A Dichotomy for Non-Repeating Queries with Negation in Probabilistic Databases

Robert Fink and Dan Olteanu

PODS June 24, 2014
Outline

The Dichotomy
The Interesting but Hard Queries
The Easy Queries
Leftovers
Relational algebra query language fragment $1RA^-$
- Included: Equi-joins, selections, projections, difference
- Excluded: Repeating relation symbols (self-joins), unions

Tuple-independent probabilistic model
- Each tuple associated with a fresh Boolean random variable $x$.
- $P(x)$ is the probability that the tuple exists in the database.
- Simplest probabilistic model in the literature.
  Beyond this model, query tractability is quickly lost.
- Used by real-world large-scale probabilistic repositories, e.g., Google Knowledge Vault.

Query Evaluation Problem: For a fixed $1RA^-$ query $Q$: Given a tuple-independent probabilistic database $D$ and a tuple $t \in Q(D)$, compute its marginal probability.
The Main Result

Data complexity of any \( 1\text{RA}^- \) query \( Q \) on tuple-independent databases:

- Polynomial time if \( Q \) is **hierarchical** and
- \#P-hard otherwise.
The Main Result

Data complexity of any $1\text{RA}^-$ query $Q$ on tuple-independent databases:

- Polynomial time if $Q$ is hierarchical and
- \#P-hard otherwise.

This result strictly extends a 2004 result by Dalvi and Suciu:

- We added the relational algebra difference operator
  - and moved from conjunctive queries without self-joins to $1\text{RA}$.

- Same syntactic characterization of tractable queries.
  - The hierarchical property can be recognized in LOGSPACE.

- The reason for tractability is however different.
Hierarchical $1RA^-$ Queries

Let $[C]$ be the equivalence class of attribute $C$ in query $Q$ as defined by the transitivity of equi-join conditions and difference operators.

- E.g., $C$ and $D$ are in the same class due to join $X(C) \bowtie_{C=D} Y(D)$ or difference $X(C) - C \leftrightarrow_D Y(D)$ under attribute mapping $C \leftrightarrow D$. 
Hierarchical 1RA⁻ Queries

Let \([C]\) be the equivalence class of attribute \(C\) in query \(Q\) as defined by the transitivity of equi-join conditions and difference operators.

- E.g., \(C\) and \(D\) are in the same class due to join \(X(C) \Join_{C=D} Y(D)\) or difference \(X(C) - C \leftrightarrow D Y(D)\) under attribute mapping \(C \leftrightarrow D\).

(Boolean*) 1RA⁻ query \(Q\) is hierarchical if

For every pair of distinct attribute equivalence classes \([A]\) and \([B]\), there is no triple of relation symbols \(R\), \(S\), and \(T\) in \(Q\) such that

- \(R^{[A]}[-B]\) has attributes in \([A]\) and not in \([B]\),
- \(S^{[A][B]}\) has attributes in both \([A]\) and \([B]\), and
- \(T^{-A}^{[B]}\) has attributes in \([B]\) and not in \([A]\).

* For non-Boolean queries, we need not check for equivalence classes with attributes in the query result.
Examples

Examples of hierarchical queries:

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

Examples of hierarchical queries:

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

Examples of non-hierarchical queries:

- \( \pi_\emptyset \left[ R(A) \Join S(A, B) \Join T(B) \right] \)
- \( \pi_\emptyset \left[ \pi_B \left( R(A) \Join S(A, B) \right) - T(B) \right] \)
- \( \pi_\emptyset \left[ T(B) - \pi_B \left( R(A) \Join S(A, B) \right) \right] \)
- \( \pi_\emptyset \left[ X(A) \Join \left[ R(A) - \pi_A \left( T(B) \Join S(A, B) \right) \right] \right] \)
Outline

The Dichotomy

The Interesting but Hard Queries

The Easy Queries

Leftovers
Hardness Proof Idea

Reduction from \#P-hard model counting problem for positive 2DNF:

- Given a non-hierarchical 1RA query \( Q \) and
- A positive bipartite DNF formula \( \Psi \),

- Construct a tuple-independent database \( D \) with
  - size polynomial in the number of variables and clauses in \( \Psi \), and
  - tuples annotated with variables in \( \Psi \) such that \( \Psi \) annotates \( Q(D) \).

- Then \( \#\Psi = 2^n \cdot P_{Q(D)} \), where
  - \( P_{Q(D)} \) is the probability of \( Q(D) \),
  - \( 1/2 \) is the probability of each variable in \( \Psi \), and
  - \( n \) is the number of variables in \( \Psi \).
Example of Hardness Reduction

Input formula and query:
- \( \Psi = x_1 y_1 \lor x_1 y_2 \),
- \( Q = \pi_\emptyset \left[ R(A) - \pi_A(T(B) \Join S(A, B)) \right] \)

Construct database such that \( \Psi \) annotates \( Q \)'s (nullary) result:
- Column \( \Phi \) holds annotations over variables in \( \Psi \).
  - Special annotations: \( \top \) (true), \( \bot \) (false)
- Variables used as constants for the attribute \( B \) in \( T \) and \( S \).

- \( S(a, b, \phi) \): Clause \( a \) has variable \( b \) exactly when \( \phi \) is true.
- \( R(a, \top) \) 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 \Join S )</th>
<th>( \pi_A(T \Join S) )</th>
<th>( R - \pi_A(T \Join 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 ( x_1 y_1 )</td>
<td></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>1 ( x_1 y_2 )</td>
<td></td>
</tr>
<tr>
<td>( y_2 \neg y_2 )</td>
<td>1 ( y_2 \bot )</td>
<td>1 ( y_2 \bot )</td>
<td>1 ( y_2 \bot )</td>
<td>2 ( \top )</td>
<td></td>
</tr>
<tr>
<td>( y_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 \bot )</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!
Example of Hardness Reduction

Input formula and query:

- $\Psi = x_1y_1 \lor x_1y_2$,
- $Q = \pi_\emptyset \left[ R(A) - \pi_A(T(B) \Join S(A, B)) \right]$ 

Construct database such that $\Psi$ annotates $Q$’s (nullary) result:

- Column $\Phi$ holds annotations over variables in $\Psi$.
  - Special annotations: $\top$ (true), $\bot$ (false)
- Variables used as constants for the attribute $B$ in $T$ and $S$.

- $S(a, b, \phi)$: Clause $a$ has variable $b$ exactly when $\phi$ is true.
- $R(a, \top)$ 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>$T \Join S$</th>
<th>$\pi_A(T \Join S)$</th>
<th>$R - \pi_A(T \Join S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\Phi$</td>
<td>$B$</td>
<td>$\Phi$</td>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>1</td>
<td>$\top$</td>
<td>$x_1 \neg x_1$</td>
<td>1</td>
<td>$x_1 \top$</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\top$</td>
<td>$y_1 \neg y_1$</td>
<td>1</td>
<td>$y_1 \top$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$y_2 \neg y_2$</td>
<td>2</td>
<td>$y_2 \bot$</td>
<td>2</td>
</tr>
</tbody>
</table>

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

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

- Binary trees with leaves $A$, $AB$, and $B$ and inner nodes $\Box$ or $-$.
  - 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 $P_{x,y}$.
Hard Query Patterns

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

- Binary trees with leaves $A$, $AB$, and $B$ and inner nodes $\mathbin{\boxtimes}$ or $\mathbin{-}$.
  - 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 $P_{x,y}$.

$P_{1.1}$ is the only hard pattern to consider w/o the difference operator!
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 \( \boxdot \) and \( - \),
  - preserves ancestor-descendant relationships,
  - maps leaves \( A, AB, B \) to relations \( R[A][-B], S[A][B], T[-A][B] \).

- The match preserves the annotation of the query pattern: \( Q \) and \( P_{x,y} \) have the same annotation for any input database.
Outline

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 the query annotation $\Psi$.
- Compile $\Psi$ into poly-size OBDD($\Psi$).
- Compute probability of OBDD($\Psi$) in time linear in its size.
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 the query annotation $\Psi$.
- Compile $\Psi$ into poly-size OBDD($\Psi$).
- Compute probability of OBDD($\Psi$) in time linear in its size.

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

- $1RA^-$ queries w/o difference: Annotations are read-once.
  - Read-once annotations admit linear-size OBBDs.

- $1RA^-$ queries: Annotations are not read-once.
  - They admit OBBDs of size linear in the database size but exponential in the query size.
From hierarchical 1RA$^-$ to RC-hierarchical $\exists$-consistent $\text{RC}^\exists$: 

- 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 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.
Query Evaluation Example

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 \otimes T</th>
<th>R \otimes T - U \otimes V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A φ</td>
<td>B φ</td>
<td>A φ</td>
<td>B φ</td>
<td>A B φ</td>
<td>A B φ</td>
</tr>
<tr>
<td>1</td>
<td>r₁</td>
<td>t₁</td>
<td>u₁</td>
<td>v₁</td>
<td>1 1 r₁ t₁</td>
<td>1 1 r₁ t₁ ¬(u₁ v₁)</td>
</tr>
<tr>
<td>2</td>
<td>r₂</td>
<td>t₂</td>
<td>u₂</td>
<td>v₂</td>
<td>1 2 r₁ t₂</td>
<td>1 2 r₁ t₂ ¬(u₁ v₂)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 1 r₂ t₁</td>
<td>2 1 r₂ t₁ ¬(u₂ v₁)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 2 r₂ t₂</td>
<td>2 2 r₂ t₂ ¬(u₂ v₂)</td>
</tr>
</tbody>
</table>
Query Evaluation Example

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>( R )</th>
<th>( T )</th>
<th>( U )</th>
<th>( V )</th>
<th>( R \Join T )</th>
<th>( R \Join T - U \Join V )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( \Phi )</td>
<td>( B )</td>
<td>( \Phi )</td>
<td>( A )</td>
<td>( \Phi )</td>
</tr>
<tr>
<td>1 ( r_1 )</td>
<td>1 ( t_1 )</td>
<td>1 ( u_1 )</td>
<td>1 ( v_1 )</td>
<td>1 1 ( r_1t_1 )</td>
<td>1 1 ( r_1t_1 \neg(u_1v_1) )</td>
</tr>
<tr>
<td>2 ( r_2 )</td>
<td>2 ( t_2 )</td>
<td>2 ( u_2 )</td>
<td>2 ( v_2 )</td>
<td>1 2 ( r_1t_2 )</td>
<td>1 2 ( r_1t_2 \neg(u_1v_2) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 1 ( r_2t_1 )</td>
<td>2 1 ( r_2t_1 \neg(u_2v_1) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 2 ( r_2t_2 )</td>
<td>2 2 ( r_2t_2 \neg(u_2v_2) )</td>
</tr>
</tbody>
</table>

The annotation of \( Q \) is:

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

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

Translate $Q = \pi_0 \left[ (R(A) \times T(B)) - (U(A) \times V(B)) \right]$ into $\text{RC}^\exists$:

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

- 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 annotation:

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

- 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.
Query Evaluation Example (3)

Compile query annotation into OBDD:

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

The Dichotomy

The Interesting but Hard Queries

The Easy Queries

Leftovers
Dichotomies Beyond 1RA−

Some known dichotomies

- Conjunctive queries w/o self-joins [Dalvi & Suciu 2004]
- Unions of conjunctive queries [Dalvi & Suciu 2010]
- Quantified relational algebra queries [F. & O. & Rath 2011]

Full relational algebra

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

Non-repeating relational algebra = 1RA− + union.

- Hierarchical property not enough.
- \( \pi_0[(R(A) \Join S_1(A, B) \cup T(B) \Join S_2(A, B)) - S(A, B)] \) is hard, though it is equivalent to a union of two hierarchical 1RA− 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-repeatable, yet not expressible in 1RA−.
- Possible (though expensive) approach:
  - Translate to RC₃ and check RC-hierarchical and \( \exists \)-consistency.
Thank you!