# Tractability in Probabilistic Databases

Dan Suciu

University of Washington

Slides available from http://www.cs.washington.edu/homes/suciu/

# Motivation

Lots of uncertain data:

- *External data* in business intelligence: blogs, emails, twitter, social network data
- Data generated by large *information extraction* systems
- Data integration in the presence of uncertainty
- Specialized data (RFID, scientific data)

# Probabilistic Databases

- In traditional databases: data is *certain*
- In new applications: data is *uncertain*

- Probabilistic Databases: model uncertainty using probabilities

- Theoretical foundations: logic+probability

# Landscape of PDB Research

Incomplete list:

- Representation
  - [Widom'05, Antova'07, Sen'07, Benjelloun'08]
- Query evaluation
  - [Dalvi'04, Antova'08,Olteanu'09, Koch'08,Kimelfeld'09,Deutsch'10]
- Data integration, data exchange
  - [Dong'07,Agrawal'10,Fagin'10]
- Applications
  - [Nierman'02,Keulen'05, Gupta'06,Hassanzadeh'09,Stoyanovich'10]
- Ranking
  - [Re'07,Soliman'07,Zhang'08,Li'09]
- Indexing
  - [Letchner'09,Kanagal'09, Kimura'10]
- Pushing ML in the db engine
  - [Wang'08,Wick'10]

Healthy research …

# Landscape of PDB Systems

- Some research prototypes:
  - MaybeMS (Oxford&Cornell), Trio (Stanford), MystiQ (UW), ProbDB (Maryland), Orion (Purdue)
  - They do not scale to large datasets

- Commercial systems: NONE !
  - Why ? Because to date we do not know how to build *scalable* probabilistic database systems

**Main challenge:** Query Evaluation

# This Talk: Query Evaluation

- I will describe recent progress: *tractable queries*
- I will discuss where we are stuck: *intractable queries*

This talk is based on:

- Dalvi, S. *Efficient query evaluation on probabilistic databases.* VLDB'04
- Dalvi,S. *The Dichotomy of Conjunctive Queries on Probabilistic Structures*, PODS'07
- Dalvi, Schnaitter, S.: *Computing query probability with incidence algebras.* PODS'10
- Gatterbauer, Jha, S., *Dissociation and Propagation for Efficient Query Evaluation over Probabilistic Databases.* MUD'2010

**Advertisement**: Koch, Olteanu, Re, S., *Probabilistic Databases*, Morgan & Claypool, planned for June, 2011

# Outline

- Problem Statement
- Intractable queries
- Tractable queries
- Summary and Open Problems

# Probabilistic Database

Every tuple t in D = random variable (present/absent)
Possible worlds semantics:
D = (**W**, p), where p:**W**→[0,1] s.t. $\Sigma_{W \in \mathbf{W}}$ p(W) = 1

R(A)

| A | |
|---|---|
| $a_1$ | .4 |
| $a_2$ | .2 |
| $a_3$ | .5 |

S(A,B)

| A | B | |
|---|---|---|
| $a_1$ | $b_1$ | .7 |
| $a_2$ | $b_2$ | .4 |
| $a_2$ | $b_3$ | .2 |
| $a_2$ | $b_4$ | .2 |
| $a_3$ | $b_5$ | .5 |

8 tuples → $2^8$ possible worlds

**W** = {$W_1$, $W_2$, …, $W_{256}$}

Need correlations ?
Normalize database !

In this talk = all tuples are independent

# Queries = UCQ

Unions of Conjunctive Queries:

$$Q := R(x_1,\ldots,x_k) \mid \exists x.Q \mid Q_1 \wedge Q_2 \mid Q_1 \vee Q_2$$

In this talk, I consider only Boolean queries:

$$\exists x.\, \exists y.\, R(x) \wedge S(x,y) \equiv R(x),S(x,y)$$

Traditional database: $D \models Q$ (true/false)

Probabilistic database: $P(Q)=\Sigma_{W \in \mathbf{W}:\, W \models Q}\ p(W)$ (in [0,1])

# Lineage

Query Q + Database D = Lineage $F_Q$

- Every tuple t in D = Boolean variable $X_t$
- The *lineage* $F_Q$ = a propositional formula
  - Definition: next slide
  - Intuitively: it says when Q is true on D
- Terminology alert: what we call here *lineage* corresponds to the *PosBool* semiring [Tannen, ICDT'10]

# Definition of Lineage $F_Q$

**Def.** $F_t = X_t$ (t = ground tuple t)

$F_{\exists x.Q} = F_{Q[a1/x]} \vee \ldots \vee F_{Q[an/x]}$ (active dom.=$\{a_1,\ldots a_n\}$)

$F_{Q1 \wedge Q2} = F_{Q1} \wedge F_{Q2}$

$F_{Q1 \vee Q2} = F_{Q1} \vee F_{Q2}$

**Example**

R(A)

| A | |
|---|---|
| $a_1$ | $X_1$ |
| $a_2$ | $X_2$ |
| $a_3$ | $X_3$ |

S(A,B)

| A | B | |
|---|---|---|
| $a_1$ | $b_1$ | $Y_1$ |
| $a_1$ | $b_2$ | $Y_2$ |
| $a_2$ | $b_3$ | $Y_3$ |
| $a_2$ | $b_4$ | $Y_4$ |
| $a_2$ | $b_5$ | $Y_5$ |

Q = R(x),S(x,y)

For any fixed Q, $F_Q$ has a DNF of size polynomial in size(D)

$F_Q = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3 \vee X_2 Y_4 \vee X_2 Y_5$

# Problem Statement

**Problem**. Fix Q. Given D, compute P(Q);
Equivalently: compute $P(F_Q)$.

This talk I discuss tractable and intractable queries

**Dichotomy Theorem.** For any, fixed UCQ Q:

- either P(Q) is in PTIME -- *tractable*
- or P(Q) is hard for $FP^{\#P}$ -- *intractable*

*Data complexity*

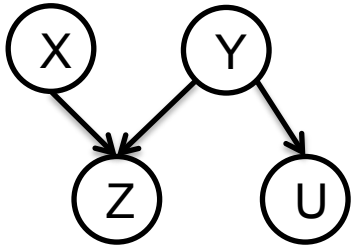
This is a form of probabilistic inference (next)

[Pearl'88,Koller&Friedman'09,Darwiche'09]

# Discussion: Probabilistic Inference

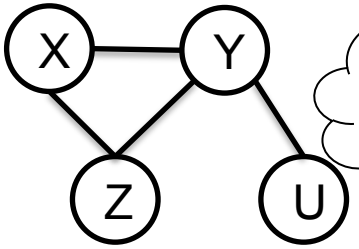
Graphical models:

**Bayesian Network**




$P(X,Y,Z,U) = P(X)\ P(Y)\ P(Z|X,Y)\ P(U|Y)$

**Markov Network**

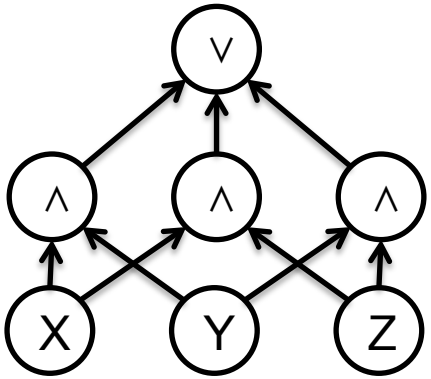


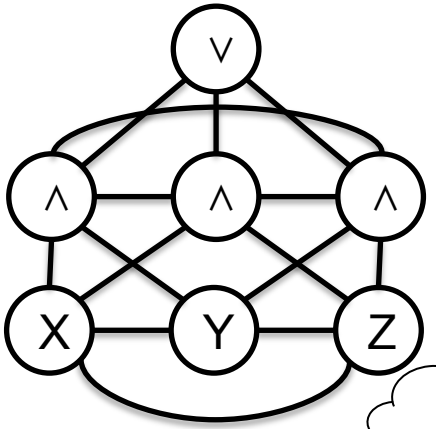

Inference: exponential in tree-width

$\Phi(X,Y,Z,U) = 1/Z * \Phi_1(X,Y,Z)\ \Phi_2(Y,U)$

Propositional formula = *very* special case

$F = X\,Y \vee Y\,Z \vee X\,Z$




Large tree-width

# Discussion: Probabilistic Inference

| | **Graphical Models** | **Prob DBs** |
|---|---|---|
| **Model** | Complex<br>$P(\mathbf{X}) = 1/Z \prod_i \Phi_i(\mathbf{X}_i)$ | Simple<br>Independent tuples |
| **Query** | Simple<br>$Q = P(X,Y \mid Z,U,V)$ | Complex<br>$Q = \exists x. \exists y. R(x) \wedge S(x,y)$ |
| **Network** | Static<br>Bayesian/Markov Network | Dynamic<br>$F_Q = Q + D$ |
| **Complexity** | f(**tree-width**, network) | f(**Q**, D) |

Inference in GM's is exponential in tw; ill suited for PDB
Need new, query-based approach: *data complexity*

# Outline

- Problem Statement
- Intractable queries
- Tractable queries
- Summary and Open Problems

# Intractable Queries

$H_0 = R(x), S(x,y), T(y)$

[Dalvi&S.'04]

$H_1 = R(x_0), S(x_0,y_0) \vee S(x_1,y_1), T(y_1)$

[Dalvi&S.'07, Dalvi,Schnaitter,S.'10]

$H_2 = R(x_0), S_1(x_0,y_0) \vee S_1(x_1,y_1), S_2(x_1,y_1) \vee S_2(x_2,y_2), T(y_2)$

$H_3 = R(x_0), S_1(x_0,y_0) \vee S_1(x_1,y_1), S_2(x_1,y_1) \vee S_2(x_2,y_2), S_3(x_2,y_2) \vee S_3(x_3,y_3), T(y_3)$

. . .

**Theorem.** Data complexity of $H_k$ is hard for $FP^{\#P}$

Will give the proof for $H_0$ and $H_1$ next

# Model Counting

**Definition** A propositional formula F is a *Positive, Partite, 2DNF* if $F = \bigvee_{i,j} X_i Y_j$

**Example:**

$F = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3 \vee X_2 Y_4 \vee X_2 Y_5$

**Theorem** The problem:

"*given a PP2DNF F, count the number of satisfying assignments #F*"

is #P-hard.

[Provan&Ball'83]

Holds also for PP2CNF

[Dalvi&S.'04]

# Proof: $H_0$ is hard for $FP^{\#P}$

$H_0$= R(x),S(x,y),T(y)

Reduction from #PP2DNF

- Given $F = X_{i1}\,Y_{j1} \vee X_{i2}\,Y_{j2} \vee \cdots$

construct D =

R

| A | |
|---|---|
| $X_1$ | ½ |
| $X_2$ | ½ |
| … | |

S

| A | B | |
|---|---|---|
| $X_{i1}$ | $Y_{j1}$ | 1 |
| $X_{i2}$ | $Y_{j2}$ | 1 |
| $X_{i3}$ | $Y_{j3}$ | 1 |
| … | | |

T

| B | |
|---|---|
| $Y_1$ | ½ |
| $Y_2$ | ½ |
| … | |

- Assignment θ ↔ possible world $R^W$,$T^W$
- θ satisfies F ↔ $(R^W, S, T^W) \models H_0$
- Therefore $P(H_0) = \#F / 2^n$, where n = number of variables

We can compute #F using an oracle for $P(H_0)$; hence $H_0$ is hard for $FP^{\#P}$

[Dalvi&S.'07]

# Proof: $H_1$ is hard for $FP^{\#P}$

$H_1 = R(x_0), S(x_0,y_0) \vee S(x_1,y_1), T(y_1)$

By reduction from #PP2CNF:

- $F = (X_{i1} \vee Y_{j1}) \wedge (X_{i2} \vee Y_{j2}) \wedge \ldots \wedge (X_{im} \vee Y_{jm})$

Construct D =

| R | | S | | T | |
|---|---|---|---|---|---|
| A | | A | B | B | |
| $X_1$ | ½ | $X_{i1}$ | $Y_{j1}$ 1-z | $Y_1$ | ½ |
| $X_2$ | ½ | $X_{I2}$ | $Y_{j2}$ 1-z | $Y_2$ | ½ |
| … | | $X_{i3}$ | $Y_{j3}$ 1-z | … | |
| | | … | | | |

- Assignment $\theta \leftrightarrow$ possible world $R^W, T^W$ (no S!)
- $P(\neg H_1 \mid \theta) = z^{\#cnt(\theta)}$, where $\#cnt(\theta) = |\{k \mid \theta(X_{ik} \vee Y_{jk}) = true\}|$
- $P(\neg H_1) = \Sigma_\theta P(\neg H_1 \mid \theta) P(\theta) = 1/2^n \Sigma_{k=0,m} a_k z^k = f(z)$
  where $a_k = |\{\theta \mid \#cnt(\theta) = k\}|$

- Compute $f(z)$ at m+1 distinct values z, using oracle for $P(H_1)$. Obtain all coefficients $a_0, a_1, \ldots, a_m$
- Return $\#F = a_m$

We can compute #F using an oracle for $P(H_1)$; hence $H_1$ is hard for $FP^{\#P}$

# Discussion of Intractable Queries

- $H_0$ corresponds to a *Restricted Boltzmann Machine*

[Salakhutdinov'10]

- The hardness proof for $H_k$, $k \geq 2$, and other queries, requires significant extensions

- There are other intractable queries: will describe all of them shortly

**Open problems:** Is the model counting problem for an intractable query #P hard ?
How do we evaluate/approximate them ?

# Outline

- Problem Statement
- Intractable queries
- Tractable queries
  - Safe plans
  - An algorithm for UCQ queries
- Summary and Open Problems

# Overview

- Traditional query processing is done with query plans using *relational operators*

- Query processing on probabilistic databases is done with query plans using simple *extensions of the relational operators*

# Example

$$Q = \exists x.\, \exists y.\, R(x), S(x,y)$$

$$P(Q) = 1 - \{1 - p_1 * [\ 1 - (1-q_1)*(1-q_2)\ ]\} *$$

$$\{1 - p_2 * [\ 1 - (1-q_3)*(1-q_4)*(1-q_5)\ ]\}$$

$$F_Q = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4 \vee X_2Y_5$$

$$= X_1(Y_1 \vee Y_2) \vee X_2(Y_3 \vee Y_4 \vee Y_5)$$

"Read-once" *

* See Sudeepa Roy's talk on Tuesday

R(A)

| A | P | |
|---|---|---|
| $a_1$ | $p_1$ | $X_1$ |
| $a_2$ | $p_2$ | $X_2$ |
| $a_3$ | $p_3$ | $X_3$ |

S(A,B)

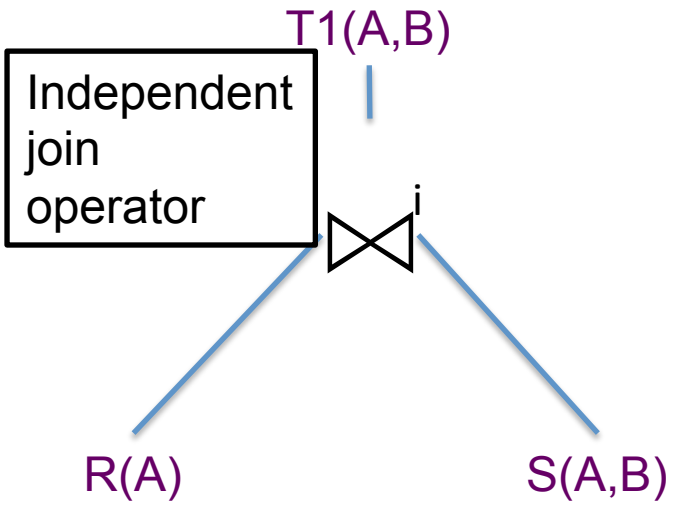
| A | B | P | |
|---|---|---|---|
| $a_1$ | $b_1$ | $q_1$ | $Y_1$ |
| $a_1$ | $b_2$ | $q_2$ | $Y_2$ |
| $a_2$ | $b_3$ | $q_3$ | $Y_3$ |
| $a_2$ | $b_4$ | $q_4$ | $Y_4$ |
| $a_2$ | $b_5$ | $q_5$ | $Y_5$ |

# Operators

| A | B | P | |
|---|---|---|---|
| a1 | b1 | p1*q1 | X1 ∧ Y1 |
| a1 | b2 | p1*q2 | X1 ∧ Y2 |
| a2 | b3 | p2*q3 | X2 ∧ Y3 |
| a2 | b4 | p2*q4 | X2 ∧ Y4 |
| a2 | b5 | p2*q5 | X2 ∧ Y5 |

| A | P | |
|---|---|---|
| a1 | 1 - (1-p1)*(1-p2) | Y1 ∨ Y2 |
| a2 | 1 - (1-p3)*(1-p4)*(1-p5) | Y3 ∨ Y4 ∨Y5 |

Independent join operator




Independent projection operator


… other operators for self-joins and for unions

| A | P | |
|---|---|---|
| a1 | p1 | X1 |
| a2 | p2 | X2 |
| a3 | p3 | X3 |

| A | B | P | |
|---|---|---|---|
| a1 | b1 | q1 | Y1 |
| a1 | b2 | q2 | Y2 |
| a2 | b3 | q3 | Y3 |
| a2 | b4 | q4 | Y4 |
| a2 | b5 | q5 | Y5 |

| A | B | P | |
|---|---|---|---|
| a1 | b1 | q1 | Y1 |
| a1 | b1 | q2 | Y2 |
| a2 | b2 | q3 | Y3 |
| a2 | b3 | q4 | Y4 |
| a2 | b2 | q5 | Y5 |

# $Q = \exists x.\, \exists y.\, R(x), S(x,y)$

[Dalvi&S.'04]

$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)(1-p_2q_4)(1-p_2q_5)$

Wrong

$\Pi^i_\Phi$

$T_1(A,B)$

| $p_1\, q_1$ |
|---|
| $p_1\, q_2$ |
| $p_2\, q_3$ |
| $p_2\, q_4$ |
| $p_2\, q_5$ |

| $p_1$ |
|---|
| $p_2$ |
| $p_3$ |

$\bowtie^i$

| $q_1$ |
|---|
| $q_2$ |
| $q_3$ |
| $q_4$ |
| $q_5$ |

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

$1-\{1-p_1[1-(1-q_1)(1-q_2)]\}^*$
$\{1-p_2[1-(1-q_4)(1-q_5)\ (1-q_6)]\}$

Right

$\Pi^i_\Phi$

$\bowtie^i$

$T_2(A)$

| $1-(1-q_1)(1-q_2)$ |
|---|
| $1-(1-q_4)(1-q_5)\ (1-q_6)$ |

$\Pi^i_A$

$R(A)$

$S(A,B)$

# Discussion

- *An extended operator* manipulates probabilities explicitly: join, projection, etc
- *A safe plan* is a plan that computes the query probability correctly
- Next: I will give an algorithm for computing all tractable UCQ queries. Each tractable query can be computed using a safe plan, using a small set of extended operators, but I will not discuss safe plans in the rest of the talk

# Outline

- Problem Statement
- Intractable queries
- Tractable queries
  - Safe plans
  - An algorithm for UCQ queries
- Summary and Open Problems

[Dalvi,Schaitter,S.'10]

# CQ with Self-Joins

$$Q_J = q_1,\ q_2 = R(x_1), S(x_1,y_1),\ T(x_2), S(x_2,y_2)$$

$$F_J = [X_1(Y_1 \vee Y_2) \vee X_2(Y_3 \vee Y_4 \vee Y_5)] \wedge [Z_1(Y_1 \vee Y_2) \vee Z_2(Y_3 \vee Y_4 \vee Y_5)]$$

Not read-once

$$Q_U = q_1 \vee q_2 = R(x_1), S(x_1,y_1) \vee T(x_2), S(x_2,y_2)$$

$$F_U = (X_1 \vee Z_1)(Y_1 \vee Y_2) \vee (X_2 \vee Z_2)(Y_3 \vee Y_4 \vee Y_5)$$

Read-once

PTIME !

$$P(Q_J) = P(q_1,\ q_2) = P(q_1) + P(q_2) - P(q_1 \vee q_2)$$

# Discussion

- In order to handle self-joins, we needed $\vee$
- CQ = not a natural class to study


- UCQ = the natural class to study
- Will give the algorithm next as five rules

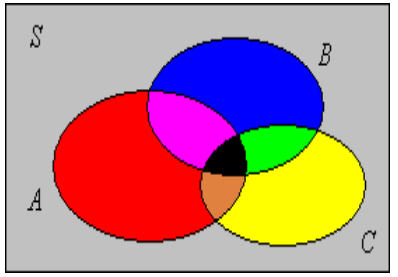
# Five Rules for Query Evaluation

**Rule 1: Inclusion/Exclusion Formula**

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$

Note1: this is the dual of the more popular:

$$P(Q1 \vee Q2 \vee Q3) = \ldots$$




Note2: this rule is not used for inference on GM or on propositional formulae; new in PDBs

# Five Rules for Query Evaluation

**Definition**. z is called *separator variable* if:

- z occurs in every atom A in Q (z is a *root variable*)
- If two atoms $A_1$, $A_2$ in Q are unifiable, then z occurs on a common position in $A_1$, $A_2$.

**Rule 2: Independent Project** If z is separator variable,

$P(\exists z.Q) = 1 - (1 - P(Q[a_1/z])) \times (1 - P(Q[a_2/z])) \times \ldots$

Where active domain = $\{a_1, a_2, a_3, \ldots a_n\}$

**Example**: $Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$

$= \exists z.(\exists y_1.R(z),S(z,y_1) \vee \exists y_2.T(z),S(z,y_2))$

z is "separator variable"

# Five Rules for Query Evaluation

**Rule 3: Independent Join**

$P(Q_1 \wedge Q_2) = P(Q_1) \times P(Q_2)$

If $Q_1$, $Q_2$ are independent (no common symbols)

**Rule 4: Independent Union**

$P(Q_1 \vee Q_2) = 1-(1-P(Q_1)) \times (1-P(Q_2))$

# Five Rules for Query Evaluation

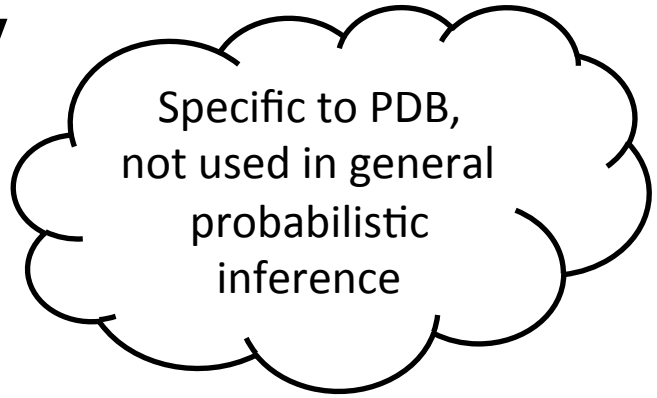
**Rule 5a: Ranking attribute-constant**
Given attribute R(…A…) and constant 'a',
replace R in Q with $R = R_1 \cup R_2$,
where $R_1 = \sigma_{A='a'}(R)$, $R_2 = \sigma_{A\neq'a'}(R)$

**Rule 5b: Ranking attribute-attribute**
Given two attributes R(…A…B…)
replace R in Q with $R = R_1 \cup R_2 \cup R_3$,
where $R_1 = \sigma_{A<B}(R)$ , $R_2 = \sigma_{A>B}(R)$ , $R_3 = \sigma_{A=B}(R)$

**Example**: $q = R(x,y),R(y,x)$ rewrite to $q = R_1(x,y),R_2(y,x) \vee R_3(z,z)$

Note: ranking is applied *before* all other rules

# Summary




- Five rules:
  1. *Inclusion/exclusion*
  2. Existential quantifier elimination (separator !)
  3. Independent AND
  4. Independent OR
  5. Ranking
- Each rule reduces P(Q) to a simpler P(Q')
  - If we succeed, then P(Q) in PTIME
  - If we fail, then P(Q) is hard for $FP^{\#P}$

[Dalvi,Schaitter,S.'10]

# An Example

= $H_1$ (hard !)

DNF

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

[Dalvi,Schaitter,S.'10]

# An Example

= $H_1$ (hard !)

Disconnected query

DNF

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

[Dalvi,Schaitter,S.'10]

# An Example

= $H_1$ (hard !)

Disconnected query

DNF

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

CNF

$$Q_V = [S(x_2,y_2),T(y_2) \vee R(x_3)] \wedge [R(x_1),S(x_1,y_1) \vee T(y_3)]$$

[Dalvi,Schaitter,S.'10]

# An Example

= $H_1$ (hard !)

Disconnected query

DNF

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

CNF

$$Q_V = [S(x_2,y_2),T(y_2) \vee R(x_3)] \wedge [R(x_1),S(x_1,y_1) \vee T(y_3)]$$

Inclusion/exclusion:

PTIME !

$$P(Q_V) = P(q_1 \wedge q_2) = P(q_1) + P(q_2) - P(q_1 \vee q_2)$$

$= R(x_3) \vee T(y_3)$

$Q_V$ has a subquery $H_1$ that is hard, yet it is in PTIME !

[Dalvi,Schaitter,S.'10]

# Another Example

$$
\begin{array}{lllll}
Q_W = [R(x_0), S_1(x_0,y_0) & \vee & S_2(x_2,y_2), S_3(x_2,y_2)] & \wedge & \text{/* Q1 */} \\
\quad [R(x_0), S_1(x_0,y_0) & \vee & S_3(x_3,y_3), T(y_3)] & \wedge & \text{/* Q2 */} \\
\quad [S_1(x_1,y_1), S_2(x_1,y_1) & \vee & S_3(x_3,y_3), T(y_3)] & & \text{/* Q3 */}
\end{array}
$$

[Dalvi,Schaitter,S.'10]

# Another Example

$$
\begin{array}{llll}
Q_W = [R(x_0), S_1(x_0,y_0) & \vee & S_2(x_2,y_2), S_3(x_2,y_2)] \wedge & \text{/* Q1 */} \\
\quad [R(x_0), S_1(x_0,y_0) & \vee & S_3(x_3,y_3), T(y_3)] \wedge & \text{/* Q2 */} \\
\quad [S_1(x_1,y_1), S_2(x_1,y_1) & \vee & S_3(x_3,y_3), T(y_3)] & \text{/* Q3 */}
\end{array}
$$

$$
\begin{aligned}
P(Q_W) = \; & P(Q_1) + P(Q_2) + P(Q_3) + \\
& - P(Q_1 \vee Q_2) - P(Q_2 \vee Q_3) - P(Q_1 \vee Q_3) \\
& + P(Q_1 \vee Q_2 \vee Q_3)
\end{aligned}
$$

Also = $H_3$

= $H_3$ (hard !)

[Dalvi,Schaitter,S.'10]

# Another Example

$$Q_W = [R(x_0), S_1(x_0,y_0) \vee S_2(x_2,y_2), S_3(x_2,y_2)] \wedge \quad /* Q1 */$$
$$[R(x_0), S_1(x_0,y_0) \vee S_3(x_3,y_3), T(y_3)] \wedge \quad /* Q2 */$$
$$[S_1(x_1,y_1), S_2(x_1,y_1) \vee S_3(x_3,y_3), T(y_3)] \quad /* Q3 */$$

$$P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) +$$
$$- P(Q_1 \vee Q_2) - P(Q_2 \vee Q_3) - \cancel{P(Q_1 \vee Q_3)}$$
$$+ \cancel{P(Q_1 \vee Q_2 \vee Q_3)}$$

PTIME !

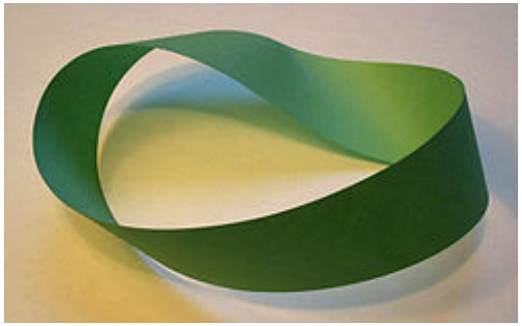
Also = $H_3$

= $H_3$ (hard !)

How do we need to detect cancellations in the inclusion/excusion rule ?

# August Ferdinand Möbius 1790-1868

- Möbius strip
- Möbius function μ in number theory
- Generalized to lattices [Stanley'97,Rota'09]
- And now to queries !



[Dalvi,Schaitter,S.'10]

# From Query Q to Lattice L(Q)

- Write Q in CNF:

  $Q = Q_1 \wedge Q_2 \wedge \ldots \wedge Q_m$.

- For $s \subseteq [m]$, denote $Q_s = \bigvee_{i \in s} Q_i$

**Def**. The *CNF lattice* of Q is L(Q)=(L,≤) where:

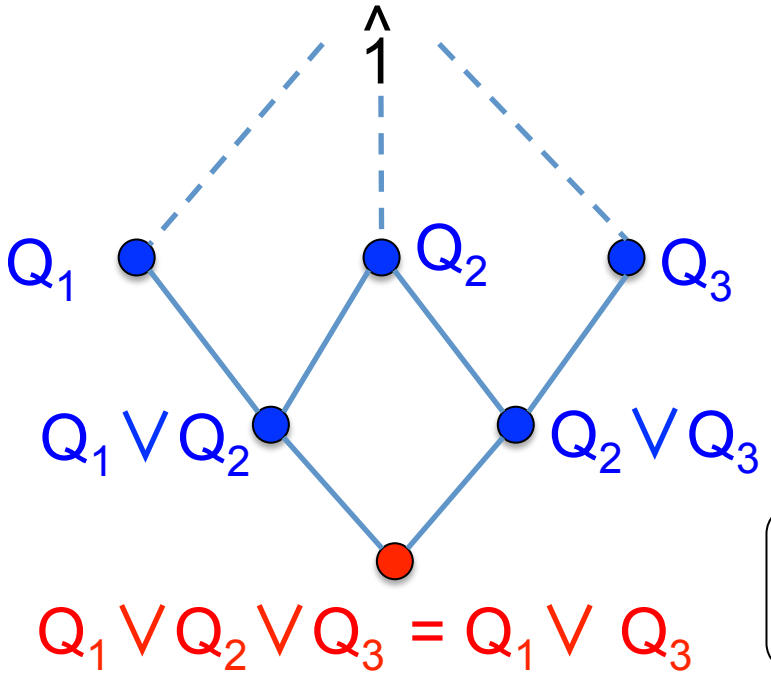
- $L = \{ Q_s \mid s \subseteq [m] \}$ (up to logical equivalence)
- $Q_{s1} \leq Q_{s2}$ if the logical implication $Q_{s2} \rightarrow Q_{s1}$ holds

[Dalvi,Schaitter,S.'10]

# Example

$$Q_W = [R(x_0), S_1(x_0,y_0) \vee S_2(x_2,y_2), S_3(x_2,y_2)] \wedge \text{/* Q1 */}$$
$$[R(x_0), S_1(x_0,y_0) \vee S_3(x_3,y_3), T(y_3)] \wedge \text{/* Q2 */}$$
$$[S_1(x_1,y_1), S_2(x_1,y_1) \vee S_3(x_3,y_3), T(y_3)] \text{ /* Q3 */}$$

$L(Q_W) =$




$\hat{1} = \max(L)$

Blue nodes are in PTIME,
Red nodes are #P hard.

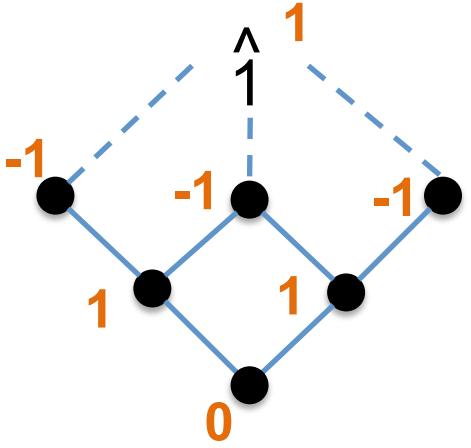
[Stanley 97]

# The Möbius' Function

**Def**. The Möbius function:

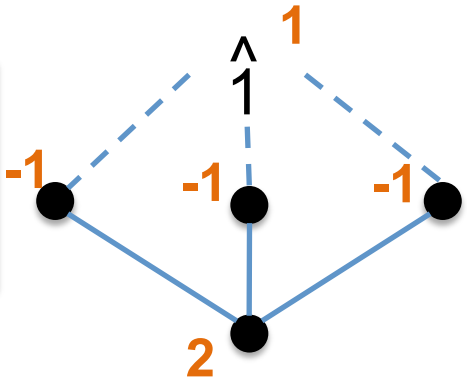
$\mu(\hat{1}, \hat{1}) = 1$

$\mu(u, \hat{1}) = - \Sigma_{u < v \leq \hat{1}} \mu(v, \hat{1})$




**Möbius' Inversion Formula**:

$$P(Q) = - \Sigma_{Qi < \hat{1}} \mu(Qi, \hat{1}) P(Qi)$$




**<u>Rule 1 (revised)</u>**

Inclusion/Exclusion → Mobius' Inversion Formula

[Dalvi,Schaitter,S.'10]

# The Dichotomy

**Dichotomy Theorem** Fix a UCQ query Q.

1. Algorithm terminates, then P(Q) is in PTIME
2. Algorithm fails, then P(Q) is hard for $FP^{\#P}$

Note 1: dichotomy into PTIME/$FP^{\#P}$ based on "syntax" where "syntax" includes the Möbius function !

Note 2: the query complexity is open.

Note 3: we cannot avoid the Möbius function (next).

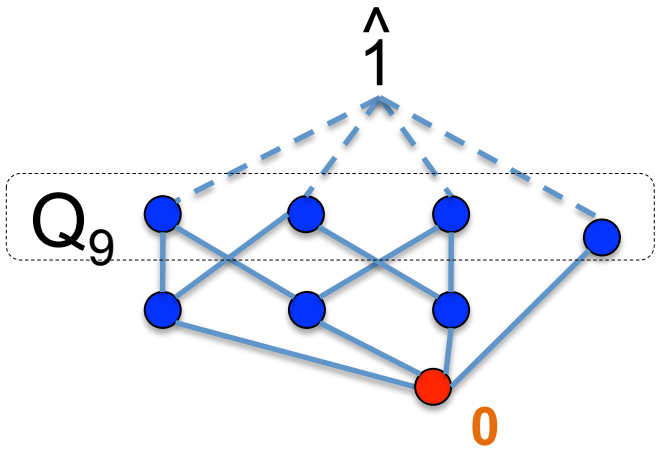
[Dalvi,Schaitter,S.'10]

# Representation Theorem

**THEOREM** For every lattice L, there exists Q s.t. L(Q) ≅ L and:
- The query at $\hat{0}$(= min(L)) is hard for FP$^{\#P}$
- All other queries are in PTIME

Proof
- Let $u_0, u_1, \ldots, u_k$ be the "join irreducibles" of L
- Associate them to the k+1 components of $H_k = h_{k0} \vee h_{k1} \vee \ldots \vee h_{kk}$
- For every $v \in L$, define $Q_v = \vee\{h_{ki} \mid u_i \not\leq v\}$
- Define $Q = \wedge \{Q_v \mid v = \text{co-atom in L}\}$

Example:




Q is in PTIME iff $\mu(\hat{0}, \hat{1})=0$ !

# Summary of Tractable Queries

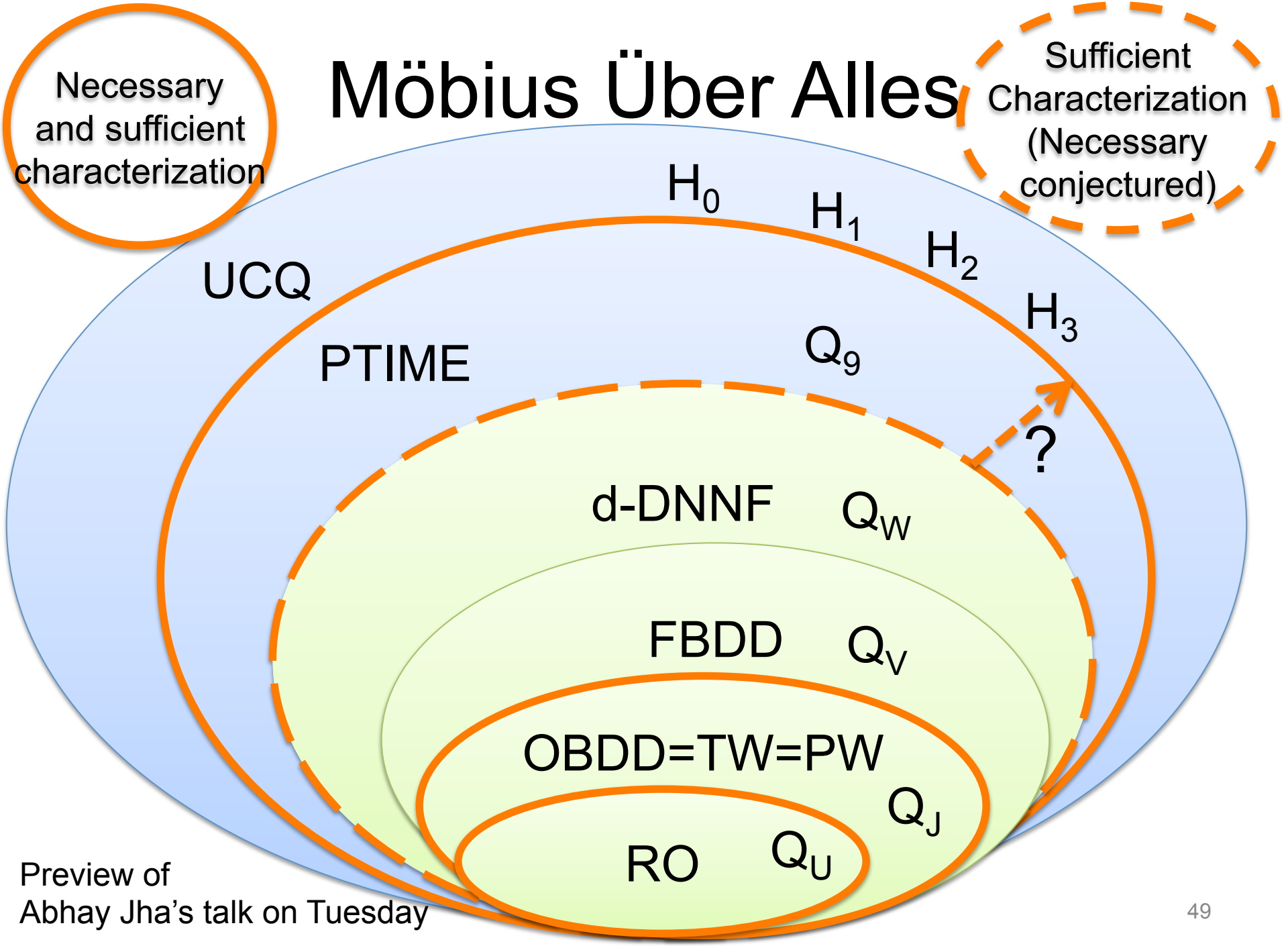
Five *simple* rules

- Each rule has operator (not shown) → *safe plan*
- Open problem: engineering challenges of safe plans

Other approaches to compute P(F) include: read-once formulae, OBDDs, FBDDs, d-DNNF's

- For probabilistic databases, the Möbius Rule rules them all (see Abhay Jha's talk Tuesday)

# Möbius Über Alles




Preview of
Abhay Jha's talk on Tuesday

# Outline

- Problem Statement
- Intractable queries
- Tractable queries
- Summary and Open Problems

# Summary

- Why we care:
  - Strong demand for managing uncertain data
- What is difficult:
  - Computational complexity of Logic + Probabilities

- What we have seen today:
  - Tractable queries: the five rules are simple
  - Intractable queries: hardness proofs are difficult
  - Dichotomy into PTIME/$FP^{\#P}$ based on syntax

# Open Problems

- Correlations using views:
  - Markov Logic [Richardson&Domingos] → Markov DBs ?
- Model counting for $H_0$ (and others) #P hard ?
- Efficient approximation of $H_0$ (and others) ?
- Query complexity for checking hardness ?
- Data (and query) complexity for:
  FO ($\neg$ $\forall$), interpreted predicates ($\neq$, $<$) ?
- Characterize UCQ(FBDD), UCQ(d-DNNF)
- Prove UCQ(d-DNNF) ≠ UCQ(PTIME)

# The 4x Grand Challenge

**Challenge**: make probabilistic dbs run at most 4x slower than deterministic dbs, by using approximations

**Suggested approach:**

- A rich class of tractable UCQ queries are known
- Given any Q, find all tractable Q', s.t. P(Q) ≈ P(Q'):
  - Model theoretic (Robert Fink's talk on Tuesday)
  - Dissociation [Gatterbauer'10] http://LaPushDB.com

This is an **engineering challenge**:
search space for Q'; optimize and execute Q'

# Thank You !

# Questions ?

Slides available from http://www.cs.washington.edu/homes/suciu/