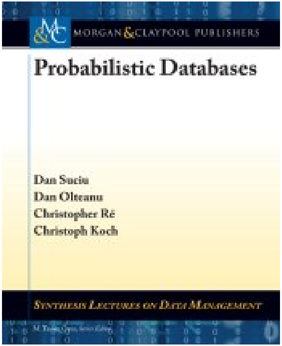
A **Dichotomy** for Queries with **Negation** in **Probabilistic** Databases

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.
- $P(x_i = \textit{true})$ gives the probability that t_i exists in the database.

Possible-worlds semantics:

- Each possible world defined by an assignment θ of the variables $(x_i)_{i \in [n]}$:
  - It consists of all tuples t_i for which $\theta(x_i) = \textit{true}$.
  - It has probability $P(\theta) = \Pi_{i \in [n]} P(x_i = \theta(x_i))$.
- 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: π **(projection)**, $\times$ **(Cartesian product)**, $-$ **(set difference)**, $\bowtie$ **(join)**, σ (selection), $\cup$ (set union), δ (renaming)
- As expressive as domain relational calculus (RC)

In this talk: **Relational algebra fragment $\mathbf{1RA}^-$**

- 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: π **(projection)**, $\times$ **(Cartesian product)**, $-$ **(set difference)**, $\bowtie$ **(join)**, σ (selection), $\cup$ (set union), δ (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)?

$$\pi_\emptyset\big[(R(A) \times T(B)) \quad - \quad (U(A) \times V(B))\big]$$

$$\exists_A \exists_B\big[(R(A) \wedge T(B)) \quad \wedge\neg \quad (U(A) \wedge V(B))\big] \qquad \text{(in RC)}$$

- Does relation S "hold hands" with both R and T?

$$\pi_\emptyset\big[R(A) \bowtie S(A, B) \bowtie T(B)\big]$$

$$\exists_A \exists_B\big[R(A) \wedge S(A, B) \wedge T(B)\big] \qquad \text{(in RC)}$$

The Query Evaluation Problem

For any Boolean $1\mathrm{RA}^{-}$ 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

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 1RA$^-$ Queries

(Boolean) 1RA$^-$ query Q is *hierarchical* if

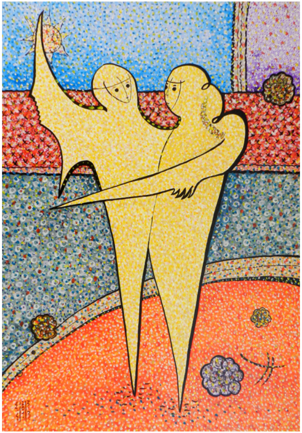
- For every pair of distinct query variables A and B in Q,
- there is no triple of relation symbols R, S, and T in Q such that:
- R has A but not B, S has both A and B, and T has B but not A.

$R(A)$ $S(A, B)$ $T(B)$

Hierarchical 1RA^{-} Queries

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

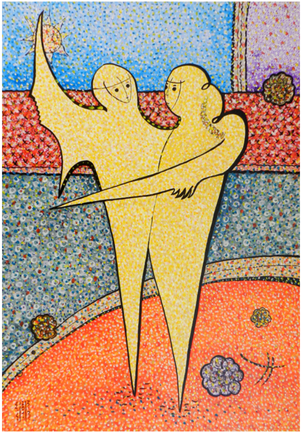
- For every pair of distinct query variables A and B in Q,
- there is no triple of relation symbols R, S, and T in Q such that:
- R has A but not B, S has both A and B, and T has B but not A.

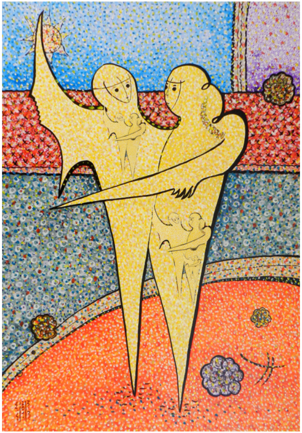


Hierarchical 1RA$^-$ Queries

(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.





Examples

Hierarchical queries:

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

Examples

Hierarchical queries:

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

Non-hierarchical queries:

- $\pi_\emptyset\big[R(A) \bowtie S(A,B) \bowtie T(B)\big]$
- $\pi_\emptyset\Big[\pi_B\big(R(A) \bowtie S(A,B)\big) - T(B)\Big]$
- $\pi_\emptyset\Big[T(B) - \pi_B\big(R(A) \bowtie S(A,B)\big)\Big]$
- $\pi_\emptyset\Big[X(A) \bowtie \big[R(A) - \pi_A\big(T(B) \bowtie S(A,B)\big)\big]\Big]$

Outline

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 Ψ over disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ 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 Ψ.

A Simple Case

Input formula and query:

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

Construct a database D such that Ψ becomes the grounding of Q wrt D:

- Column Φ holds formulas over random variables.
  - We use $\top$ for *true* and $\bot$ for *false*.
- Variables also used as constants for A and B.
- $S(\mathtt{x_i}, \mathtt{y_j}, \top)$: x_iy_j is a clause in Ψ.
- $R(\mathtt{x_i}, x_i)$ and $T(\mathtt{y_j}, y_j)$: x_i is a variable in $\mathbf{X}$ and y_j is a variable in $\mathbf{Y}$.

R

A	Φ
$\mathtt{x_1}$	x_1
$\mathtt{x_2}$	x_2

T

B	Φ
$\mathtt{y_1}$	y_1
$\mathtt{y_2}$	y_2

S

A	B	Φ
$\mathtt{x_1}$	$\mathtt{y_1}$	$\top$
$\mathtt{x_1}$	$\mathtt{y_2}$	$\top$
$\mathtt{x_2}$	$\mathtt{y_1}$	$\top$

$R \bowtie S \bowtie T$

A	B	Φ
$\mathtt{x_1}$	$\mathtt{y_1}$	x_1y_1
$\mathtt{x_1}$	$\mathtt{y_2}$	x_1y_2
$\mathtt{x_2}$	$\mathtt{y_1}$	x_2y_1

$\pi_\emptyset\left[R \bowtie S \bowtie T\right]$

	Φ
()	Ψ

A Simple Case

Input formula and query:

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

Construct a database D such that Ψ becomes the grounding of Q wrt D:

- Column Φ holds formulas over random variables.
  - We use $\top$ for *true* and $\bot$ for *false*.
- Variables also used as constants for A and B.

- $S(\mathtt{x_i}, \mathtt{y_j}, \top)$: $x_i y_j$ is a clause in Ψ.
- $R(\mathtt{x_i}, x_i)$ and $T(\mathtt{y_j}, y_j)$: x_i is a variable in $\mathbf{X}$ and y_j is a variable in $\mathbf{Y}$.

R	
A	Φ
$\mathtt{x_1}$	x_1
$\mathtt{x_2}$	x_2

T	
B	Φ
$\mathtt{y_1}$	y_1
$\mathtt{y_2}$	y_2

S		
A	B	Φ
$\mathtt{x_1}$	$\mathtt{y_1}$	$\top$
$\mathtt{x_1}$	$\mathtt{y_2}$	$\top$
$\mathtt{x_2}$	$\mathtt{y_1}$	$\top$

$R \bowtie S \bowtie T$		
A	B	Φ
$\mathtt{x_1}$	$\mathtt{y_1}$	$x_1 y_1$
$\mathtt{x_1}$	$\mathtt{y_2}$	$x_1 y_2$
$\mathtt{x_2}$	$\mathtt{y_1}$	$x_2 y_1$

$\pi_\emptyset [R \bowtie S \bowtie T]$	
	Φ
()	Ψ

This is the only minimal hard pattern for *positive* $1\mathrm{RA}^-$ queries!

A Surprising Case

Input formula and query:

- $\Psi = x_1 y_1 \vee x_1 y_2$ over sets $\mathbf{X} = \{x_1\}, \mathbf{Y} = \{y_1, y_2\}$
- $Q = \pi_\emptyset \Big[R(A) - \pi_A \big(T(B) \bowtie S(A, B) \big) \Big]$

Construct a database D such that Ψ becomes the grounding of Q wrt D:

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

R

A	Φ
1	$\top$
2	$\top$

T

B	Φ
$\mathtt{x}_1$	$\neg x_1$
$\mathtt{y}_1$	$\neg y_1$
$\mathtt{y}_2$	$\neg y_2$

S

A	B	Φ
1	$\mathtt{x}_1$	$\top$
1	$\mathtt{y}_1$	$\top$
2	$\mathtt{x}_1$	$\top$
2	$\mathtt{y}_2$	$\top$

$T \bowtie S$

A	B	Φ
1	$\mathtt{x}_1$	$\neg x_1$
1	$\mathtt{y}_1$	$\neg y_1$
2	$\mathtt{x}_1$	$\neg x_1$
2	$\mathtt{y}_2$	$\neg y_2$

$\pi_A(T \bowtie S)$

A	Φ
1	$\neg x_1 \vee \neg y_1$
2	$\neg x_1 \vee \neg y_2$

$R - \pi_A(T \bowtie S)$

A	Φ
1	$x_1 y_1$
2	$x_1 y_2$

A Surprising Case

Input formula and query:

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

Construct a database D such that Ψ becomes the grounding of Q wrt D:

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

R

A	Φ
1	$\top$
2	$\top$

T

B	Φ
$\mathtt{x}_1$	$\neg x_1$
$\mathtt{y}_1$	$\neg y_1$
$\mathtt{y}_2$	$\neg y_2$

S

A	B	Φ
1	$\mathtt{x}_1$	$\top$
1	$\mathtt{y}_1$	$\top$
2	$\mathtt{x}_1$	$\top$
2	$\mathtt{y}_2$	$\top$

$T \bowtie S$

A	B	Φ
1	$\mathtt{x}_1$	$\neg x_1$
1	$\mathtt{y}_1$	$\neg y_1$
2	$\mathtt{x}_1$	$\neg x_1$
2	$\mathtt{y}_2$	$\neg y_2$

$\pi_A(T \bowtie S)$

A	Φ
1	$\neg x_1 \vee \neg y_1$
2	$\neg x_1 \vee \neg y_2$

$R - \pi_A(T \bowtie S)$

A	Φ
1	$x_1 y_1$
2	$x_1 y_2$

This query is already hard when T is the only probabilistic input relation!

A More Involved Case

Input formula and query:

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

We need a different reduction gadget:

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

R

A	Φ
x_1	x_1
x_2	x_2

T

B	Φ
y_1	x_1
y_2	y_2

S

A	B	Φ
x_1	y_1	$\neg z_1$
x_1	y_2	$\neg z_2$
x_2	y_1	$\neg z_3$

$S - R \times T$

A	B	Φ
x_1	y_1	$\neg z_1 \neg (x_1 y_1)$
x_1	y_2	$\neg z_2 \neg (x_1 y_2)$
x_2	y_1	$\neg z_3 \neg (x_2 y_1)$

$\pi_\emptyset[S - R \times T]$

	Φ
()	$\bigvee_{i=1}^{\lvert E\rvert} \neg z_i \neg \psi_i$

- Compute $\#\Psi$ using linearly many calls to the oracle for $P_Q = 1 - P(\Upsilon)$.

The Small Print (1/2)

- $\Psi = \bigvee_{(i,j)\in E} x_i y_j = \psi_1 \vee \cdots \vee \psi_{|E|}$ over disjoint variable sets $\mathbf{X}$ and $\mathbf{Y}$
- Let Θ be the set of assignments of variables $\mathbf{X} \cup \mathbf{Y}$ that satisfy Ψ:

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

- Partition Θ into disjoint sets $\Theta_0 \cup \cdots \cup \Theta_{|E|}$, such that $\theta \in \Theta_i$ if and only if θ satisfies exactly i clauses of Ψ:

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

- $|\Theta_1|, \ldots, |\Theta_{|E|}|$ can be computed using an oracle for P_Υ:

$$\Upsilon = \bigvee_{i=1}^{|E|} \neg z_i \wedge \neg\psi_i \qquad \text{or, equivalently} \qquad \neg\Upsilon = \bigwedge_{i=1}^{|E|} (z_i \vee \psi_i)$$

The Small Print (2/2)

Express the probability of $\neg\Upsilon = \bigwedge_{i=1}^{|E|}(z_i \vee \psi_i)$ as a function of $|\Theta_1|, \ldots, |\Theta_{|E|}|$:

- Fix the probabilities of variables in $\mathbf{X} \cup \mathbf{Y}$ to $1/2$ and of variables in $\mathbf{Z}$ to p_z. Then:

$$P_{\neg\Upsilon} = \sum_{k=0}^{|E|} \underbrace{P\left(\neg\Upsilon \middle| \begin{array}{l}\text{exactly } k \text{ clauses}\\ \text{of } \Psi \text{ are satisfied}\end{array}\right)}_{p_z^{|E|-k}} \cdot \underbrace{P\left(\begin{array}{l}\text{exactly } k \text{ clauses}\\ \text{of } \Psi \text{ are satisfied}\end{array}\right)}_{\frac{1}{2}^{|\mathbf{X}|+|\mathbf{Y}|} \cdot |\Theta_k|}$$

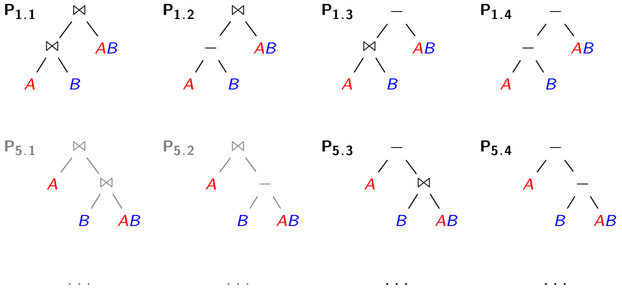
$$= \frac{1}{2}^{|\mathbf{X}|+|\mathbf{Y}|} \sum_{k=0}^{|E|} p_z^{|E|-k} |\Theta_k|$$

- 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_Υ) using Lagrange's polynomial interpolation formula.
- $|E|+1$ oracle calls to P_Υ 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 ⋈ 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.

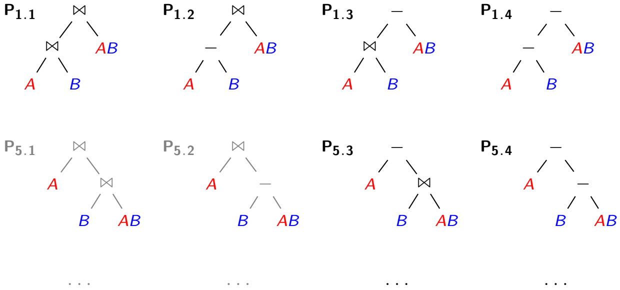


- There is a database construction scheme for each pattern.
- Each non-hierarchical query Q *matches* a pattern $\mathbf{P}_{\mathrm{x.y}}$.

Hard Query Patterns

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

- Binary trees with leaves *A*, *AB*, and *B* and inner nodes ⋈ 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.



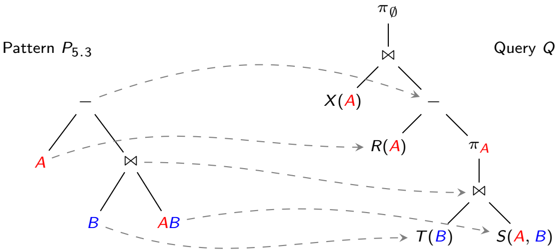
- There is a database construction scheme for each pattern.
- Each non-hierarchical query Q *matches* a pattern $\mathbf{P}_{x.y}$.

In the absence of negation, $\mathbf{P}_{1.1}$ is the only hard pattern to consider!

Non-hierarchical Queries Match Minimal Hard Patterns

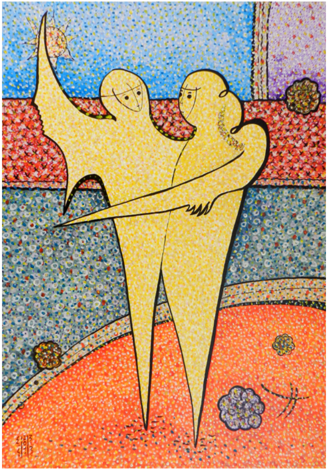
Each non-hierarchical query Q *matches* a pattern $\mathbf{P}_{x.y}$:

- There is a total mapping from $\mathbf{P}_{x.y}$ to Q's parse tree that
  - is identity on inner nodes $\bowtie$ and $-$,
  - 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 $\mathbf{P}_{x.y}$ 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_Ψ of Q's grounding Ψ.
- Compile Ψ into OBDD(Ψ) in polynomial time.
- Compute probability of OBDD(Ψ) in time linear in its size.

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_Ψ of Q's grounding Ψ.
- Compile Ψ into OBDD(Ψ) in polynomial time.
- Compute probability of OBDD(Ψ) in time linear in its size.

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

- $1RA^-$ *without* negation: Grounding formulas are read-once.
  - Read-once formulas admit linear-size OBDDs.

- $1RA^-$: Grounding formulas are <u>not</u> read-once.
  - They admit OBBDs of sizes linear in the database size <u>but</u> 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 $Q_1 \vee \cdots \vee Q_k$.
  - 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.

Query Evaluation Example (1/3)

Consider the following query and tuple-independent database:

$$Q = \pi_\emptyset\Big[\big(R(A) \times T(B)\big) - \big(U(A) \times V(B)\big)\Big]$$

R

A	Φ
1	r_1
2	r_2

T

B	Φ
1	t_1
2	t_2

U

A	Φ
1	u_1
2	u_2

V

B	Φ
1	v_1
2	v_2

$R \times T$

A	B	Φ
1	1	$r_1 t_1$
1	2	$r_1 t_2$
2	1	$r_2 t_1$
2	2	$r_2 t_2$

$R \times T - U \times V$

A	B	Φ
1	1	$r_1 t_1 \neg(u_1 v_1)$
1	2	$r_1 t_2 \neg(u_1 v_2)$
2	1	$r_2 t_1 \neg(u_2 v_1)$
2	2	$r_2 t_2 \neg(u_2 v_2)$

Query Evaluation Example (1/3)

Consider the following query and tuple-independent database:

$$Q = \pi_{\emptyset}\Big[\big(R(A) \times T(B)\big) - \big(U(A) \times V(B)\big)\Big]$$

R

A	Φ
1	r_1
2	r_2

T

B	Φ
1	t_1
2	t_2

U

A	Φ
1	u_1
2	u_2

V

B	Φ
1	v_1
2	v_2

$R \times T$

A	B	Φ
1	1	$r_1 t_1$
1	2	$r_1 t_2$
2	1	$r_2 t_1$
2	2	$r_2 t_2$

$R \times T - U \times V$

A	B	Φ
1	1	$r_1 t_1 \neg(u_1 v_1)$
1	2	$r_1 t_2 \neg(u_1 v_2)$
2	1	$r_2 t_1 \neg(u_2 v_1)$
2	2	$r_2 t_2 \neg(u_2 v_2)$

The grounding of Q is:

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

- Variables entangle in Ψ beyond read-once factorization.
- This is the pivotal intricacy introduced by the difference operator.

Query Evaluation Example (2/3)

Translate $Q = \pi_\emptyset\Big[(R(A) \times T(B)) - (U(A) \times V(B))\Big]$ into $\mathrm{RC}^\exists$:

$$Q_{RC} = \underbrace{\exists_A\big(R(A) \land \neg U(A)\big) \land \exists_B T(B)}_{Q_1} \quad \lor \quad \underbrace{\exists_A R(A) \land \exists_B \big(T(B) \land \neg V(B)\big)}_{Q_2}.$$

- 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:

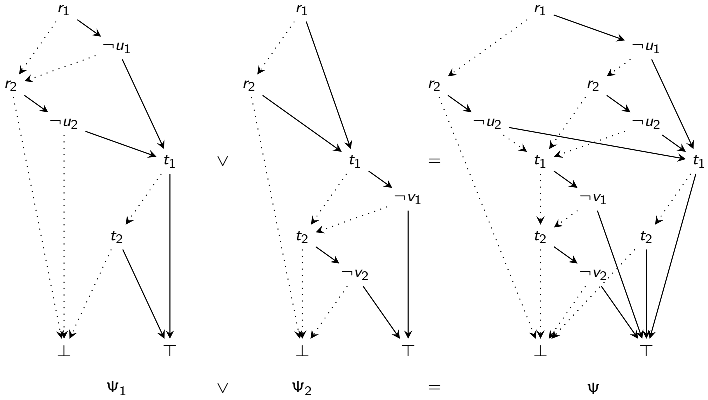
$$\Psi = \underbrace{(r_1\neg u_1 \lor r_2\neg u_2) \land (t_1 \lor t_2)}_{\Psi_1} \quad \lor \quad \underbrace{(r_1 \lor r_2) \land (t_1\neg v_1 \lor t_2\neg v_2)}_{\Psi_2}.$$

- Both Ψ_1 and Ψ_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 grounding formula into OBDD:

$$\Psi = \underbrace{(r_1\neg u_1 \vee r_2\neg u_2) \wedge (t_1 \vee t_2)}_{\Psi_1} \quad \vee \quad \underbrace{(r_1 \vee r_2) \wedge (t_1\neg v_1 \vee t_2\neg v_2)}_{\Psi_2}.$$



Outline

Dichotomies Beyond $1RA^-$

Some known dichotomies

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

Non-repeating relational algebra $= 1RA^- +$ union.

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

Non-repeating relational calculus

- $S(x,y) \wedge \neg R(x)$ is tractable, $S(x,y) \wedge (R(x) \vee T(y))$ is hard.
  - Both are non-repeating, yet not expressible in $1RA^-$.
- Possible (though expensive) approach:
  - Translate to $RC^\exists$ and check 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!