formulas admit linear-size OBBDs.

- $1\text{RA}^-$: Grounding formulas are not read-once.
  - They admit OBBDs of sizes linear in the database size but exponential in the query size.
The Inner Workings

From hierarchical 1RA\(^{-}\) to RC-hierarchical \(\exists\)-consistent RC\(^{3}\):

- Translate query \(Q\) into an equivalent disjunction of disjunction-free existential relational calculus queries \(Q_1 \lor \cdots \lor Q_k\).
  - \(k\) can be very large for queries with projection under difference!

- **RC-hierarchical**: For each \(\exists X (Q')\), every relation symbol in \(Q'\) has variable \(X\).
  - Each of the disjuncts yields 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.
Consider the following query and tuple-independent database:

\[
Q = \pi_\emptyset \left[ (R(A) \times T(B)) - (U(A) \times V(B)) \right]
\]

<table>
<thead>
<tr>
<th></th>
<th>( R ) ( A \Phi )</th>
<th>( T ) ( B \Phi )</th>
<th>( U ) ( A \Phi )</th>
<th>( V ) ( B \Phi )</th>
<th>( R \times T ) ( A ) ( B ) ( \Phi )</th>
<th>( R \times T - U \times V ) ( A ) ( B ) ( \Phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 ( r_1 )</td>
<td>1 ( t_1 )</td>
<td>1 ( u_1 )</td>
<td>1 ( v_1 )</td>
<td>1 1 ( r_1 t_1 )</td>
<td>1 1 ( r_1 t_1 \neg (u_1 v_1) )</td>
</tr>
<tr>
<td></td>
<td>2 ( r_2 )</td>
<td>2 ( t_2 )</td>
<td>2 ( u_2 )</td>
<td>2 ( v_2 )</td>
<td>1 2 ( r_1 t_2 )</td>
<td>1 2 ( r_1 t_2 \neg (u_1 v_2) )</td>
</tr>
</tbody>
</table>

Variables entangle in \( \Psi \) beyond read-once factorization. This is the pivotal intricacy introduced by the difference operator.
Query Evaluation Example (1/3)

Consider the following query and tuple-independent database:

\[ Q = \pi_\emptyset \left[ (R(A) \times T(B)) - (U(A) \times V(B)) \right] \]

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>T</th>
<th>U</th>
<th>V</th>
<th>R \times T</th>
<th>R \times T - U \times V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A \Phi</td>
<td>B \Phi</td>
<td>A \Phi</td>
<td>B \Phi</td>
<td>A B \Phi</td>
<td>A B \Phi</td>
</tr>
<tr>
<td>1</td>
<td>1 r_1</td>
<td>1 t_1</td>
<td>1 u_1</td>
<td>1 v_1</td>
<td>1 1 r_1 t_1</td>
<td>1 1 r_1 t_1 (u_1 v_1)</td>
</tr>
<tr>
<td>2</td>
<td>2 r_2</td>
<td>2 t_2</td>
<td>2 u_2</td>
<td>2 v_2</td>
<td>1 2 r_1 t_2</td>
<td>1 2 r_1 t_2 (u_1 v_2)</td>
</tr>
</tbody>
</table>

The grounding 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 the difference operator.
Query Evaluation Example (2/3)

Translate \( Q = \pi_0 \left[ (R(A) \times T(B)) - (U(A) \times V(B)) \right] \) 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 \( \exists \)-consistent: Same order \( \exists_A \exists_B \) for \( Q_1 \) and \( Q_2 \).

Query grounding:

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

- 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.
Compile grounding formula into OBDD:

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

Probabilistic Databases 101

The Dichotomy

The Interesting but Hard Queries

The Easy Queries

Leftovers
Dichotomies Beyond $1\text{RA}^-$

Some known dichotomies

- Non-repeating CQ, UCQ  
  [Dalvi & Suciu 2004, 2010]
- Quantified queries, ranking queries  
  [O.& team 2011, 2012]

Non-repeating relational algebra $= 1\text{RA}^- + \text{union}$.

- Hierarchical property not enough, consistency also needed.
- $\pi_\emptyset[(R(A) \Join S_1(A, B) \cup T(B) \Join S_2(A, B)) - S(A, B)]$ is hard, though it is equivalent to a union of two hierarchical $1\text{RA}^-$ queries.

Non-repeating relational calculus

- $S(x, y) \land \neg R(x)$ is tractable,  
  $S(x, y) \land (R(x) \lor T(y))$ is hard.
  - Both are non-repeating, yet not expressible in $1\text{RA}^-$.  
- Possible (though expensive) approach:
  - Translate to $\exists \text{RC}$ and check $\exists \text{RC}$-hierarchical and $\exists$-consistency.

Full relational algebra (or full relational calculus)

- It is undecidable whether the union of two equivalent relational algebra queries, one hard and one tractable, is tractable.
Thank you!