Scalable Uncertainty Management
05 – Query Evaluation in Probabilistic Databases

Rainer Gemulla

Jun 1, 2012
Overview

In this lecture

- Primer: relational calculus
- Understand complexity of query evaluation
- How to determine whether a query is “easy” or “hard”
- How to efficiently evaluate easy queries
  → extensional query evaluation
- How to evaluate hard queries
  → intensional query evaluation
- How to approximately evaluate queries

Not in this lecture

- Possible answer set semantics
- Most representation systems but tuple-independent databases
Outline

1 Primer: Relational Calculus

2 The Query Evaluation Problem

3 Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans

4 Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques

5 Summary
Relational calculus (\( RC \))

- Similar to \( nr\text{-datalog}^- \), but uses a *single query expression*
- Suitable to reason over query expressions as a whole
- Queries are built from logical connectives

\[
q ::= u = v \mid R(x) \mid \exists x.q_1 \mid q_1 \land q_2 \mid q_1 \lor q_2 \mid \neg q_1,
\]

where \( u, v \) are either variables of constants

- Extended \( RC \): adds arithmetic expressions
- Free variables in \( q \) are called *head variables*

**Example**

\( RA \) query:

\[
\pi_{\text{HotelNo, Name, City}}(\text{Hotel} \Join \sigma_{\text{Price} > 500 \lor \text{Type} = \text{'suite'}}(\text{Room}))
\]

\( RC \) query and its abbreviation:

\[
q(h, n, c) \leftarrow \exists r. \exists t. \exists p. \text{Hotel}(h, n, c) \land \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})
\]

\[
q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \land \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})
\]

Alternative \( RC \) query:

\[
q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \land \exists r. \exists t. \exists p. \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})
\]
**Boolean query**

**Definition**

A *Boolean query* is an $\mathcal{RC}$ query with no head variables.

- Asks whether the query result is empty
- Can be obtained from $\mathcal{RC}$-query by
  1. Adding existential quantifiers for the head variables
  2. Replacing head variables by constants (potential results)

**Example**

$\mathcal{RC}$-query:

$$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \land \exists r. \exists t. \exists p. \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{suite})$$

Boolean $\mathcal{RC}$-query (“Is there an answer?”):

$$q \leftarrow \exists h. \exists n. \exists c. \text{Hotel}(h, n, c) \land \exists r. \exists t. \exists p. \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{suite})$$

Another Boolean $\mathcal{RC}$-query (“Is (H1,Hilton,Paris) an answer?”):

$$q \leftarrow \text{Hotel}('H1', 'Hilton', 'Paris') \land \exists r. \exists t. \exists p. \text{Room}(r, 'H1', t, p) \land (p > 500 \lor t = \text{suite})$$
Query semantics

- **Active domain**: set of all constants occurring in the database
- **Active domain semantics**
  1. Every quantifier $\exists x$ ranges over active domain
  2. Query answers are restricted to active domain
- **Domain-independent query**: query result independent of domain (cf. safe queries for datalog)
- Domain-independent queries and query evaluation under active domain semantics are equally expressive

**Example**

- Active domain of $R$: $\{1, 2\}$
- Domain-independent query
  
  $$q(x) \leftarrow \exists y. R(x, y)$$

- Domain-dependent queries
  
  $$q(x) \leftarrow \exists y. \exists z. R(y, z)$$
  $$q(x) \leftarrow \exists y. \neg R(x, y)$$
Relationships between query languages

**Theorem**

Each row of languages in the following table is equally expressive (we consider only safe rules with a single output relation for nr-datalog and domain-independent rules for RC).

<table>
<thead>
<tr>
<th>Relational algebra</th>
<th>nr-datalog⁻</th>
<th>Relational calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPJRU</td>
<td>No repeated head predicates, no negation</td>
<td>(conjunctive queries: CQ)</td>
</tr>
<tr>
<td>(positive RA)</td>
<td>(nr-datalog)</td>
<td>(unions of CQ: UCQ)</td>
</tr>
<tr>
<td>SPJRUDE</td>
<td>−</td>
<td>(RC)</td>
</tr>
<tr>
<td>(RA)</td>
<td>(nr-datalog⁻)</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1 Primer: Relational Calculus
2 The Query Evaluation Problem
3 Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4 Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5 Summary
The query evaluation problem

- Database systems are expected to *scale* to large datasets and *parallelize* to a large number of processors
  → Same behavior is expected from probabilistic databases
- We consider the possible tuple semantics, i.e., a query answer is an ordered set of answer-probability pairs

\[
\{ (t_1, p_1), (t_2, p_2), \ldots \} \quad \text{with} \quad p_1 \geq p_2 \geq \ldots
\]

**Definition (Query evaluation problem)**

Fix a query \( q \). Given a (representation of a) probabilistic database \( \mathcal{D} \) and a possible answer tuple \( t \), compute its marginal probability \( \mathbb{P}( t \in q(\mathcal{D}) ) \).
Questions of interest

- Characterize which queries are hard
  → Understand what makes query evaluation hard
- Given a query, determine whether it is hard
  → Guide query processing
- Given an easy query, solve the QEP
  → Be efficient whenever possible
- Given a hard query, solve the QEP (exactly or approximately)
  → Don’t give up on hard queries
Query evaluation on deterministic databases

Definition

The *data complexity* of a query $q$ is the complexity of evaluating it as a function of the size of the input database. A query is *tractable* if its data complexity is in polynomial time; otherwise, it is *intractable*.

Example

- Fix a relation schema $R$ and consider an instance $I$ with $n$ tuples
- $q(R) = R \rightarrow O(n)$
- $q(R) = \sigma_E(R) \rightarrow O(n)$
- $q(R) = \pi_U(R) \rightarrow O(n^2)$; can be tightened

Theorem

*On deterministic databases, the data complexity of every RA query is in polynomial time. Thus query evaluation is always tractable.*
Query evaluation on probabilistic databases

Corollary

Query evaluation over probabilistic databases is tractable.

Proof.

Fix query $q$. Given a probabilistic database $D = (\mathcal{I}, \mathbb{P})$ with $\mathcal{I} = \{ I^1, \ldots, I^n \}$, perform the following steps:

1. Compute $q(I^k)$ for $1 \leq k \leq n \rightarrow$ polynomial time
2. For each tuple $t \in q(I^k)$ for some $k$, compute

$$\mathbb{P}(t \in q(D)) = \sum_{k: t \in q(I^k)} \mathbb{P}(I^k)$$

$\rightarrow$ polynomially many tuples, polynomial time per tuple

This result is treacherous: It talks about probabilistic databases but not about probabilistic representation systems!
Lineage trees and the query evaluation problem

Example

\[ q(h) \leftarrow \exists n.\exists c.\text{Hotel}(h, n, c) \land \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'}) \]

Room (R)

<table>
<thead>
<tr>
<th>RoomNo</th>
<th>Type</th>
<th>HotelNo</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Suite</td>
<td>H1</td>
<td>$50</td>
</tr>
<tr>
<td>R2</td>
<td>Single</td>
<td>H1</td>
<td>$600</td>
</tr>
<tr>
<td>R3</td>
<td>Double</td>
<td>H1</td>
<td>$80</td>
</tr>
</tbody>
</table>

Hotel (H)

<table>
<thead>
<tr>
<th>HotelNo</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>Hilton</td>
<td>SB</td>
</tr>
</tbody>
</table>

ExpensiveHotels

<table>
<thead>
<tr>
<th>HotelNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
</tr>
</tbody>
</table>

\[ X_1 \land (X_2 \lor X_3) \]

Theorem

Fix a RA query \( q \). Given a boolean pc-table \((T, P)\), we can compute the lineage \( \Phi_t \) of each possible output tuple \( t \) in polynomial time, where \( \Phi_t \) is a propositional formula. We have

\[ P(t \in q(T)) = P(\Phi_t). \]
How can we compute $\Phi_t$?

**A naive approach**

Let $\omega(\Phi)$ be the set of assignments over $\text{Var}(T)$ that make $\Phi$ true. Then apply $\mathbb{P}(\Phi) = \sum_{\theta \in \omega(\Phi)} \mathbb{P}(\theta)$.

Exponential time: $n$ variables $\rightarrow 2^n$ assignments to check!

**Definition (Model counting problem)**

Given a propositional formula $\Phi$, count the number of satisfying assignments $\#\Phi = |\omega(\Phi)|$.

**Definition (Probability computation problem)**

Given a propositional formula $\Phi$ and a probability $\mathbb{P}(X) \in [0, 1]$ for each variable $X$, compute the probability $\mathbb{P}(\Phi) = \sum_{\theta \in \omega(\Phi)} \mathbb{P}(\theta)$.
Model counting is a special case of probability computation

- Suppose we have an algorithm to compute $P(\Phi)$
- We can use the algorithm to compute $\#\Phi$
- Define $P(X) = \frac{1}{2}$ for every variable $X$
- $P(\theta) = \frac{1}{2^n}$ for every assignment ($n =$ number of variables)
- $\#\Phi = P(\Phi) \cdot 2^n$

**Example**

- $\Phi = (X_1 \lor X_2) \land X_4; \ n = 3$
- $\#\Phi = 3$
- $P(\Phi) = \frac{3}{8} = \frac{\#\Phi}{2^n}$
The complexity class \( \#P \)

**Definition**

The complexity class \( \#P \) consists of all function problems of the following type: Given a polynomial-time, non-deterministic Turing machine, compute the number of accepting computations.

**Theorem (Valiant, 1979)**

*Model counting (\( \#SAT \)) is complete for \( \#P \).*

- NP asks whether there exists at least one accepting computation
- \( \#P \) counts the number of accepting computations
- SAT is NP-complete
- \( \#SAT \) is \( \#P \)-complete

Directly implies that probability computation is hard for \( \#P \)!
A graph problem

**Definition (Bipartite vertex cover)**

Given a bipartite graph \((V, E)\), compute \(|\{ S \subseteq V : (u, w) \in E \rightarrow u \in S \lor w \in S \}|\).

**Example**

![Graph diagram]

80 possible ways

**Theorem (Provan and Ball, 1983)**

*Bipartite vertex cover is \#P-complete.*
# PP2DNF and #PP2CNF

**Definition**

Let \( X_1, X_2, \ldots \) and \( Y_1, Y_2, \ldots \) be two disjoint sets of Boolean variables.

- A positive, partitioned 2-CNF propositional formula (PP2CNF) has form \( \Psi = \bigwedge_{(i,j) \in E} (X_i \lor Y_j) \).
- A positive, partitioned 2-DNF propositional formula (PP2DNF) has form \( \Phi = \bigvee_{(i,j) \in E} X_i \land Y_j \).

**Theorem**

#PP2CNF and #PP2DNF are #P-complete.

**Proof.**

#PP2CNF reduces to bipartite vertex cover. For any given \( E \), we have \( \#\Phi = 2^n - \#\Psi \), where \( n \) is the total number of variables.

Note: 2-CNF is in P.
A hard query

**Theorem**

The query evaluation problem of the CQ query $H_0$ given by

$$H_0 \leftarrow R(x) \land S(x, y) \land T(y)$$

on tuple-independent databases is hard for $\#P$.

**Proof.**

Given a PP2DNF formula $\Phi = \bigvee_{(i,j) \in E} X_i Y_j$, where $E = \{(X_{e_1}, Y_{e_1}), (X_{e_2}, Y_{e_2}), \ldots \}$, construct the tuple-independent DB:

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$X$</td>
<td>$Y$</td>
</tr>
<tr>
<td>$X_1$</td>
<td>$X_{e_1}$</td>
<td>$Y_1$</td>
</tr>
<tr>
<td>$1/2$</td>
<td>$X_{e_2}$</td>
<td>$1/2$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$Y_{e_1}$</td>
<td>$Y_2$</td>
</tr>
<tr>
<td>$1/2$</td>
<td>$Y_{e_2}$</td>
<td>$1/2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Then $\#\Phi = 2^n \mathbb{P}(H_0)$, where $n$ is the total number of variables.
More hard queries

Theorem

All of the following $\mathcal{RC}$ queries on tuple-independent databases are $\#P$-hard:

\[
H_0 \leftarrow R(x) \land S(x, y) \land T(y)
\]
\[
H_1 \leftarrow [R(x_0) \land S(x_0, y_0)] \lor [S(x_1, y_1) \land T(y_1)]
\]
\[
H_2 \leftarrow [R(x_0) \land S_1(x_0, y_0)] \lor [S_1(x_1, y_1) \land S_2(x_1, y_1)]
\]
\[
\lor [S_2(x_2, y_2) \land T(y_2)]
\]
\[
\vdots
\]

Queries can be tractable even if they have intractable subqueries!
- $q(x, y) \leftarrow R(x) \land S(x, y) \land T(y)$ is tractable
- $q \leftarrow H_0 \lor T(y)$ is tractable
Extensional and intensional query evaluation

- We’ll say more about data complexity as we go
- Extensional query evaluation
  - Evaluation process guided by query expression $q$
  - Not always possible
  - When possible, data complexity is in polynomial time
- Extensional plans
  - Extensional query evaluation in the database
  - Only minor modifications to RDBMS necessary
  - Scalability, parallelizability retained
- Intensional query evaluation
  - Evaluation process guided by query lineage
  - Reduces query evaluation to the problem of computing the probability of a propositional formula
  - Works for every query
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary
Problem statement

- Tuple-independent database
  - Each tuple $t$ annotated with a unique boolean variable $X_t$
  - We write $P(t) = P(X_t)$
- Boolean query $Q$
  - With lineage $\Phi_Q$
  - We write $P(Q) = P(\Phi_Q)$
- Goal: compute $P(Q)$ when $Q$ is tractable
  - Evaluation process guided by query expression $q$
  - I.e., without first computing lineage!

Example

**Birds**

<table>
<thead>
<tr>
<th>Species</th>
<th>$P$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finch</td>
<td>0.80</td>
<td>$X_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toucan</td>
<td>0.71</td>
<td></td>
<td>$X_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nightingale</td>
<td>0.65</td>
<td></td>
<td></td>
<td>$X_3$</td>
<td></td>
</tr>
<tr>
<td>Humming bird</td>
<td>0.55</td>
<td></td>
<td></td>
<td></td>
<td>$X_4$</td>
</tr>
</tbody>
</table>

- $P(\text{Finch}) = P(X_1) = 0.8$
- Is there a finch? $Q \leftarrow \text{Birds(Finch)}$
  - $\Phi_Q = X_1$
  - $P(Q) = 0.8$
- Is there some bird? $Q \leftarrow \text{Birds}(s)$?
  - $\Phi_Q = X_1 \lor X_2 \lor X_3 \lor X_4$
  - $P(Q) \approx 99.1\%$
Overview of extensional query evaluation

- Break the query into “simpler” subqueries
- By applying one of the rules
  1. Independent-join
  2. Independent-union
  3. Independent-project
  4. Negation
  5. Inclusion-exclusion (or Möbius inversion formula)
  6. Attribute ranking

- Each rule application is polynomial in size of database
- Main results for UCQ queries
  - Completeness: Rules succeed iff query is tractable
  - Dichotomy: Query is \#P-hard if rules don't succeed
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
     - Six Simple Rules
     - Tractability and Completeness
     - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary
**Unifiable atoms**

**Definition**

Two relational atoms $L_1$ and $L_2$ are said to be unifiable (or to unify) if they have a common image. I.e., there exists substitutions such that $L_1[a_1/x_1] = L_2[a_2/x_2]$, where $x_1$ are the variables in $L_1$ and $x_2$ are the variables in $L_2$.

**Example**

### Unifiable:
- $R(a), R(a)$ via $[]$, $[]$
- $R(x), R(y)$ via $[a/x], [a/y]$
- $R(a, y), R(x, y)$ via $[b/y], [(a, b)/(x, y)]$
- $R(a, b), R(x, y)$ via $[], [(a, b)/(x, y)]$
- $R(a, y), R(x, b)$ via $[b/y], [a/x]$

### Not unifiable:
- $R(a), R(b)$
- $R(a, y), R(b, y)$
- $R(x), S(x)$

Unifiable atoms must use the same relation symbol.
**Syntactic independence**

**Definition**

Two queries $Q_1$ and $Q_2$ are called *syntactically independent* if no two atoms from $Q_1$ and $Q_2$ unify.

**Example**

Syntactically independent:
- $R(a)$, $R(b)$
- $R(a, y)$, $R(b, y)$
- $R(x)$, $S(x)$
- $R(a, x) \lor S(x)$, $R(b, x) \land T(x)$

Not syntactically independent:
- $R(a)$, $R(x)$
- $R(x)$, $R(y)$
- $R(x)$, $S(x) \land \neg R(x)$

Checking for syntactic independence can be done in polynomial time in the size of the queries.
Syntactic independence and probabilistic independence

Proposition

Let $Q_1, Q_2, \ldots, Q_k$ be pairwise syntactically independent. Then $Q_1, \ldots, Q_k$ are independent probabilistic events.

Proof.

The sets $\text{Var}(\Phi_{Q_1}), \ldots, \text{Var}(\Phi_{Q_k})$ are pairwise disjoint, i.e., the lineage formulas do not share any variables. Since all variables are independent (because we have a tuple-independent database), the proposition follows.

Example

Syntactically independent:
- $R(a), R(b)$
- $R(a, y), R(b, y)$
- $R(x), S(x)$
- $R(a, x) \lor S(x), R(b, x) \land T(x)$

Not syntactically independent:
- $R(a), R(x)$
- $R(x), R(y)$
- $R(x), S(x) \land \neg R(x)$
Probabilistic independence and syntactic independence

**Proposition**

*Probabilistic independence does not necessarily imply syntactic independence.*

**Example**

- Consider
  
  \[
  Q_1 \leftarrow R(x, y) \land R(x, x)
  \]
  
  \[
  Q_2 \leftarrow R(a, b)
  \]

- If \( \Phi_{Q_1} \) does not contain \( X_{R(a,b)} \), \( Q_1 \) and \( Q_2 \) are independent

- Otherwise, \( \Phi_{Q_1} \) contains \( X_{R(a,b)} \) and therefore \( X_{R(a,b)} \land X_{R(a,a)} \)

- Then, \( \Phi_{Q_1} \) also contains \( X_{R(a,a)} \land X_{R(a,a)} = X_{R(a,a)} \)

- Thus, by the absorption law,
  
  \[
  (X_{R(a,b)} \land X_{R(a,a)}) \lor X_{R(a,a)} = X_{R(a,a)}
  \]

- \( X_{R(a,b)} \) can be eliminated from \( \Phi_{Q_1} \) so that \( Q_1 \) and \( Q_2 \) are independent
Outline

1. Primer: Relational Calculus

2. The Query Evaluation Problem

3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans

4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques

5. Summary
Base case: Atoms

Definition
If $Q$ is an atom, i.e., of form $Q = R(a)$, simply lookup its probability in the database.

Example

Sightings

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>$\mathbb{P}$</th>
<th>$X_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
<td>$X_1$</td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
<td>$X_2$</td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
<td>$X_3$</td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
<td>$X_4$</td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
<td>$X_5$</td>
</tr>
</tbody>
</table>

- Did Mary see a toucan?
- $Q = $ Sightings(Mary, Toucan)
- $\mathbb{P}(Q) = 0.3$
Rule 1: Independent-join

Definition
If $Q_1$ and $Q_2$ are syntactically independent, then

$$P(Q_1 \land Q_2) = P(Q_1) \cdot P(Q_2).$$

Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>$P$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
<td>$X_1$</td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
<td>$X_2$</td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
<td>$X_3$</td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
<td>$X_4$</td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
<td>$X_5$</td>
</tr>
</tbody>
</table>

- Did both Mary and Susan see a toucan?
- $Q = S(Mary, Toucan) \land S(Susan, Toucan)$
- $Q_1 = S(Mary, Toucan)$ $P(Q_1) = 0.3$
- $Q_2 = S(Susan, Toucan)$ $P(Q_2) = 0.5$
- $P(Q) = P(Q_1) \cdot P(Q_2) = 0.15$
Rule 2: Independent-union

**Definition**
If $Q_1$ and $Q_2$ are syntactically independent, then

$$P(Q_1 \lor Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2)). \quad (\text{independent-union})$$

**Example**

**Sightings**

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>$P$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
<td>$X_1$</td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
<td>$X_2$</td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
<td>$X_3$</td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
<td>$X_4$</td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
<td>$X_5$</td>
</tr>
</tbody>
</table>

- Did Mary or Susan see a toucan?
- $Q = S(\text{Mary, Toucan}) \lor S(\text{Susan, Toucan})$
- $Q_1 = S(\text{Mary, Toucan}) \quad P(Q_1) = 0.3$
- $Q_2 = S(\text{Susan, Toucan}) \quad P(Q_2) = 0.5$
- $P(Q) = 1 - (1 - P(Q_1))(1 - P(Q_2)) = 0.65$
Root variables and separator variables

**Definition**

Consider atom $L$ and query $Q$. Denote by $\text{Pos}(L, x)$ the set of positions where $x$ occurs in $Q$ (maybe empty). If $Q$ is of form $Q = \exists x. Q'$:

- Variable $x$ is a *root variable* if it occurs in all atoms, i.e., $\text{Pos}(L, x) \neq \emptyset$ for every atom $L$ that occurs in $Q'$.
- A root variable $x$ is a *separator variable* if for any two atoms that unify, $x$ occurs on a common position, i.e., $\text{Pos}(L_1, x) \cap \text{Pos}(L_2, x) \neq \emptyset$.

**Example**

$Q_1 \leftarrow \exists x. \text{Likes}(a, x) \land \text{Likes}(x, a)$
- $\text{Pos}(\text{Likes}(a, x), x) = \{2\}$
- $\text{Pos}(\text{Likes}(x, a), x) = \{1\}$
- $x$ is root variable
- $x$ is no separator variable

$Q_2 \leftarrow \exists x. \text{Likes}(a, x) \land \text{Likes}(x, x)$
- $x$ is root variable
- $x$ is a separator variable

$Q_3 \leftarrow \exists x. \text{Likes}(a, x) \land \text{Popular}(a)$
- $x$ is no root variable
- $x$ is no separator variable
Separator variables and syntactic independence

Lemma

Let \( x \) be a separator variable in \( Q = \exists x. Q' \). Then for any two distinct constants \( a, b \), the queries \( Q'[a/x], Q'[b/x] \) are syntactically independent.

Proof.

Any two atoms \( L_1, L_2 \) that unify in \( Q' \) do not unify in \( Q'[a/x] \) and \( Q'[b/x] \). Since \( x \) is a separator variable, there is a position at which both \( L_1 \) and \( L_2 \) have \( x \); at this position, \( L_1[a/x] \) has \( a \) and \( L_2[b/x] \) has \( b \).

Example

Sightings

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>( P )</th>
<th>( \exists )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
<td>( X_1 )</td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
<td>( X_2 )</td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
<td>( X_3 )</td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
<td>( X_4 )</td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
<td>( X_5 )</td>
</tr>
</tbody>
</table>

- Has anybody seen a toucan?
- \( Q = \exists x. \text{Sightings}(x, \text{Toucan}) \)
- \( Q'(x) = \text{Sightings}(x, \text{Toucan}) \)
- \( Q'[\text{Mary}/x] = \text{Sightings}({\text{Mary, Toucan}}) \)
- \( Q'[\text{Susan}/x] = \text{Sightings}({\text{Susan, Toucan}}) \)
Rule 3: Independent-project

Definition

If $Q$ is of form $Q = \exists x. Q'$ and $x$ is a separator variable, then

$$P(Q) = 1 - \prod_{a \in \text{ADom}} (1 - P(Q'[a/x])), \hspace{1cm} \text{(independent-project)}$$

where $\text{ADom}$ is the active domain of the database.

Example

Has anybody seen a toucan?

- $Q = \exists x. S(x, \text{Toucan})$
- $Q' = S(x, \text{Toucan})$
- $P(Q) = 1 - \prod_{x \in \{M,S,F,\ldots\}} (1 - P(S(x, T)))$

$$= 1 - (1 - 0.3)(1 - 0.5)1 \cdots 1$$

$$= 0.65$$

Sightings

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>$P$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rule 4: Negation

**Definition**

If the query is \( \neg Q \), then

\[
P(\neg Q) = 1 - P(Q)
\]

(negation)

**Example**

**Sightings**

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>( P )</th>
<th>( X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
<td>( X_1 )</td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
<td>( X_2 )</td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
<td>( X_3 )</td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
<td>( X_4 )</td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
<td>( X_5 )</td>
</tr>
</tbody>
</table>

\( Q = \neg[\exists x. S(x, Toucan)] \)

\[
P(\neg Q) = 1 - P(\exists x. S(x, Toucan)) = 0.35
\]
Rule 5: Inclusion-exclusion

Definition

Suppose \( Q = Q_1 \land Q_2 \land \ldots Q_k \). Then,

\[
P(Q) = - \sum_{\emptyset \neq S \subseteq \{1, \ldots, k\}} (-1)^{|S|} P(\bigvee_{i \in S} Q_i)
\]  

(inclusion-exclusion)

Example

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 12 & 13 & 23 & 123 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 2 & 1 & 1 & 2 \\
1 & 1 & 1 & 2 & 2 & 2 & 3 \\
0 & 0 & 1 & 1 & 1 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & 0 & 0 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
\]

\[
P(Q_1 \land Q_2 \land Q_3) = P(Q_1) + P(Q_2) + P(Q_3) - P(Q_1 \lor Q_2) - P(Q_1 \lor Q_3) - P(Q_2 \lor Q_3) + P(Q_1 \lor Q_2 \lor Q_3)
\]
Inclusion-exclusion for independent-project

Goal of inclusion-exclusion is to apply the rewrite

\[(\exists x_1.Q_1) \lor (\exists x_2.Q_2) \equiv \exists x. (Q_1[x/x_1] \lor Q_2[x/x_2]).\]

Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>(P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>0.8</td>
</tr>
<tr>
<td>Mary</td>
<td>Toucan</td>
<td>0.3</td>
</tr>
<tr>
<td>Susan</td>
<td>Finch</td>
<td>0.2</td>
</tr>
<tr>
<td>Susan</td>
<td>Toucan</td>
<td>0.5</td>
</tr>
<tr>
<td>Susan</td>
<td>Nightingale</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Has both Mary seen some bird and someone seen a finch?

\[
P \left( (\exists x.S(M, x)) \land (\exists y.S(y, F)) \right)
= P(\exists x.S(M, x)) + P(\exists y.S(y, F)) - P(\exists x.S(M, x) \lor (\exists y.S(y, F)))
= 0.86 + 0.84 - P(\exists x.S(M, x) \lor S(x, F))
= 1.7 - P(\exists x.S(M, x) \lor S(x, F))
\]

Now we are stuck → Need another rule (attribute-constant ranking)!
Rule 6: Attribute ranking

**Definition**

*Attribute-constant ranking.* If $Q$ is a query that contains a relation name $R$ with attribute $A$, and there exists two unifiable atoms such that the first has constant $a$ at position $A$ and the second has a variable, substitute each occurrence of form $R(\ldots)$ by $R_1(\ldots) \lor R_2(\ldots)$, where

$$R_1 = \sigma_{A=a}(R), \quad R_2 = \sigma_{A\neq a}(R).$$

*Attribute-attribute ranking.* If $Q$ is a query that contains a relation name $R$ with attributes $A$ and $B$, substitute each occurrence of form $R(\ldots)$ by $R_1(\ldots) \lor R_2(\ldots) \lor R_3(\ldots)$, where

$$R_1 = \sigma_{A<B}(R), \quad R_2 = \sigma_{A=B}(R), \quad R_3 = \sigma_{A>B}(R).$$

*Syntactic rewrites.* For selections of form $\sigma_{A=\cdot}$, decrease the arity of the resulting relation by 1 and add an equality predicate.
Attribute-constant ranking (continues prev. example)

Example

Has both Mary seen some bird and someone seen a finch?

\[
P \left( (\exists x. S(M, x)) \land (\exists y. S(y, F)) \right) \\
= 1.7 - P \left( \exists x. S(M, x) \lor S(x, F) \right) \\
= 1.7 - P \left( \exists x. S_M(x) \lor S_{\neg M}(M, x) \lor [S_M(F) \land x = M] \lor S_{\neg M}(x, F) \right) \\
= 1.7 - P \left( \exists x. S_M(x) \lor S_M(F) \lor S_{\neg M}(x, F) \right) \\
= 1.7 - P \left( \exists x. [S_{MF}(x) \land x = F] \lor S_{MF}(x) \lor S_{MF}(x) \lor S_{\neg M}(x, F) \right) \\
= 1.7 - P \left( S_{MF}(x) \lor \exists x. S_{MF}(x) \lor S_{\neg M}(x, F) \right) \\
= 1.7 - 1 + (1 - P (S_{MF}()))(1 - P (\exists x. S_{MF}(x) \lor S_{\neg M}(x, F))) \\
= 0.7 + (1 - 0.8) \left[ \prod_{x \in \{M, S, F, T, N\}} (1 - P (S_{MF}(x) \lor S_{\neg M}(x, F))) \right] \\
= 0.7 + 0.2 \left[ \prod_{x \in \{M, S, F, T, N\}} (1 - P (S_{MF}(x))) \right] \left[ (1 - P (S_{\neg M}(x, F))) \right] \\
= 0.7 + 0.2 [11 \cdot 1 (1 - 0.2) \cdot 11 \cdot (1 - 0.3) 1 \cdot 11] \\
= 0.812
\]
The goal of attribute ranking is to establish syntactic independence and new separators by exploiting disjointness.

Example

Are there two people who like each other?

\[ \mathbb{P} ( \exists x. \exists y. \text{Likes}(x, y) \land \text{Likes}(y, x) ) \]

\[ = \mathbb{P} ( \exists x. \exists y. \) 

\[ \quad \text{(Likes}_<(x, y) \lor (\text{Likes}_=(x) \land x = y) \lor \text{Likes}_>(x, y)) \lor \) 

\[ \quad \text{(Likes}_<(y, x) \lor (\text{Likes}_=(x) \land x = y) \lor \text{Likes}_>(y, x)) \) \]

\[ = \mathbb{P} ( \exists x. \exists y. \text{Like}_<(x, y) \lor (\text{Like}_=(x) \land x = y) \lor \text{Like}_>(x, y) \lor \text{Like}_<(y, x) ) \]

\[ = \mathbb{P} ( (\exists x. \exists y. \text{Like}_<(x, y) \lor \text{Like}_>(x, y)) \lor (\exists x. \text{Like}_=(x) ) \lor (\exists x. \exists y. \text{Like}_<(y, x) ) \lor (\exists x. \text{Like}_>(x) ) ) \]

Now we can apply independent-union, then independent-project, then independent-join.
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary
Inclusion-exclusion and cancellation

Consider the query

\[ Q \leftarrow (Q_1 \lor Q_3) \land (Q_1 \lor Q_4) \land (Q_2 \lor Q_4) \]

Apply inclusion exclusion to get

\[
\mathbb{P}(Q) = \mathbb{P}(Q_1 \lor Q_3) + \mathbb{P}(Q_1 \lor Q_4) + \mathbb{P}(Q_2 \lor Q_4) \\
- \mathbb{P}(Q_1 \lor Q_3 \lor Q_4) - \mathbb{P}(Q_1 \lor Q_2 \lor Q_3 \lor Q_4) - \mathbb{P}(Q_1 \lor Q_2 \lor Q_4) \\
+ \mathbb{P}(Q_1 \lor Q_2 \lor Q_3 \lor Q_4) \\
= \mathbb{P}(Q_1 \lor Q_3) + \mathbb{P}(Q_1 \lor Q_4) + \mathbb{P}(Q_2 \lor Q_4) \\
- \mathbb{P}(Q_1 \lor Q_3 \lor Q_4) - \mathbb{P}(Q_1 \lor Q_2 \lor Q_4)
\]

One can construct cases in which \( Q_1 \lor Q_2 \lor Q_3 \lor Q_4 \) is hard, but any subset is not (e.g., consider \( H_3 \) on slide 20).

The inclusion-exclusion formula needs to be replaced by the Möbius inversion formula.
Möbius inversion formula (example)

Given a query expression of form \( Q_1 \land \ldots \land Q_k \):

1. Put the formulas \( Q_S = \bigvee_{i \in S} Q_i \), \( \emptyset \neq S \subseteq \{1, \ldots, j\} \), in a lattice (plus special element \( \hat{1} \))
2. Eliminate duplicates (equivalent formulas)
3. Use the partial order \( Q_{S_1} \geq Q_{S_2} \) iff \( Q_{S_1} \preceq Q_{S_2} \)
4. Label each node by its Möbius value

\[
\mu(\hat{1}) = 1
\]
\[
\mu(u) = - \sum_{u < w \leq \hat{1}} \mu(w)
\]

5. Use the inversion formula

\[
P(Q_1 \land \ldots \land Q_k)
= -\sum_{u < \hat{1} : \mu(u) \neq 0} \mu(u) P(Q_u)
\]

\[
P(Q) = P(Q_1 \lor Q_3) + P(Q_1 \lor Q_4) + P(Q_2 \lor Q_4)
- P(Q_1 \lor Q_3 \lor Q_4) - P(Q_1 \lor Q_2 \lor Q_4)
\]
An nondeterministic algorithm

Consider the algorithm:

1. As long as possible, apply one of the rules R1–R6
2. If all formulas are atoms, SUCCESS
3. If there is a formula that is not an atom, FAILURE

Definition

A rule is **R₆-safe** if the above algorithm succeeds.

- Order of rule application does not affect SUCCESS
- Algorithm is polynomial in size of database
  - Easy to see for independent-join, independent-union, negation, Möbius inversion formula, attribute ranking → do not depend on database
  - Independent-project increases number of queries by a factor of |ADom| → applied at most $k$ times, where $k$ is the maximum arity of a relation
How the rules fail

Example

Consider the hard query

\[ H_0 \leftarrow \exists x. \exists y. R(x) \land S(x, y) \land T(y) \]

- independent-join, independent-union, independent-project, negation, Möbius inversion formula all do not apply
- But we could rank \( S \):

\[
\begin{align*}
H_0 & \leftarrow H_{01} \lor H_{02} \lor H_{03} \\
H_{01} & \leftarrow \exists x. \exists y. R(x) \land S_<(x, y) \land T(y) \\
H_{02} & \leftarrow \exists x. R(x) \land S_= (x) \land T(x) \\
H_{03} & \leftarrow \exists x. \exists y. R(x) \land S_>(y, x) \land T(y)
\end{align*}
\]

- Now we are stuck at \( H_{01} \) and \( H_{03} \)
Dichotomy theorem for UCQ

- Safety is a syntactic property
- Tractability is a semantic property
- What is their relationship?

Theorem (Dalvi and Suciu, 2010)

For any UCQ query Q, one of the following holds:

- Q is $R_6$-safe, or
- the data complexity of Q is hard for #$P$.

- No queries of “intermediate” difficulty
- Can check for tractability in time polynomial in database size (can be done by assuming an active domain of size 1)
- Query complexity is unknown (Möbius inversion formula)
- For $RC$, completeness/dichotomy unknown

We can handle all safe UCQ queries!
Overview of extensional plans

Can we evaluate safe queries directly in an RDBMS?

- Extensional query evaluation
  - Based on the query expression
  - Uses rules to break query into simpler pieces
  - For UCQ, detects whether queries are tractable or intractable

- Extensional operators
  - Extend relational operators by probability computation
  - Standard database algorithms can be used

- Extensional plans
  - Can be safe (correct) or unsafe (incorrect)
  - For tractable UCQ queries, we can always produce a safe plan
  - Plan construction based on R₆ rules
  - Can be written in SQL (though not “best” approach)
  - Enables scalable query processing on probabilistic databases
Definition
Annotate each tuple by its probability. The operators

- Independent join ($\Join^i$)
- Independent project ($\pi^i$)
- Independent union ($\cup^i$)
- Construction / selection / renaming

correspond to the positive $K$-relational algebra over $([0, 1], 0, 1, \oplus, \cdot)$, where $p_1 \oplus p_2 = 1 - (1 - p_1)(1 - p_2)$.

(Union needs to be replaced by outer join for non-matching schemas; see Sucio, Olteneau, Ré, Koch, 2011.)

([0, 1], 0, 1, \oplus, \cdot) is not a semiring $\rightarrow$ unsafe plans!
Example plans

Who incriminates someone who has an alibi?

\[ Q_1(w) \leftarrow \exists s. \exists x. \text{Incriminates}(w,s) \land \text{Alibi}(s, x) \]

\[ Q_2(w) \leftarrow \exists s. \text{Incriminates}(w, s) \land \exists x. \text{Alibi}(s, x) \]

Plan 1
Incorrect (unsafe)

Plan 2
Correct (safe)

Not all plans are safe!
Weighted sum

How to deal with the Möbius inversion formula?

Definition

The *weighted sum* of relations $R_1, \ldots, R_k$ with parameters $\mu_1, \ldots, \mu_k$ is given by:

$$
\left( \sum U (R_1, \ldots, R_k) \right) \mu_1, \ldots, \mu_k = R_1 \bowtie \cdots \bowtie R_k
$$

$$
\left( \sum U (R_1, \ldots, R_k) \right) (t) = \mu_1(R_1(t)) + \cdots + \mu_k(R_k(t))
$$

Intuitively,

- Computes the natural join
- Sums up the weighted probabilities of joining tuples
Weighted sum (example)

Example

Consider relations/subqueries $V_1(A, B)$ and $V_2(A, C)$ and the query:

$$Q(x, y, z) \leftarrow V_1(x, y) \land V_2(x, z)$$

Suppose we apply the Möbius inversion formula to get:

- $Q_1(x, y) = V_1(x, y)$ with $\mu_1 = 1$
- $Q_2(x, z) = V_2(x, z)$ with $\mu_2 = 1$
- $Q_3(x, y, z) = V_1(x, y) \lor V_2(x, z)$ with $\mu_3 = -1$

We obtain:

$$\sum_{\{A, B, C\}}^{1, 1, -1} (Q_1, Q_2, Q_3)[] = Q_1 \bowtie Q_2 \bowtie Q_3 = V_1 \bowtie V_2$$

$$\sum_{\{A, B, C\}}^{1, 1, -1} (Q_1, Q_2, Q_3) = \left\{ (t, p_{t_1} + p_{t_2} - p_{t_3}) : t[AB] = t_1 \in Q_1, t[AC] = t_2 \in Q_2, t[ABC] = t_3 \in Q_3 \right\}$$
Complement

How to deal with negation?

Definition

The *complement* of a deterministic relation $R$ of arity $k$ is given by

$$C(R) = \left\{ (t, 1 - \mathbb{P}(t \in R)) : t \in \text{ADom}^k \right\}.$$  

In practice, every complement operation can be replaced by difference (since queries are domain-independent).

Example

- **Query:** $Q \leftarrow R(x) \land \lnot S(x)$
- **Result:** $R^{-i} S = \left\{ (t, \mathbb{P}(t \in R)(1 - \mathbb{P}(t \in S))) : t \in R \right\}$
Definition
A query plan for $Q$ is safe if it computes the correct probabilities for all input databases.

Theorem
There is an algorithm $A$ that takes in a query $Q$ and outputs either FAIL of a safe plan for $Q$. If $Q$ is a UCQ query, $A$ fails only if $Q$ is intractable.

- Key idea: Apply rules R1–R6, but produce a query plan instead of computing probabilities
- Extension to non-Boolean queries: treat head variables as “constants”
- Ranking step produces “views” that are treated as base tables
Computation of safe plans (2)

1: if $Q = Q_1 \land Q_2$ and $Q_1, Q_2$ are syntactically independent then
2: return $\text{plan}(Q_1) \searrow^i \text{plan}(Q_2)$
3: end if
4: if $Q = Q_1 \lor Q_2$ and $Q_1, Q_2$ are syntactically independent then
5: return $\text{plan}(Q_1) \cup^i \text{plan}(Q_2)$
6: end if
7: if $Q(x) = \exists z. Q_1(x, z)$ and $z$ is a separator variable then
8: return $\pi^i_x(\text{plan}(Q_1(x, z)))$
9: end if
10: if $Q = Q_1 \land \ldots \land Q_k$, $k \geq 2$ then
11: Construct CNF lattice $Q'_1, \ldots, Q'_m$
12: Compute Möbius coefficients $\mu_1, \ldots, \mu_m$
13: return $\sum_{\mu_1, \ldots, \mu_m} \text{plan}(Q'_1), \ldots, \text{plan}(Q'_m))$
14: end if
15: if $Q = \neg Q_1$ then
16: return $C(\text{plan} Q_1)$
17: end if
18: if $Q(x) = R(x)$ where $R$ is a base table (possibly ranked) then
19: return $R(x)$
20: end if
21: otherwise FAIL
Computation of safe plans (example)

\[ Q(w) \leftarrow \exists s.\exists x.\text{Incriminates}(w, s) \land \text{Alibi}(s, x) \]

1. Apply independent-project to \( Q \) on \( s \)
   - \( Q_1(w, s) \leftarrow \exists x.\text{Incriminates}(w, s) \land \text{Alibi}(s, x) \)

2. \( x \) is not a root variable in \( Q_1 \) → push \( \exists x: \)
   - \( Q_2(w, s) \leftarrow \text{Incriminates}(w, s) \land \exists x.\text{Alibi}(s, x) \)

3. Apply independent-join to \( Q_2 \)
   - \( Q_3(w, s) \leftarrow \text{Incriminates}(w, s) \)
   - \( Q_4(s) \leftarrow \exists x.\text{Alibi}(s, x) \)

4. \( Q_3 \) is an atom

5. Apply independent-project to \( Q_4 \) on \( x \)
   - \( Q_5(s, x) = \text{Alibi}(s, x) \)

6. \( Q_5 \) is an atom
Safe plans with PostgreSQL (example)

\[ Q(w) \leftarrow \exists s. \exists x. \text{Incriminates}(w, s) \land \text{Alibi}(s, x) \]

- \( Q_4 \leftarrow \pi^i_{\text{Suspect}}(\text{Alibi}) \)
- \( Q_2 \leftarrow \text{Incriminates} \bowtie^i_{\text{Suspect}} Q_4 \)
- \( Q \leftarrow \pi^i_{\text{Witness}}(Q_2) \)

```sql
SELECT Witness, 1-PRODUCT(1-P) AS P
FROM (
    SELECT Witness, Incriminates.Suspect,
           Incriminates.P * Q4.P AS P
    FROM Incriminates,
    (SELECT Suspect, 1-PRODUCT(1-P) AS P
     FROM Alibi
     GROUP BY Suspect
    ) AS Q4
    WHERE Incriminates.Suspect = Q4.Suspect
) AS Q2
GROUP BY Witness
```
Deterministic tables

- Often: Mix of probabilistic and deterministic tables
- Naive approach: Assign probability 1 to tuples in a deterministic table
  → Suboptimal: Some tractable queries are missed!

**Example**

- If $T$ is known to be deterministic, the query
  
  $$Q \leftarrow R(x), S(x, y), T(y)$$

  becomes tractable!
- Why? $S \bowtie T$ now is a tuple-independent table!
- We can use the safe plan
  
  $$\pi^i_\emptyset \left[ R(x) \bowtie^i_x (S(x, y) \bowtie^i_y T(y)) \right]$$

Additional information about the nature of the tables (e.g., deterministic, tuple-independent with keys, BID tables) can help extensional query processing.
Outline

1 Primer: Relational Calculus
2 The Query Evaluation Problem
3 Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4 Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5 Summary
Overview

Given a query $Q(x)$, a TI database $D$; for each output tuple $t$

1. Compute the lineage $\Phi = \Phi^D_{Q(t)}$
   - $|\Phi| = O(|\text{ADom}|^m)$, where $m$ is the number of variables in $\Phi$
   - Data complexity is polynomial time
   - Difference to extensional query evaluation: $|\Phi|$ depends on input
     $\rightarrow$ rules exponential in $|\Phi|$ also exponential in the size of the input!

2. Compute the probability $\mathbb{P}(\Phi)$
   - Intensional query evaluation $\approx$ probability computation on propositional formulas
   - Studied in verification and AI communities
   - Different approaches: rule-based evaluation, formula compilation, approximation

Can deal with hard queries.
Example (tractable query)

\[ q(h) \leftarrow \exists n. \exists c. \text{Hotel}(h, n, c) \land \exists r. \exists t. \exists p. \text{Room}(r, h, t, p) \land (p > 500 \lor t = 'suite') \]

<table>
<thead>
<tr>
<th>Room (R)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RoomNo</td>
<td>Type</td>
<td>HotelNo</td>
</tr>
<tr>
<td>R1</td>
<td>Suite</td>
<td>H1</td>
</tr>
<tr>
<td>R2</td>
<td>Single</td>
<td>H1</td>
</tr>
<tr>
<td>R3</td>
<td>Double</td>
<td>H1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hotel (H)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HotelNo</td>
<td>Name</td>
<td>City</td>
</tr>
<tr>
<td>H1</td>
<td>Hilton</td>
<td>SB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ExpensiveHotels</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HotelNo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Phi = X_4 \land (X_1 \lor X_2) \]

\[ P(\Phi) = P(X_4) \cdot [1 - (1 - P(X_1)) \cdot (1 - P(X_2))] \]

E.g., \[ P(X_i) = \frac{1}{2} \] for all \( i \rightarrow P(\Phi) = 0.375 \]
Example (intractable query)

Example

\begin{align*}
H_0 & \iff \exists x \exists y. R(x), S(x, y), T(y) \\
\Phi & = X_2 Y_1 \lor X_3 Y_2 \\
\mathbb{P}(\Phi) & = 1 - (1 - \mathbb{P}(X_2) \mathbb{P}(Y_1))(1 - \mathbb{P}(X_3) \mathbb{P}(Y_2)) = 0.4375 \\
\text{Model counting: } \#\Phi & = 2^6 \mathbb{P}(\Phi) = 28 \\
\text{Bipartite vertex cover: } \#\Psi & = 2^6 - \#\Phi = 36 = 2 \cdot 3 \cdot 3 \cdot 2
\end{align*}
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary
Overview of rule-based intensional query evaluation

- Break the lineage formula into “simpler” formulas
- By applying one of the rules
  1. Independent-and
  2. Independent-or
  3. Disjoint-or
  4. Negation
  5. Shannon expansion
- Rules work on lineage, not on query → data dependent
- Rules *always* succeed
- Rule 5 may lead to exponential blowup

Can be used on any query but data complexity can be exponential. However, depending on the database, even a hard query might be “easy” to evaluate.
Support

Definition

For a propositional formula $\Phi$, denote by $V(\Phi)$ the set of variables that occur in $\Phi$. Denote by $\text{Var}(\Phi)$ the set of variables on which $\Phi$ depends; $\text{Var}(\Phi)$ is called the support of $\Phi$. $X \in \text{Var}(\Phi)$ iff there exists an assignment $\theta$ to all variables but $X$ and constants $a \neq b$ such that $\Phi[\theta \cup \{ X \mapsto a \}] \neq \Phi[\theta \cup \{ X \mapsto b \}]$.

Example

$\Phi = X \lor (Y \land Z)$
- $V(\Phi) = \{ X, Y, Z \}$
- $\text{Var}(\Phi) = \{ X, Y, Z \}$

$\Phi = Y \lor (X \land Y) \equiv Y$
- $V(\Phi) = \{ X, Y \}$
- $\text{Var}(\Phi) = \{ Y \}$
Syntactic independence

**Definition**

\( \Phi_1 \) and \( \Phi_2 \) are *syntactically independent* if they have disjoint support, i.e.,
\[ \text{Var}(\Phi_1) \cap \text{Var}(\Phi_2) = \emptyset. \]

**Example**

\( \Phi_1 = X \quad \Phi_2 = Y \quad \Phi_3 = \neg X \neg Y \lor XY \)

- \( \Phi_1 \) and \( \Phi_2 \) are syntactically independent
- All other combinations are not

Checking for syntactic independence is co-NP-complete in general.

**Practical approach:**

**Proposition**

*A sufficient condition for syntactic independence is* \( V(\Phi_1) \cap V(\Phi_2) = \emptyset. \)
Probabilistic independence

Proposition

If $\Phi_1, \Phi_2, \ldots, \Phi_k$ are pairwise syntactically independent, then the probabilistic events $\Phi_1, \Phi_2, \ldots, \Phi_k$ are independent.

Note that pairwise probabilistic independence does not imply probabilistic independence!

Example

$\Phi_1 = X$, $\Phi_2 = Y$, $\Phi_3 = \neg X \neg Y \lor XY$

- $\Phi_1$ and $\Phi_2$ are probabilistically independent
- $\Phi_1, \Phi_2, \Phi_3$ are not pairwise syntactically independent

Assume $\mathbb{P}(X) = \mathbb{P}(Y) = \frac{1}{2}$

- $\Phi_1, \Phi_2, \Phi_3$ are pairwise independent
- $\Phi_1, \Phi_2, \Phi_3$ are not independent!
Outline

1 Primer: Relational Calculus
2 The Query Evaluation Problem
3 Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4 Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5 Summary
**Rules 1 and 2: independent-and, independent-or**

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $\Phi_1$ and $\Phi_2$ be two syntactically independent propositional formulas:</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
P(\Phi_1 \land \Phi_2) &= P(\Phi_1) \cdot P(\Phi_2) \quad (\text{independent-and}) \\
P(\Phi_1 \lor \Phi_2) &= 1 - (1 - P(\Phi_1))(1 - P(\Phi_2)) \quad (\text{independent-or})
\end{align*}
\]
Independent-and, independent-or (example)

<table>
<thead>
<tr>
<th>Witness</th>
<th>Suspect</th>
<th>Incriminates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Paul</td>
<td>$X_1 (p_1)$</td>
</tr>
<tr>
<td>Mary</td>
<td>John</td>
<td>$X_2 (p_2)$</td>
</tr>
<tr>
<td>Susan</td>
<td>John</td>
<td>$X_3 (p_3)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suspect</th>
<th>Claim</th>
<th>Alibi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul</td>
<td>Cinema</td>
<td>$Y_1 (q_1)$</td>
</tr>
<tr>
<td>Paul</td>
<td>Friend</td>
<td>$Y_2 (q_2)$</td>
</tr>
<tr>
<td>John</td>
<td>Bar</td>
<td>$Y_3 (q_3)$</td>
</tr>
</tbody>
</table>

$Q(w) \leftarrow \exists s. \exists x. \text{Incriminates}(w, s) \land \text{Alibi}(s, x)$

- $\Phi_S = X_3 Y_3$
  - Independent-and: $P(\Phi_S) = p_3 q_3$

- $\Phi_M = X_1(Y_1 \lor Y_2) \lor X_2 Y_3$
  - Independent-or:
    - $P(\Phi_M) = 1 - (1 - P(X_1(Y_1 \lor Y_2)))(1 - P(X_2 Y_3))$
  - Independent-and: $P(X_2 Y_3) = p_2 q_3$
  - Independent-and:
    - $P(X_1(Y_1 \lor Y_2)) = p_1 P(Y_1 \lor Y_2)$
  - Independent-or:
    - $P(Y_1 \lor Y_2) = 1 - (1 - q_1)(1 - q_2)$
    - $P(\Phi_M) = 1 - [1 - p_1(1 - (1 - q_1)(1 - q_2))](1 - p_2 q_3)$
Rule 3: Disjoint-or

Definition
Two propositional formulas $\Phi_1$ and $\Phi_2$ are *disjoint* if $\Phi_1 \land \Phi_2$ is not satisfiable.

Definition
If $\Phi_1$ and $\Phi_2$ are disjoint:

$$P(\Phi_1 \lor \Phi_2) = P(\Phi_1) + P(\Phi_2) \quad (\text{disjoint-or})$$

Example

- $P(X) = 0.2$; $P(Y) = 0.7$
- $\Phi_1 = XY$; $P(XY) = P(X)P(Y) = 0.14$
- $\Phi_2 = \neg X$; $P(\neg X) = 0.8$
- $P(\Phi_1 \lor \Phi_2) = P(\Phi_1) + P(\Phi_2) = 0.94$

Checking for disjointness is NP-complete in general. But disjoint-or will play a major role for Shannon expansion.
### Rule 4: Negation

#### Definition

\[ P(\neg \Phi) = 1 - P(\Phi) \] (negation)

#### Example

- \( P(X) = 0.2; \ P(Y) = 0.7 \)
- \( P(XY) = P(X)P(Y) = 0.14 \)
- \( P(\neg(XY)) = 1 - 0.14 = 0.86 \)
Shannon expansion

**Definition**

The *Shannon expansion* of a propositional formula $\Phi$ w.r.t. a variable $X$ with domain $\{a_1, \ldots, a_m\}$ is given by:

$$\Phi \equiv (\Phi[X \mapsto a_1] \land (X = a_1)) \lor \ldots \lor (\Phi[X \mapsto a_m] \land (X = a_m))$$

**Example**

- $\Phi = XY \lor XZ \lor YZ$
- $\Phi \equiv (\Phi[X \mapsto \text{TRUE}] \land X) \lor (\Phi[X \mapsto \text{FALSE}] \land \neg X)$
  $$= (Y \lor Z)X \lor YZ \neg X$$

In the Shannon expansion rule, every $\land$ is an independent-and; every $\lor$ is a disjoint-or.
Rule 5: Shannon expansion

Definition

Let $\Phi$ be a propositional formula and $X$ be a variable:

$$P(\Phi) = \sum_{a \in \text{dom}(X)} P(\Phi[X \mapsto a]) P(X = a) \quad (\text{Shannon expansion})$$

Example

- $\Phi = XY \lor XZ \lor YZ$
- $P(\Phi) = P(Y \lor Z) P(X) + P(YZ) P(\neg X)$

- Can always be applied
- Effectively eliminates $X$ from the formula
- But may lead to exponential blowup!
Shannon expansion (example)

<table>
<thead>
<tr>
<th>Witness</th>
<th>Suspect</th>
<th>Incriminates</th>
<th>Alibi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Paul</td>
<td>$X_1$ ($p_1$)</td>
<td>Paul</td>
</tr>
<tr>
<td>Mary</td>
<td>John</td>
<td>$X_2$ ($p_2$)</td>
<td>Cinema</td>
</tr>
<tr>
<td>Susan</td>
<td>John</td>
<td>$X_3$ ($p_3$)</td>
<td>John</td>
</tr>
</tbody>
</table>

$Q(w) \leftarrow \exists s. \exists x. \text{Incriminates}(w, s) \land \text{Alibi}(s, x)$

$$\Phi_M = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$$

1. Independent-or: $$P(\Phi_M) = 1 - (1 - P(X_1 Y_1 \lor X_1 Y_2))(1 - P(X_2 Y_3))$$

2. Independent-and: $$P(X_2 Y_3) = p_2 q_3$$

3. Shannon expansion: $$P(X_1 Y_1 \lor X_1 Y_2) = P(Y_1 \lor Y_2) P(X_1) + P(\text{FALSE}) P(\neg X_1)$$

4. Independent-or: $$P(Y_1 \lor Y_2) = 1 - (1 - q_1)(1 - q_2)$$

5. $$P(\Phi_M) = 1 - [1 - p_1(1 - (1 - q_1)(1 - q_2))](1 - p_2 q_3)$$

The intensional rules work on all plans!
A non-deterministic algorithm

1: if $\Phi = \Phi_1 \land \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then
2:   return $P(\Phi_1) \cdot P(\Phi_2)$
3: end if
4: if $\Phi = \Phi_1 \lor \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then
5:   return $1 - (1 - P(\Phi_1))(1 - P(\Phi_2))$
6: end if
7: if $\Phi = \Phi_1 \lor \Phi_2$ and $\Phi_1, \Phi_2$ are disjoint then
8:   return $P(\Phi_1) + P(\Phi_2)$
9: end if
10: if $\Phi = \neg \Phi_1$ then
11:   return $1 - P(\Phi_1)$
12: end if
13: Choose $X \in \text{Var}(\Phi)$
14: return $\sum_{a \in \text{dom}(X)} P(\Phi[X \mapsto a])P(X = a)$

Should be implemented with dynamic programming to avoid evaluating the same subformula multiple times.
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
     - Approximation Techniques
5. Summary
Materialized views in TID databases (1)

- TID databases complete only with views
- How to deal with views in a PDBMS?
  1. Store just the view definition
  2. Store the view result and probabilities
  3. Store the view result and lineage
  4. Store the view results and “compiled lineage”
- Trade-off between precomputation and query cost (just as in DBMS)

Example (ExpensiveHotel view)

\[ q(h) \leftarrow \exists n. \exists c. \text{Hotel}(h, n, c) \land \exists r. \exists t. \exists p. \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'}) \]

<table>
<thead>
<tr>
<th>Room (R)</th>
<th>Hotel (H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoomNo</td>
<td>HotelNo</td>
</tr>
<tr>
<td>R1</td>
<td>H1</td>
</tr>
<tr>
<td>R2</td>
<td>H1</td>
</tr>
<tr>
<td>R3</td>
<td>H1</td>
</tr>
<tr>
<td>ExpensiveHotels</td>
<td></td>
</tr>
<tr>
<td>HotelNo</td>
<td>0.375</td>
</tr>
<tr>
<td>H1</td>
<td>(X_1)</td>
</tr>
<tr>
<td>ExpensiveHotels</td>
<td></td>
</tr>
<tr>
<td>HotelNo</td>
<td>(X_4 \land (X_1 \lor X_2))</td>
</tr>
<tr>
<td>H1</td>
<td>(X_4 \land (X_1 \lor X_2))</td>
</tr>
<tr>
<td>ExpensiveHotels</td>
<td></td>
</tr>
<tr>
<td>HotelNo</td>
<td>(X_4 \land (X_1 \lor X_2))</td>
</tr>
<tr>
<td>H1</td>
<td>(X_4 \land (X_1 \lor X_2))</td>
</tr>
</tbody>
</table>
Materialized views in TID databases (2)

Example (Continued)

Consider the query

\[ q(h) \leftarrow \exists c. \text{ExpensiveHotel}(h), \text{Hotel}(h, 'Hilton', c), \]

which asks for expensive Hilton hotels using a view. Can we answer this query when ExpensiveHotel is a precomputed materialized view?

<table>
<thead>
<tr>
<th>Hotel (H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HotelNo</td>
</tr>
<tr>
<td>H1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ExpensiveHotels</th>
<th>ExpensiveHotels</th>
<th>ExpensiveHotels</th>
</tr>
</thead>
<tbody>
<tr>
<td>HotelNo</td>
<td>HotelNo</td>
<td>HotelNo</td>
</tr>
<tr>
<td>H1</td>
<td>H1</td>
<td>H1</td>
</tr>
<tr>
<td>( X_4 \land (X_1 \lor X_2) )</td>
<td>0.375</td>
<td>( X_4 \land^i (X_1 \lor^i X_2) )</td>
</tr>
</tbody>
</table>

Yes, combine lineages

No, dependency between ExpensiveHotels and Hotels lost

Yes, combine “compiled lineages” \( \rightarrow \) Need to be able to combine compiled lineages efficiently!

<table>
<thead>
<tr>
<th>ExpensiveHiltons</th>
<th>ExpensiveHiltons</th>
</tr>
</thead>
<tbody>
<tr>
<td>HotelNo</td>
<td>HotelNo</td>
</tr>
<tr>
<td>H1</td>
<td>H1</td>
</tr>
<tr>
<td>([X_4 \land (X_1 \lor X_2)] \land X_4)</td>
<td>(X_4 \land^i (X_1 \lor X_2))</td>
</tr>
</tbody>
</table>
Query compilation

- “Compile” \( \Phi \) into a Boolean circuit with certain desirable properties
- \( \mathbb{P}(\Phi) \) can be computed in linear time in the size of the circuit
  - Many other tasks can be solved in polynomial time
  - E.g., combining formulas \( \Phi_1 \land \Phi_2 \) (even when not independent!)
  - Key application in PDBMS: Compile materialized views
- Tractable compilation = circuit of size polynomial in database
  \( \implies \) Implies tractable computation of \( \mathbb{P}(\Phi) \) (converse may not be true)
- Compilation targets
  1. RO (read-once formula)
  2. OBDD (ordered binary decision diagram)
  3. FBDD (free binary decision diagram)
  4. d-DNF (deterministic-decomposable normal form)

Goals: (1) Reusability. (2) Understand complexity of intensional QE.
Restricted Boolean circuit (RBC)

- Rooted, labeled DAG
- All variables are Boolean
- Each node (called gate) represents a propositional formula \( \Psi \)
- Let \( \Psi \) be represented by a gate with children representing \( \Psi_1, \ldots, \Psi_n \); we consider the following gates & restrictions:
  - **Independent-and** (\( \land^i \)): \( \Psi_1, \ldots, \Psi_n \) are syntactically independent
  - **Independent-or** (\( \lor^i \)): \( \Psi_1, \ldots, \Psi_n \) syntactically independent
  - **Disjoint-or** (\( \lor^d \)): \( \Psi_1, \ldots, \Psi_n \) are disjoint
  - **Not** (\( \neg \)): single child, represents \( \neg \psi \)
  - **Conditional gate** (\( X \)): two children representing \( X \land \Psi_1 \) and \( \neg X \land \Psi_2 \), where \( X \notin \text{Var}(\Psi_1) \) and \( X \notin \text{Var}(\Psi_2) \)
  - **Leaf node** (0, 1, \( X \)): represents FALSE, TRUE, \( X \)

The different compilation targets restrict which and where gates may be used.
Restricted Boolean circuit (example)

Example

Who incriminates someone who has an alibi?
Lineage of unsafe plan: $\Phi_M = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$

“Documents” the non-deterministic algorithm for intensional query evaluation.
Deterministic-decomposable normal form (d-DNF)

- Restricted to gates: $\land^i$, $\lor^d$, $\neg$
  - $\land^i$-gates are called decomposable (D)
  - $\lor^d$-gates are called deterministic (d)

**Example**

$\Phi = XYU \lor XYZ \neg U$

![Diagram of the example expression]
RBC and d-DNF

Theorem

Every RBC with \( n \) gates can be transformed into an equivalent d-DNF with at most \( 5n \) gates, a polynomial increase in size.

Proof.

We are not allowed to use \( \lor^i \) and conditional nodes. Apply the transformations:

A \( \lor^i \)-node is replaced by 4 new nodes. A conditional node is replaced by (at most) 5 new nodes.
Application: knowledge compilation

- Tries to deal with intractability of propositional reasoning
- Key idea
  1. Slow offline phase: Compilation into a target language
  2. Fast online phase: Answers in polynomial time
  → Offline cost amortizes over many online queries
- Key aspects
  ▶ Succinctness of target language (d-DNF, FBDD, OBDD, ...)
  ▶ Class of queries that can be answered efficiently once compiled (consistency, validity, entailment, implicants, equivalence, model counting, probability computation, ...)
  ▶ Class of transformations that can be performed efficiently once compiled (∧, ∨, ¬, conditioning, forgetting, ...)
- How to pick a target language?
  1. Identify which queries/transformations are needed
  2. Pick the *most succinct* language

Which queries admit polynomial representation in which target language?

Darwiche and Marquis, 2002
Free binary decision diagram (FBDD)

- Restricted to conditional gates
- *Binary decision diagram*: Each node decides on the value of a variable
- *Free*: Each variable occurs only on every root-leaf path

Example

Who incriminates someone who has an alibi?

Lineage of safe plan: $\Phi_M = X_1(Y_1 \lor Y_2) \lor X_2 Y_3$
Ordered binary decision diagram (OBDD)

- An ordered FBDD, i.e.,
  - Same ordering of variables on each root-leaf path
  - Omissions are allowed

Example

The FBDD on slide 88 is an OBDD with ordering $X_1, Y_1, Y_2, X_2, Y_3$.

Theorem

*Given two ODDBs $\Psi_1$ and $\Psi_2$ with a common variable order, we can compute an ODDB for $\Psi_1 \land \Psi_2$, $\Psi_1 \lor \Psi_2$, or $\neg \Psi_1$ in polynomial time.*

*Note that $\Psi_1$ and $\Psi_2$ do not need to be independent or disjoint.*

(Many other results of this kind exist. Many BDD software packages exist, e.g., BuDDy, JDD, CUDD, CAL).
Read-once formulas (RO)

Definition
A propositional formula $\Phi$ is read-once (or repetition-free) if there exists a formula $\Phi'$ such that $\Phi \equiv \Phi'$ and every variable occurs at most once in $\Phi'$.

Example
- $\Phi = X_1 \lor X_2 \lor X_3 \rightarrow$ read-once
- $\Phi = X_1Y_1 \lor X_1Y_2 \lor X_2Y_3 \lor X_2Y_4 \lor X_2Y_5$
  $\quad \Rightarrow \Phi' = X_1(Y_1 \lor Y_2) \lor X_2(Y_3 \lor Y_4 \lor Y_5) \rightarrow$ read-once
- $\Phi = XY \lor XU \lor YU \rightarrow$ not read-once

Theorem
If $\Phi$ is given as a read-once formula, we can compute $\mathbb{P}(\Phi)$ in linear time.

Proof.
All $\land$'s and $\lor$'s are independent, and negation is easily handled.
When is a formula read-once? (1)

**Definition**

Let $\Phi$ be given in DNF such that no conjunct is a strict subset of some other conjunct. $\Phi$ is *unate* if every propositional variable $X$ occurs either only positively or negatively. The *primal graph* $G(V, E)$ where $V$ is the set of propositional variables in $\Phi$ and there is an edge $(X, Y) \in E$ if $X$ and $Y$ occur together in some conjunct.

**Example**

- Unate: $XY \lor \neg ZX$
- Not unate: $XY \lor Z \neg X$

\[
\begin{align*}
    & XU \lor XV \lor YU \lor YV \\
    & XY \lor YU \lor UV \\
    & XY \lor XU \lor YU
\end{align*}
\]
When is a formula read-once? (2)

**Definition**

A primal graph $G$ for $\Phi$ is $P_4$-free if no induced subgraph is isomorphic to $P_4$ ($\bigcirc\bigcirc\bigcirc\bigcirc$). $G$ is normal if for every clique in $G$, there is a conjunct in $\Phi$ that contains all of the clique’s variables.

**Example**

<table>
<thead>
<tr>
<th>Formula</th>
<th>Graph</th>
<th>$P_4$-free</th>
<th>Normal</th>
<th>Read-once</th>
</tr>
</thead>
<tbody>
<tr>
<td>$XU \lor XV \lor YU \lor YV$</td>
<td><img src="image1" alt="Graph" /></td>
<td>$P_4$-free</td>
<td>Normal</td>
<td>Read-once</td>
</tr>
<tr>
<td>$XY \lor YU \lor UV$</td>
<td><img src="image2" alt="Graph" /></td>
<td>Not $P_4$-free</td>
<td>Normal</td>
<td>Not read-once</td>
</tr>
<tr>
<td>$XY \lor XU \lor YU$</td>
<td><img src="image3" alt="Graph" /></td>
<td>$P_4$-free</td>
<td>Not normal</td>
<td>Not read-once</td>
</tr>
</tbody>
</table>

**Theorem**

A unate formula is read-once iff it is $P_4$-free and normal.
Query compilation hierarchy

Denote by $\mathcal{L}(\mathcal{T})$ the class of queries from $\mathcal{L}$ that can be compiled efficiently to target $\mathcal{T}$. The following relationships hold for UCQ-queries:
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
     - Approximation Techniques
5. Summary
Why approximation?

- Exact inference may require exponential time $\rightarrow$ expensive
- Often absolute probability values of little interest; ranking desired $\rightarrow$ Good approximations of $P(\Phi)$ suffice

Desiderata

- (Provably) low approximation error
- Efficient
- Polynomial in database size
- Anytime algorithm (gradual improvement)

Approaches

- Probability intervals
- Monte-Carlo approximation

We will show: Approximation is tractable for all $RA$-queries w.r.t. absolute error and for all $UCQ$-queries w.r.t. relative error!
**Theorem**

Let $\Phi_1$ and $\Phi_2$ be propositional formulas. Then,

\[
\max(P(\Phi_1), P(\Phi_2)) \leq P(\Phi_1 \lor \Phi_2) \leq \min(P(\Phi_1) + P(\Phi_2), 1)
\]

\[
\max(0, P(\Phi_1) + P(\Phi_2) - 1) \leq P(\Phi_1 \land \Phi_2) \leq \min(P(\Phi_1), P(\Phi_2)).
\]

*via inclusion-exclusion*

*Boole's inequality / union bound*

**Example**

Border cases:

1. $P(\Phi_1 \lor \Phi_2)$: 1
2. $P(\Phi_1 \land \Phi_2)$: 0
3. $P(\Phi_1) + P(\Phi_2)$: $P(\Phi_1)$
4. $P(\Phi_1) + P(\Phi_2) - 1$: $P(\Phi_2)$
Computation of probability intervals

**Theorem**

Let $\Phi_1$ and $\Phi_2$ be propositional formulas with bounds $[L_1, U_1]$ and $[L_2, U_2]$, respectively. Then,

- $\Phi_1 \lor \Phi_2$: $[L, U] = [\max(L_1, L_2), \min(U_1 + U_2, 1)]$
- $\Phi_1 \land \Phi_2$: $[L, U] = [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$
- $\neg \Phi_1$: $[L, U] = [1 - U_1, 1 - L_1]$

**Example (Does Mary incriminate someone who has an alibi?)**

<table>
<thead>
<tr>
<th>Incriminates</th>
<th>Alibi</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Phi = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$</td>
<td>$X_1 X_2$</td>
</tr>
<tr>
<td>$X_1 Y_1$ : $[0.75, 0.85]$</td>
<td>$Y_1 Y_2 Y_3$</td>
</tr>
<tr>
<td>$X_1 Y_2$ : $[0.65, 0.75]$</td>
<td>$X_1 X_2$</td>
</tr>
<tr>
<td>$X_2 Y_3$ : $[0.45, 0.65]$</td>
<td>$X_1$ $X_2$</td>
</tr>
<tr>
<td>$X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$ : $[0.75, 1]$</td>
<td>$X_1 X_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness</th>
<th>Suspect</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Paul</td>
<td>0.9</td>
</tr>
<tr>
<td>Mary</td>
<td>John</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suspect</th>
<th>Claim</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul</td>
<td>Cinema</td>
<td>0.85</td>
</tr>
<tr>
<td>Paul</td>
<td>Friend</td>
<td>0.75</td>
</tr>
<tr>
<td>John</td>
<td>Bar</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Bounds can be computed in linear time in size of $\Phi$. 

97 / 119
Probability intervals and intensional query evaluation

1. if $\Phi = \Phi_1 \land \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then return $[L, U] = [L_1 \cdot L_2, U_1 \cdot U_2]$
2. end if
3. if $\Phi = \Phi_1 \lor \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then return $[L, U] = [L_1 \oplus L_2, U_1 \oplus U_2]$
4. end if
5. if $\Phi = \Phi_1 \lor \Phi_2$ and $\Phi_1, \Phi_2$ are disjoint then return $[L, U] = [L_1 + L_2, \min(U_1 + U_2, 1)]$
6. end if
7. if $\Phi = \neg \Phi_1$ then return $[L, U] = [1 - U_1, 1 - L_1]$  
8. end if
9. Choose $X \in \text{Var}(\Phi)$
10. Shannon expansion to $\Phi = \bigvee_i \Phi_i \land (X = a_i)$
11. return $[L, U] = [\sum_i L_i \mathbb{P}(X = a_i), \min(\sum_i U_i \mathbb{P}(X = a_i), 1)]$

Independence and disjointness allow for tighter bounds.
Example

<table>
<thead>
<tr>
<th>Witness</th>
<th>Suspect</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Paul</td>
<td>0.9</td>
</tr>
<tr>
<td>Mary</td>
<td>John</td>
<td>0.8</td>
</tr>
</tbody>
</table>

\( \Phi = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \)

- \( X_1 Y_1 : [0.75, 0.85] \)
- \( X_1 Y_2 : [0.65, 0.75] \)
- \( X_2 Y_3 : [0.45, 0.65] \)
- \( \Phi : [0.75, 1] \)
Discussion

- Incremental construction of RBC circuit
- If all leaf nodes are atomic, computes exact probability
- If some leaf nodes are not atomic, computes probability bounds
- Anytime algorithm (makes incremental progress)
- Can be stopped as soon as bounds become accurate enough
  - Absolute $\epsilon$-approximation: $U - L \leq 2\epsilon \rightarrow \text{choose } \hat{p} \in [U - \epsilon, L + \epsilon]$
  - Relative $\epsilon$-approximation:
    \[(1 - \epsilon)U \leq (1 + \epsilon)L \rightarrow \text{choose } \hat{p} \in [(1 - \epsilon)U, (1 + \epsilon)L]\]
- But: no apriori runtime bounds!

Definition

A value $\hat{p}$ is an absolute $\epsilon$-approximation of $p = \mathbb{P}(\Phi)$ if

$$p - \epsilon \leq \hat{p} \leq p + \epsilon;$$

it is an relative $\epsilon$-approximation of $p$ if

$$\left(1 - \epsilon\right)p \leq \hat{p} \leq \left(1 + \epsilon\right)p.$$
Monte-Carlo approximation w/ naive estimator

Let $\Phi$ be a propositional formula with $V(\Phi) = \{X_1, \ldots, X_l\}$.

- Pick a value $n$ and for $k \in \{1, 2, \ldots, n\}$, do
  1. Pick a random assignment $\theta_k$ by setting
     
     $X_i = \begin{cases} 
     \text{TRUE} & \text{with probability } \mathbb{P}(X_i) \\
     \text{FALSE} & \text{otherwise}
     \end{cases}$
  2. Evaluate $Z_k = \Phi[\theta_k]$
- Return $\hat{p} = \frac{1}{n} \sum_k Z_k$

How good is this algorithm?

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>$Z_k$</th>
<th>$\hat{p}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.66</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.80</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.83</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.85</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.88</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.89</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.90</td>
</tr>
</tbody>
</table>
Naive estimator: expected value

**Theorem**

The naive estimator $\hat{p}$ is unbiased, i.e., $E[\hat{p}] = P(\Phi)$ so that $\hat{p}$ is correct in expectation.

**Proof.**

$$E[\hat{p}] = E\left[ \frac{1}{n} \sum_{k=1}^{n} Z_k \right] = \frac{1}{n} \sum_{k=1}^{n} E[Z_k]$$

$$= E[Z_1]$$

$$= \sum_{\theta} \Phi[\theta] P(\theta)$$

$$= P(\Phi).$$

But: Is the actual estimate likely to be close to the expected value?
Chernoff bound (1)

Theorem (Two-sided Chernoff bound, simple form)

Let $Z_1, \ldots, Z_n$ be i.i.d. 0/1 random variables with $\mathbb{E}[Z_1] = p$ and set $
abla Z = \frac{1}{n} \sum_k Z_k$. Then,

$$P\left(|\nabla Z - p| \geq \gamma p\right) \leq 2 \exp\left(-\frac{\gamma^2}{2 + \gamma pn}\right)$$

In words:
- Take a coin with (unknown) probability of heads $p$ (thus tail $1 - p$)
- Flip the coin $n$ times: outcomes $Z_1, \ldots, Z_n$
- Compute the fraction $\nabla Z$ of heads
- Estimate $p$ using $\nabla Z$
- Then: Probability that relative error larger than $\gamma$
  1. Decreases exponentially with increasing number of flips $n$
  2. Decreases with increasing error bound $\gamma$
  3. Decreases with increasing probability of heads $p$

Very important result with many applications!
Theorem (Two-sided Chernoff bound, simple form)

Let $Z_1, \ldots, Z_n$ be i.i.d. 0/1 random variables with $\mathbb{E}[Z_1] = p$ and set $\bar{Z} = \frac{1}{n} \sum_k Z_k$. Then,

$$
\mathbb{P}(\left|\bar{Z} - p\right| \geq \gamma p) \leq 2 \exp \left( -\frac{\gamma^2}{2 + \gamma^2} pn \right)
$$

Proof (outline).

We give the first steps of the proof of the one-sided Chernoff bound. First,

$$
\mathbb{P}(Z \geq q) = \mathbb{P}(e^{tZ} \geq e^{tq}).
$$

for any $t > 0$. Use the Markov inequality $\mathbb{P}(\left|X\right| \geq a) \leq \mathbb{E}[\left|X\right|]/a$ to obtain

$$
\mathbb{P}(Z \geq q) \leq \mathbb{E}[e^{tZ}]/e^{tq} = \mathbb{E}[e^{tZ_1} \cdots e^{tZ_n}]/e^{tq} = \mathbb{E}[e^{tZ_1}] \cdots \mathbb{E}[e^{tZ_n}]/e^{tq} = \mathbb{E}[e^{tZ_1}]^n/e^{tq}
$$

Use definition of expected value and find the value of $t$ that minimizes RHS to obtain the precise one-sided Chernoff bound. Relax the RHS to obtain the simple form.
Naive estimator: absolute \((\epsilon, \delta)\)-approximation (1)

**Theorem (sampling theorem)**

To obtain an absolute \(\epsilon\)-approximation with probability at least \(1 - \delta\), it suffices to run

\[
 n \geq \frac{2 + \epsilon}{\epsilon^2} \ln \frac{2}{\delta} = O \left( \frac{1}{\epsilon^2} \ln \frac{1}{\delta} \right)
\]

sampling steps.

**Proof.**

Take \(\gamma = \epsilon / p\) and apply the Chernoff bound to obtain

\[
 \mathbb{P} (|\bar{Z} - p| \geq \epsilon) \leq 2 \exp \left( -\frac{\epsilon^2 / p^2}{2 + \epsilon / p} pn \right) = 2 \exp \left( -\frac{\epsilon^2}{2p + \epsilon} n \right)
\]

\[
 \leq 2 \exp \left( -\frac{\epsilon^2}{2 + \epsilon} n \right)
\]

since \(p \leq 1\). Now solve RHS \(\leq \delta\) for \(n\).
Naive estimator: absolute ($\epsilon, \delta$)-approximation (2)

The number of sampling steps given by the sampling theorem is independent of $\Phi$.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$N$</th>
<th>$1 - \delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>1000</td>
<td>0.9</td>
</tr>
<tr>
<td>0.04</td>
<td>10000</td>
<td>0.95</td>
</tr>
<tr>
<td>0.06</td>
<td>100000</td>
<td>0.99</td>
</tr>
</tbody>
</table>

![Graph showing the relationship between $\epsilon$, $N$, and $1 - \delta$.]
Naive estimator: relative \((\epsilon, \delta)\)-approximation (1)

**Theorem**

To obtain a relative \(\epsilon\)-approximation with probability at least \(1 - \delta\), it suffices to run

\[
n \geq \frac{2 + \epsilon}{p\epsilon^2} \ln \frac{2}{\delta} = O \left( \frac{1}{p\epsilon^2} \ln \frac{1}{\delta} \right)
\]

sampling steps.

**Proof.**

Take \(\gamma = \epsilon\) and apply the Chernoff bound to obtain

\[
\mathbb{P} \left( |\tilde{Z} - p| \geq \epsilon p \right) \leq 2 \exp \left( -\frac{\epsilon^2}{2 + \epsilon} pn \right)
\]

Now solve RHS \(\leq \delta\) for \(n\).
Naive estimator: relative \((\epsilon, \delta)\)-approximation (2)

The number of sampling steps given by the sampling theorem now is dependent on \(\Phi\); we cannot compute the number of required steps in advance! Obtaining small relative error for small \(p\) (i.e., \(\Phi\) is often false) requires a large number of sampling steps.

\[
1 - \delta = 0.9
\]
Why care about relative $\epsilon$-approximation?

1. Absolute error ill-suited to compare estimates of small probabilities
   - $p_1 = 0.001$, $p_2 = 0.01$, $\epsilon = 0.1$
   - Absolute error: $I_1 = [0, 0.101]$, $I_2 = [0, 0.11]$
   - Relative error: $I_1 = [0.0009, 0.0011]$, $I_2 = [0.009, 0.011]$
   → Ranking of tuples more sensitive to absolute error

2. For $p \in [0, 1)$, relative error $\epsilon$ is always tighter than absolute error $\epsilon$
   (esp. when probabilities are small)

Can we get a relative $\epsilon$-approximation in which the minimum number of sampling steps does not depend on $\mathbb{P}(\Phi)$?
The problem with the naive estimator

\[ \Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3 \]

Large probabilities

Small probabilities \((\times 10^{-2})\)

- When \(\mathbb{P}(\Phi)\) is small, \(\Phi\) not satisfied on most samples
  \[ \rightarrow \text{Slow convergence} \]

Idea: Change the sampling strategy so that \(\Phi\) is satisfied on every sample.
Karp-Luby estimator (basic idea)

Let \( \Phi \) be a propositional DNF formula with \( V(\Phi) = \{ X_1, \ldots, X_l \} \), i.e.,

\[
\Phi = C_1 \vee C_2 \vee \cdots \vee C_m.
\]

Easy to find satisfying assignments!

Set \( q_i = \mathbb{P}(C_i) \) and \( Q = \sum_i q_i \). Note that \( p \leq Q \) (union bound).

\[
\mathbb{P}(\Phi) = \mathbb{P}(C_1) + \mathbb{P}(\neg C_1 \land C_2) + \cdots + \mathbb{P}(\neg(C_1 \lor \cdots \lor C_{m-1}) \land C_m)
\]

\[
= \mathbb{P}(\text{TRUE} \mid C_1) \mathbb{P}(C_1) + \mathbb{P}(\neg C_1 \mid C_2) \mathbb{P}(C_2) + \cdots + \mathbb{P}(\neg(C_1 \lor \cdots \lor C_{m-1}) \mid C_m) \mathbb{P}(C_m)
\]

\[
= Q \left[ \mathbb{P}(\text{TRUE} \mid C_1) q_1/Q + \mathbb{P}(\neg C_1 \mid C_2) q_2/Q + \cdots + \mathbb{P}(\neg(C_1 \lor \cdots \lor C_{m-1}) \mid C_m) q_m/Q \right]
\]

Idea of Karp-Luby estimator:

1. \( q_i/Q \) is computed exactly (in linear time)
2. \( \mathbb{P}(\neg(C_1 \lor \cdots \lor C_{i-1}) \mid C_i) \) are estimated
   - Impact of estimate proportional to \( \mathbb{P}(C_i) \)
     - Focus on clauses with highest probability
Karp-Luby estimator

- Pick a value $n$ and for $k \in \{1, 2, \ldots, n\}$, do
  1. Pick a random clause $C_i$ (with probability $q_i/Q$)
  2. Pick a random assignment $\theta_k$
     - For a variable $X \in V(C_i)$
       \[
       X = \begin{cases} 
       \text{TRUE} & \text{if } X \text{ is positive in } C_i \\
       \text{FALSE} & \text{if } X \text{ is negative in } C_i 
       \end{cases}
       \]
       $\rightarrow$ Clause $C_i$ is satisfied (and thus $\Phi$)
     - For the other variables $X \notin V(C_i)$
       \[
       X = \begin{cases} 
       \text{TRUE} & \text{with probability } P(X) \\
       \text{FALSE} & \text{otherwise}
       \end{cases}
       \]
       $\rightarrow$ All other variables take random values
  3. Evaluate
     \[
     Z_k = \begin{cases} 
     1 & \text{if } \neg(\bigvee_{1 \leq j < i} C_j[\theta]) \\
     0 & \text{otherwise}
     \end{cases}
     \]
- Return $\hat{\rho} = \frac{Q}{n} \sum_{k=1}^{n} Z_k$
Example of KL estimator

\[ \Phi = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 \]

- \( m = 3 \), probabilities of \( X_1 \) and \( Y_3 \) reduced to 1/10th
- \( C_1 = X_1 Y_1, \quad q_1 = 0.09 \cdot 0.85 = 0.0765, \quad q_1/Q \approx 0.39 \)
- \( C_2 = X_1 Y_2, \quad q_2 = 0.09 \cdot 0.75 = 0.0675, \quad q_2/Q \approx 0.34 \)
- \( C_3 = X_2 Y_3, \quad q_3 = 0.8 \cdot 0.065 = 0.052, \quad q_3/Q \approx 0.27 \)
- \( Q = 0.196, \quad p \approx 0.134 \)

<table>
<thead>
<tr>
<th>i</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Y_3 )</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
<th>( C_3 )</th>
<th>( Z_k )</th>
<th>( \hat{p} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.196</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.196</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.131</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.147</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.157</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.131</td>
</tr>
</tbody>
</table>
Theorem

The KL estimator $\hat{p}$ is unbiased, i.e., $\mathbb{E} [\hat{p}] = \mathbb{P} (\Phi)$ so that $\hat{p}$ is correct in expectation.

Proof.

\[
\mathbb{E} [\hat{p}] = \mathbb{E} \left[ \frac{Q}{n} \sum_{k=1}^{n} Z_k \right] = Q \mathbb{E} [Z_1] = Q \mathbb{E} [\mathbb{E} [Z_1 | C_i \text{ picked}]]
\]

\[
= Q \sum_{i=1}^{m} \frac{q_i}{Q} \mathbb{E} [Z_1 | C_i \text{ picked}]
\]

\[
= \sum_{i=1}^{m} \mathbb{P} (C_i) \mathbb{P} (\neg \bigvee_{1 \leq j < i} C_j | C_i)
\]

\[
= \mathbb{P} (\Phi).
\]
KL estimator: relative \((\epsilon, \delta)\)-approximation

**Theorem**

To obtain a relative \(\epsilon\)-approximation with probability at least \(1 - \delta\), it suffices to run

\[
 n \geq m \frac{2 + \epsilon}{\epsilon^2} \ln \frac{2}{\delta} = O \left( \frac{m}{\epsilon^2} \ln \frac{1}{\delta} \right)
\]

sampling steps of the KL estimator.

**Proof.**

Use the Chernoff bound with \(\gamma = \epsilon\) and \(\mathbb{E} [ \tilde{Z} ] = Q^{-1} p\).

\[
 \mathbb{P} \left( \left| \tilde{Z} - Q^{-1} p \right| \geq \epsilon Q^{-1} p \right) \leq 2 \exp \left( -\epsilon^2 / (2 + \epsilon) Q^{-1} p n \right) \\
 \mathbb{P} \left( \left| Q^{-1} \hat{p} - Q^{-1} p \right| \geq \epsilon Q^{-1} p \right) = \mathbb{P} \left( \left| \hat{p} - p \right| \geq \epsilon p \right) \leq 2 \exp \left( -\epsilon^2 / (2 + \epsilon) m^{-1} n \right),
\]

since \(mp \geq Q\). Now solve RHS \(\leq \delta\) for \(n\).
KL estimator: discussion

- KL estimator provides relative \((\epsilon, \delta)\)-approximation in polynomial time in size of \(\Phi\) and \(\frac{1}{\epsilon}\)
  \(\rightarrow\) fully polynomial-time randomized approximation scheme (FPTRAS)

- Example: \(\Phi = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3\)

For \(\epsilon, \delta\) fixed and relative error, the naive estimator requires \(O(p^{-1})\) sampling steps and the KL estimator requires \(O(m)\) steps. In general, the naive estimator is preferable when the DNF is very large. The KL estimator preferable if probabilities are small.
Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary
Lessons learned

- Relational calculus is a great tool for query analysis & manipulation
- Query evaluation computes marginal probabilities $\mathbb{P}(t \in q(D))$
- On tuple-independent DBs and $\mathcal{UCQ}$, data complexity either P or #P
- Extensional query evaluation
  - Detects and evaluates the subset of safe queries (P)
  - Leverages query structure to obtain polynomial-time algorithm
  - Uses $R_6$-rules to create an extensional plan that can be executed in an (extended) RDBMS $\rightarrow$ highly scalable
  - Rules are sound and complete for $\mathcal{UCQ}$
- Intensional query evaluation
  - Applies to all queries, but focus is on hard (sub)queries
  - Ignores query structure, leverages data properties
  - Computes probabilities of propositional lineage formulas
  - Rule-based evaluation computes probabilities precisely, but potentially exponential blow-up $\rightarrow$ stop early to obtain probability bounds
  - Sampling techniques apply to all formulas; FPTRAS for $\mathcal{UCQ}$
- Hybrids of extensional and intensional query evaluation promising
Suggested reading

- Serge Abiteboul, Richard Hull, Victor Vianu
  *Foundations of Databases: The Logical Level* (ch. 12)
  Addison Wesley, 1994

- Dan Sucio, Dan Olteanu, Christopher Ré, Christoph Koch
  *Probabilistic Databases* (ch. 3–5)
  Morgan&Claypool, 2011

- Michael Mitzenmacher, Eli Upfal
  *Probability and Computing: Randomized Algorithms and Probabilistic Analysis* (ch. 10)
  Cambridge University Press, 2005