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 efficently evaluate easy queries
  $\rightarrow$ extensional query evaluation
- How to evaluate hard queries
  $\rightarrow$ intensional query evaluation
- How to approximately evaluate queries

Not in this lecture

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

Outline

Relational calculus ($\mathcal{RC}$)

- Similar to nr-datalog$^\neg$, but uses a *single query expression*
- Suitable to reason over query expressions as a whole
- Queries are built from logical connectives

$$q ::= u = v \mid R(\mathbf{x}) \mid \exists x.q_1 \mid q_1 \wedge q_2 \mid q_1 \vee q_2 \mid \neg q_1,$$

  where u, v are either variables of constants
- Extended $\mathcal{RC}$: adds arithmetic expressions
- Free variables in q are called *head variables*

Example

$\mathcal{RA}$ query:

$$\pi_{\text{HotelNo},\text{Name},\text{City}}(\text{Hotel} \bowtie \sigma_{\text{Price}>500 \vee \text{Type}=\text{'suite'}}(\text{Room}))$$

$\mathcal{RC}$ query and its abbreviation:

$$q(h, n, c) \leftarrow \exists r.\exists t.\exists p.\text{Hotel}(h, n, c) \wedge \text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

$$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \wedge \text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

Alternative $\mathcal{RC}$ query:

$$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

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:

$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$

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

$q \leftarrow \exists h.\exists n.\exists c.\text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$

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

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

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

$$q(x) \leftarrow \exists y. \exists z. R(y, z)$$

$$q(x) \leftarrow \exists y. \neg R(x, y)$$

R

1	1
1	2

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$^\neg$ and domain-independent rules for $\mathcal{RC}$).

Relational algebra	nr-datalog$^\neg$	Relational calculus
SPJR	No repeated head predicates, no negation	$\exists, \wedge$ *(conjunctive queries: $\mathcal{CQ}$)*
SPJRU *(positive $\mathcal{RA}$)*	No negation *(nr-datalog)*	$\exists, \wedge, \vee$ *(unions of $\mathcal{CQ}$: $\mathcal{UCQ}$)*
SPJRUD *($\mathcal{RA}$)*	– *(nr-datalog$^\neg$)*	$\exists, \wedge, \vee, \neg$ *($\mathcal{RC}$)*

Outline

The query evaluation problem

- Database systems are expected to *scale* to large datasets and *parallelize* to a large number of processors
  $\rightarrow$ 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

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

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
  $\rightarrow$ Understand what makes query evaluation hard
- Given a query, determine whether it is hard
  $\rightarrow$ Guide query processing
- Given an easy query, solve the QEP
  $\rightarrow$ Be efficient whenever possible
- Given a hard query, solve the QEP (exactly or approximately)
  $\rightarrow$ 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 $\mathcal{RA}$ 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 $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ 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

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

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

This result is treacherous: It talks about probabilistic databases but not about *probabilistic representation systems*!

Lineage trees and the query evaluation problem

Example

$$q(h) \leftarrow \exists n.\exists c.\text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

Room (R)

RoomNo	Type	HotelNo	Price	
R1	Suite	H1	$50	X_1
R2	Single	H1	$600	X_2
R3	Double	H1	$80	X_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge (X_1 \vee X_2)$

Theorem

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

$$\mathbb{P}(\,t \in q(T)\,) = \mathbb{P}(\,\Phi_t\,).$$

How can we compute Φ_t?

A naive approach

Let $\omega(\Phi)$ be the set of assignments over $\mathrm{Var}(T)$ that make Φ 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 Φ, count the number of satisfying assignments $\#\Phi = |\omega(\Phi)|$.

Definition (Probability computation problem)

Given a propositional formula Φ 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 $\mathbb{P}(\Phi)$
- We can use the algorithm to compute $\#\Phi$
- Define $\mathbb{P}(X) = \frac{1}{2}$ for every variable X
- $\mathbb{P}(\theta) = 1/2^n$ for every assignment ($n =$ number of variables)
- $\#\Phi = \mathbb{P}(\Phi) \cdot 2^n$

Example

- $\Phi = (X_1 \vee X_2) \wedge X_4$; $n = 3$
- $\#\Phi = 3$
- $\mathbb{P}(\Phi) = \frac{3}{8} = \frac{\#\Phi}{2^n}$

X_1	X_2	X_4	Φ_θ	$\mathbb{P}(\theta)$
0	0	0	FALSE	1/8
0	0	1	FALSE	1/8
0	1	0	FALSE	1/8
0	1	1	TRUE	1/8
1	0	0	FALSE	1/8
1	0	1	TRUE	1/8
1	1	0	FALSE	1/8
1	1	1	TRUE	1/8

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

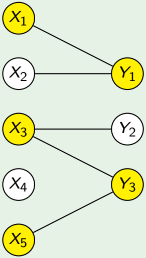
Directly implies that probability computation is hard for #P!

A graph problem

Definition (Bipartite vertex cover)

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

Example



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

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 $\mathcal{CQ}$ query H_0 given by

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

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

Proof.

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

R

X	
X_1	1/2
X_2	1/2
$\vdots$	

S

X	Y	
X_{e_1}	Y_{e_1}	1
X_{e_2}	Y_{e_2}	1
$\vdots$	$\vdots$	$\vdots$

T

Y	
Y_1	1/2
Y_2	1/2
$\vdots$	

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

More hard queries

Theorem

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

$$
\begin{aligned}
H_0 &\leftarrow R(x) \wedge S(x,y) \wedge T(y) \\
H_1 &\leftarrow [R(x_0) \wedge S(x_0,y_0)] \vee [S(x_1,y_1) \wedge T(y_1)] \\
H_2 &\leftarrow [R(x_0) \wedge S_1(x_0,y_0)] \vee [S_1(x_1,y_1) \wedge S_2(x_1,y_1)] \\
&\qquad \vee [S_2(x_2,y_2) \wedge T(y_2)] \\
&\vdots
\end{aligned}
$$

Queries can be tractable even if they have intractable subqueries!

- $q(x,y) \leftarrow R(x) \wedge S(x,y) \wedge T(y)$ is tractable
- $q \leftarrow H_0 \vee T(y)$ 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

Problem statement

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

Example

Birds

Species	$\mathbb{P}$	
Finch	0.80	X_1
Toucan	0.71	X_2
Nightingale	0.65	X_3
Humming bird	0.55	X_4

- $\mathbb{P}(\text{Finch}) = \mathbb{P}(X_1) = 0.8$
- Is there a finch? $Q \leftarrow \text{Birds(Finch)}$
  - $\Phi_Q = X_1$
  - $\mathbb{P}(Q) = 0.8$
- Is there some bird? $Q \leftarrow \text{Birds}(s)$?
  - $\Phi_Q = X_1 \vee X_2 \vee X_3 \vee X_4$
  - $\mathbb{P}(Q) \approx 99.1\%$

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 $\mathcal{UCQ}$ queries
  - Completeness: Rules succeed iff query is tractable
  - Dichotomy: Query is #P-hard if rules don't succeed

Outline

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 $L_1[\mathbf{a}_1/\mathbf{x}_1] = L_2[\mathbf{a}_2/\mathbf{x}_2]$, where $\mathbf{x}_1$ are the variables in L_1 and $\mathbf{x}_2$ 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:

- $R(a)$, $R(b)$
- $R(a, y)$, $R(b, y)$
- $R(x)$, $S(x)$
- $R(a, x) \vee S(x)$, $R(b, x) \wedge T(x)$

Not syntactically independent:

- $R(a)$, $R(x)$
- $R(x)$, $R(y)$
- $R(x)$, $S(x) \wedge \neg R(x)$

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 $\mathrm{Var}(\Phi_{Q_1}), \ldots, \mathrm{Var}(\Phi_{Q_k})$ 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. $\square$

Example

Syntactically independent:

- $R(a)$, $R(b)$
- $R(a, y)$, $R(b, y)$
- $R(x)$, $S(x)$
- $R(a, x) \vee S(x)$, $R(b, x) \wedge T(x)$

Not syntactically independent:

- $R(a)$, $R(x)$
- $R(x)$, $R(y)$
- $R(x)$, $S(x) \wedge \neg R(x)$

Probabilistic independence and syntactic independence

Proposition

Probabilistic independence does not necessarily imply syntactic independence.

Example

- Consider

$$Q_1 \leftarrow R(x, y) \wedge R(x, x)$$
$$Q_2 \leftarrow R(a, b)$$

- If Φ_{Q_1} does not contain $X_{R(a,b)}$, Q_1 and Q_2 are independent
- Otherwise, Φ_{Q_1} contains $X_{R(a,b)}$ and therefore $X_{R(a,b)} \wedge X_{R(a,a)}$
- Then, Φ_{Q_1} also contains $X_{R(a,a)} \wedge X_{R(a,a)} = X_{R(a,a)}$
- Thus, by the absorption law,

$$(X_{R(a,b)} \wedge X_{R(a,a)}) \vee X_{R(a,a)} = X_{R(a,a)}$$

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

Outline

Base case: Atoms

Definition

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

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did Mary see a toucan?
- $Q = \text{Sightings}(\text{Mary}, \text{Toucan})$
- $\mathbb{P}(Q) = 0.3$

Rule 1: Independent-join

Definition

If Q_1 and Q_2 are syntactically independent, then

$$\mathbb{P}(\, Q_1 \wedge Q_2 \,) = \mathbb{P}(\, Q_1 \,) \cdot \mathbb{P}(\, Q_2 \,)\,. \qquad (\textit{independent-join})$$

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did both Mary and Susan see a toucan?
- $Q = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \wedge \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan})$
- $Q_1 = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan})$ $\quad \mathbb{P}(\, Q_1 \,) = 0.3$
- $Q_2 = \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan})$ $\quad \mathbb{P}(\, Q_2 \,) = 0.5$
- $\mathbb{P}(\, Q \,) = \mathbb{P}(\, Q_1 \,) \cdot \mathbb{P}(\, Q_2 \,) = 0.15$

Rule 2: Independent-union

Definition

If Q_1 and Q_2 are syntactically independent, then

$$\mathbb{P}(\,Q_1 \vee Q_2\,) = 1 - (1 - \mathbb{P}(\,Q_1\,))(1 - \mathbb{P}(\,Q_2\,)). \quad (\textit{independent-union})$$

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did Mary or Susan see a toucan?
- $Q = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \vee \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan})$
- $Q_1 = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \qquad \mathbb{P}(\,Q_1\,) = 0.3$
- $Q_2 = \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan}) \qquad \mathbb{P}(\,Q_2\,) = 0.5$
- $\mathbb{P}(\,Q\,) =$
  $1 - (1 - \mathbb{P}(\,Q_1\,))(1 - \mathbb{P}(\,Q_2\,)) = 0.65$

Root variables and separator variables

Definition

Consider atom L and query Q. Denote by $\mathrm{Pos}(L, x)$ 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., $\mathrm{Pos}(L, x) \neq \emptyset$ 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., $\mathrm{Pos}(L_1, x) \cap \mathrm{Pos}(L_2, x) \neq \emptyset$.

Example

$Q_1 \leftarrow \exists x.\mathrm{Likes}(a, x) \wedge \mathrm{Likes}(x, a)$

- $\mathrm{Pos}(\mathrm{Likes}(a, x), x) = \{ 2 \}$
- $\mathrm{Pos}(\mathrm{Likes}(x, a), x) = \{ 1 \}$
- x is root variable
- x is no separator variable

$Q_2 \leftarrow \exists x.\mathrm{Likes}(a, x) \wedge \mathrm{Likes}(x, x)$

- x is root variable
- x is a separator variable

$Q_3 \leftarrow \exists x.\mathrm{Likes}(a, x) \wedge \mathrm{Popular}(\mathrm{a})$

- 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

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

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

Rule 3: Independent-project

Definition

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

$$\mathbb{P}(\,Q\,) = 1 - \prod_{a \in \mathrm{ADom}} \left(1 - \mathbb{P}(\,Q'[a/x]\,)\right), \qquad (\textit{independent-project})$$

where ADom is the active domain of the database.

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Has anybody seen a toucan?
- $Q = \exists x.S(x, \mathrm{Toucan})$
- $Q' = S(x, \mathrm{Toucan})$
- $$\begin{aligned}\mathbb{P}(\,Q\,) &= 1 - \prod_{x \in \{\mathrm{M,S,F,\ldots}\}} (1 - \mathbb{P}(\,S(x, T)\,)) \\ &= 1 - (1 - 0.3)(1 - 0.5)1 \cdots 1 \\ &= 0.65\end{aligned}$$

Rule 4: Negation

Definition

If the query is $\neg Q$, then

$$\mathbb{P}(\neg Q) = 1 - \mathbb{P}(Q) \qquad \text{(\textit{negation})}$$

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did nobody see a toucan?
- $Q = \neg[\exists x.S(x, \text{Toucan})]$
- $\mathbb{P}(Q) = 1 - \mathbb{P}(\exists x.S(x, \text{Toucan})) = 0.35$

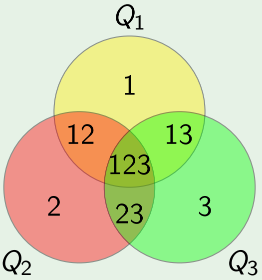
Rule 5: Inclusion-exclusion

Definition

Suppose $Q = Q_1 \wedge Q_2 \wedge \ldots Q_k$. Then,

$$\mathbb{P}(\,Q\,) = -\sum_{\emptyset \neq S \subseteq \{1,\ldots,k\}} (-1)^{|S|}\,\mathbb{P}(\bigvee_{i \in S} Q_i\,) \qquad (\textit{inclusion-exclusion})$$

Example



1	2	3	12	13	23	123	$\mathbb{P}(\,Q_1 \wedge Q_2 \wedge Q_3\,) =$
1	0	0	1	1	0	1	$+\mathbb{P}(\,Q_1\,)$
1	1	0	2	1	1	2	$+\mathbb{P}(\,Q_2\,)$
1	1	1	2	2	2	3	$+\mathbb{P}(\,Q_3\,)$
0	0	1	1	1	1	2	$-\mathbb{P}(\,Q_1 \vee Q_2\,)$
-1	0	0	0	0	0	1	$-\mathbb{P}(\,Q_1 \vee Q_3\,)$
-1	-1	-1	-1	-1	-1	0	$-\mathbb{P}(\,Q_2 \vee Q_3\,)$
0	0	0	0	0	0	1	$+\mathbb{P}(\,Q_1 \vee Q_2 \vee Q_3\,)$

Inclusion-exclusion for independent-project

Goal of inclusion-exclusion is to apply the rewrite

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

Example

Sightings

Name	Species	$\mathbb{P}$
Mary	Finch	0.8
Mary	Toucan	0.3
Susan	Finch	0.2
Susan	Toucan	0.5
Susan	Nightingale	0.6

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

$$\begin{aligned}
&\mathbb{P}\left((\exists x.\mathrm{S}(\mathrm{M},x)) \wedge (\exists y.\mathrm{S}(y,\mathrm{F})) \right) && (ie)\\
&= \mathbb{P}\left(\exists x.\mathrm{S}(\mathrm{M},x) \right) + \mathbb{P}\left(\exists y.\mathrm{S}(y,\mathrm{F}) \right) - \mathbb{P}\left((\exists x.\mathrm{S}(\mathrm{M},x)) \vee (\exists y.\mathrm{S}(y,\mathrm{F})) \right) && (ip/ip/rewrite)\\
&= 0.86 + 0.84 - \mathbb{P}\left(\exists x.\mathrm{S}(\mathrm{M},x) \vee \mathrm{S}(x,\mathrm{F}) \right)\\
&= 1.7 - \mathbb{P}\left(\exists x.\mathrm{S}(\mathrm{M},x) \vee \mathrm{S}(x,\mathrm{F}) \right)
\end{aligned}$$

Now we are stuck $\rightarrow$ 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 occurence of form $R(\ldots)$ by $R_1(\ldots) \vee R_2(\ldots)$, where

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

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

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

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?

$$
\begin{aligned}
&\mathbb{P}(\,(\exists x.\mathrm{S}(\mathrm{M},x)) \wedge (\exists y.\mathrm{S}(y,\mathrm{F}))\,) \\
&= 1.7 - \mathbb{P}(\,\exists x.\mathrm{S}(\mathrm{M},x) \vee \mathrm{S}(x,\mathrm{F})\,) && (\textit{rank}\ (\mathrm{Name{=}Mary})) \\
&= 1.7 - \mathbb{P}(\,\exists x.\mathrm{S}_{\mathrm{M}}(x) \vee S_{\neg\mathrm{M}}(\mathrm{M},x) \vee [\mathrm{S}_{\mathrm{M}}(\mathrm{F}) \wedge x = \mathrm{M}] \vee \mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,) && (\textit{simplify}) \\
&= 1.7 - \mathbb{P}(\,\exists x.\mathrm{S}_{\mathrm{M}}(x) \vee \mathrm{S}_{\mathrm{M}}(\mathrm{F}) \vee \mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,) && (\textit{rank}\ (\mathrm{Species{=}Finch})) \\
&= 1.7 - \mathbb{P}(\,\exists x.[\mathrm{S}_{\mathrm{MF}}() \wedge x = \mathrm{F}] \vee \mathrm{S}_{\mathrm{M}\neg\mathrm{F}}(x) \vee \mathrm{S}_{\mathrm{MF}}() \vee \mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,) && (\textit{push}\ \exists x) \\
&= 1.7 - \mathbb{P}(\,\mathrm{S}_{\mathrm{MF}}() \vee \exists x.\mathrm{S}_{\mathrm{M}\neg\mathrm{F}}(x) \vee \mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,) && (\textit{iu}) \\
&= 1.7 - 1 + (1 - \mathbb{P}(\,\mathrm{S}_{\mathrm{MF}}()\,))(1 - \mathbb{P}(\,\exists x.\mathrm{S}_{\mathrm{M}\neg\mathrm{F}}(x) \vee \mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,) && (\textit{base}/\textit{ip}) \\
&= 0.7 + (1 - 0.8)\left[\textstyle\prod_{x\in\{\mathrm{M,S,F,T,N}\}}(1 - \mathbb{P}(\,\mathrm{S}_{\mathrm{M}\neg\mathrm{F}}(x) \vee \mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,)\right] && (\textit{iu}) \\
&= 0.7 + 0.2\left[\textstyle\prod_{x\in\{\mathrm{M,S,F,T,N}\}}(1 - \mathbb{P}(\,\mathrm{S}_{\mathrm{M}\neg\mathrm{F}}(x)\,))(1 - \mathbb{P}(\,\mathrm{S}_{\neg\mathrm{M}}(x,\mathrm{F})\,))\right] && (\textit{product}) \\
&= 0.7 + 0.2[11 \cdot 1(1 - 0.2) \cdot 11 \cdot (1 - 0.3)1 \cdot 11] \\
&= 0.812
\end{aligned}
$$

S

N	S	$\mathbb{P}$
M	F	0.8
M	T	0.3
S	F	0.2
S	T	0.5
S	N	0.6

S_M

S	$\mathbb{P}$
F	0.8
T	0.3

$S_{\neg M}$

N	S	$\mathbb{P}$
S	F	0.2
S	T	0.5
S	N	0.6

S_{MF}

$\mathbb{P}$
0.8

$S_{M\neg F}$

S	$\mathbb{P}$
T	0.3

Attribute-attribute ranking (example)

The goal of attribute ranking is to establish syntactic independence and new separators by exploiting disjointness.

Example

L

P_1	P_2	$\mathbb{P}$
A	B	0.8
B	A	0.7
C	A	0.2
C	C	0.9

$L_<$

P_1	P_2	$\mathbb{P}$
A	B	0.8

$L_=$

P_{12}	$\mathbb{P}$
C	0.9

$L_>$

P_1	P_2	$\mathbb{P}$
B	A	0.7
C	A	0.2

Are there two people who like each other?

$$
\begin{aligned}
&\mathbb{P}(\exists x.\exists y.\mathrm{Likes}(x,y) \wedge \mathrm{Likes}(y,x)) && (\mathit{rank})\\
&= \mathbb{P}(\exists x.\exists y.\\
&\qquad (\mathrm{Likes}_<(x,y) \vee (\mathrm{Likes}_=(x) \wedge x = y) \vee \mathrm{Likes}_>(x,y)) \wedge\\
&\qquad (\mathrm{Likes}_<(y,x) \vee (\mathrm{Likes}_=(x) \wedge x = y) \vee \mathrm{Likes}_>(y,x))) && (\mathit{expand}, \mathit{disjoint})\\
&= \mathbb{P}(\exists x.\exists y.\mathrm{L}_<(x,y)\mathrm{L}_>(y,x) \vee (\mathrm{L}_=(x) \wedge x = y) \vee \mathrm{L}_>(x,y)\mathrm{L}_<(y,x)) && (\mathit{push}\ \exists)\\
&= \mathbb{P}((\exists x.\exists y.\mathrm{L}_<(x,y)\mathrm{L}_>(y,x))\\
&\qquad \vee (\exists x.\mathrm{L}_=(x))\\
&\qquad \vee (\exists x.\exists y.\mathrm{L}_>(x,y)\mathrm{L}_<(y,x))) && (1st \equiv 3rd)\\
&= \mathbb{P}((\exists x.\exists y.\mathrm{L}_<(x,y)\mathrm{L}_>(y,x))\\
&\qquad \vee (\exists x.\mathrm{L}_=(x)))
\end{aligned}
$$

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

$$Q \leftarrow (Q_1 \vee Q_3) \wedge (Q_1 \vee Q_4) \wedge (Q_2 \vee Q_4)$$

Apply inclusion exclusion to get

$$\begin{aligned}
\mathbb{P}(Q) = {} & \mathbb{P}(Q_1 \vee Q_3) + \mathbb{P}(Q_1 \vee Q_4) + \mathbb{P}(Q_2 \vee Q_4) \\
& - \mathbb{P}(Q_1 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_4) \\
& + \mathbb{P}(Q_1 \vee Q_2 \vee Q_3 \vee Q_4) \\
= {} & \mathbb{P}(Q_1 \vee Q_3) + \mathbb{P}(Q_1 \vee Q_4) + \mathbb{P}(Q_2 \vee Q_4) \\
& - \mathbb{P}(Q_1 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_4)
\end{aligned}$$

One can construct cases in which $Q_1 \vee Q_2 \vee Q_3 \vee Q_4$ 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 $Q_1 \wedge \ldots \wedge Q_k$:

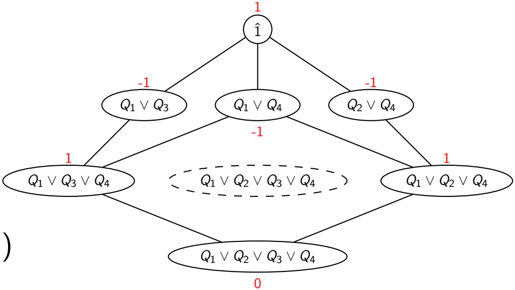
1. Put the formulas $Q_S = \bigvee_{i \in S} Q_i$, $\emptyset \neq S \subseteq \{1, \ldots, j\}$, in a lattice (plus special element $\hat{1}$)
2. Eliminate duplicates (equivalent formulas)
3. Use the partial order $Q_{S_1} \geq Q_{S_2}$ iff $Q_{S_1} \Leftarrow Q_{S_2}$
4. Label each node by its Möbius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

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

5. Use the inversion formula

$$\mathbb{P}(Q_1 \wedge \ldots \wedge Q_k) = -\sum_{u < \hat{1} : \mu(u) \neq 0} \mu(u)\,\mathbb{P}(Q_u)$$

$Q \leftarrow (Q_1 \vee Q_3) \wedge (Q_1 \vee Q_4) \wedge (Q_2 \vee Q_4)$



$$\mathbb{P}(Q) = \mathbb{P}(Q_1 \vee Q_3) + \mathbb{P}(Q_1 \vee Q_4) + \mathbb{P}(Q_2 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_4)$$

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 $\mathbf{R}_6$-*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 $\rightarrow$ do not depend on database
  - Independent-project increases number of queries by a factor of $|\mathrm{ADom}|$ $\rightarrow$ applied at most k times, where k is the maximum arity of a relation

How the rules fail

Example

Consider the hard query

$$H_0 \leftarrow \exists x.\exists y.R(x) \wedge S(x,y) \wedge T(y)$$

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

$$\begin{aligned} H_0 &\leftarrow H_{01} \vee H_{02} \vee H_{03} \\ H_{01} &\leftarrow \exists x.\exists y.R(x) \wedge S_<(x,y) \wedge T(y) \\ H_{02} &\leftarrow \exists x.R(x) \wedge S_=(x) \wedge T(x) \\ H_{03} &\leftarrow \exists x.\exists y.R(x) \wedge S_>(y,x) \wedge T(y) \end{aligned}$$

- Now we are stuck at H_{01} and H_{03}

Dichotomy theorem for $\mathcal{UCQ}$

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

Theorem (Dalvi and Suciu, 2010)

For any $\mathcal{UCQ}$ query Q, one of the following holds:

- *Q is $\mathbf{R}_6$-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 $\mathcal{RC}$, completeness/dichotomy unknown

We can handle all safe $\mathcal{UCQ}$ queries!

Outline

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 $\mathcal{UCQ}$, 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 $\mathcal{UCQ}$ queries, we can always produce a safe plan
  - Plan construction based on $\mathbf{R}_6$ rules
  - Can be written in SQL (though not "best" approach)
  - **Enables scalable query processing on probabilistic databases**

Basic operators

Definition

Annotate each tuple by its probability. The operators

- *Independent join* ($\bowtie^{\mathrm{i}}$)
- *Independent project* (π^{i})
- *Independent union* ($\cup^{\mathrm{i}}$)
- 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.)

$\boxed{([0,1], 0, 1, \oplus, \cdot) \text{ is not a semiring} \rightarrow \text{unsafe plans!}}$

Example plans

Incriminates

Witness	Suspect	
Mary	Paul	p_1
Mary	John	p_2
Susan	John	p_3

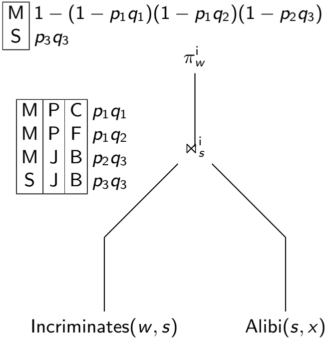
Alibi

Suspect	Claim	
Paul	Cinema	q_1
Paul	Friend	q_2
John	Bar	q_3

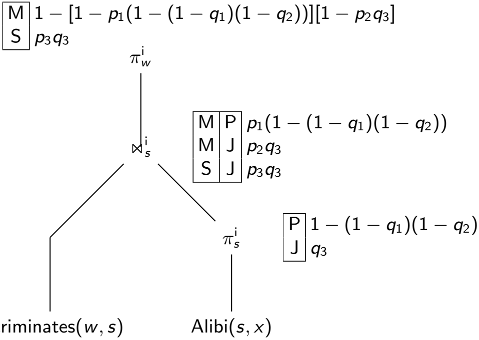
Who incriminates someone
who has an alibi?

$$Q_1(w) \leftarrow \exists s.\exists x.\text{Incriminates}(w, s) \wedge \text{Alibi}(s, x)$$

$$Q_2(w) \leftarrow \exists s.\text{Incriminates}(w, s) \wedge \exists x.\text{Alibi}(s, x)$$



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:

$$\left(\sum_{U}^{\mu_1,\ldots,\mu_k} (R_1, \ldots, R_k) \right) [] = R_1 \bowtie \cdots \bowtie R_k$$

$$\left(\sum_{U}^{\mu_1,\ldots,\mu_k} (R_1, \ldots, R_k) \right) (t) = \mu_1(R_1(t)) + \cdots \mu_k(R_k(t))$$

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:

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

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 $\mu_2 = 1$
- $Q_3(x, y, z) = V_1(x, y) \vee V_2(x, z)$ with $\mu_3 = -1$

We obtain:

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

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

Complement

How to deal with negation?

Definition

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

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

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

Example

- Query: $Q \leftarrow R(x) \wedge \neg S(x)$
- Result: $R -^{\mathrm{i}} S = \{\, (t, \mathbb{P}(\, t \in R\,)\,(1 - \mathbb{P}(\, t \in S\,))) : t \in R\,\}$

Computation of safe plans (1)

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 $\mathcal{UCQ}$ 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

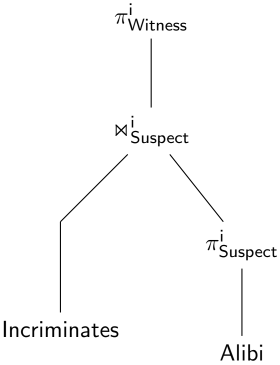
Computation of safe plans (2)

if $Q = Q_1 \wedge Q_2$ and Q_1, Q_2 are syntactically independent **then**
    return $\text{plan}(Q_1) \bowtie^{\text{i}} \text{plan}(Q_2)$
end if
if $Q = Q_1 \vee Q_2$ and Q_1, Q_2 are syntactically independent **then**
    return $\text{plan}(Q_1) \cup^{\text{i}} \text{plan}(Q_2)$
end if
if $Q(\mathbf{x}) = \exists z.Q_1(\mathbf{x}, z)$ and z is a separator variable **then**
    return $\pi^{\text{i}}_{\mathbf{x}}(\text{plan}(Q_1(\mathbf{x}, z)))$
end if
if $Q = Q_1 \wedge \ldots \wedge Q_k,\ k \geq 2$ **then**
    Construct CNF lattice $Q'_1, \ldots, Q'_m$
    Compute Möbius coefficients $\mu_1, \ldots, \mu_m$
    return $\sum^{\mu_1, \ldots, \mu_m}(\text{plan}(Q'_1), \ldots, \text{plan}(Q'_m))$
end if
if $Q = \neg Q_1$ **then**
    return $C(\text{plan } Q_1)$
end if
if $Q(\mathbf{x}) = R(\mathbf{x})$ where R is a base table (possibly ranked) **then**
    return $R(\mathbf{x})$
end if
otherwise FAIL

Computation of safe plans (example)

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

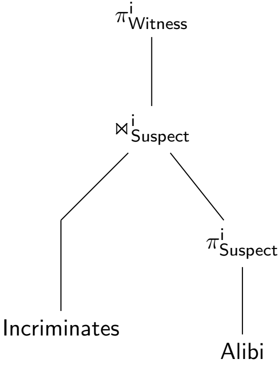
1. Apply independent-project to Q on s
   - $Q_1(w, s) \leftarrow \exists x.\text{Incriminates}(w, s) \wedge \text{Alibi}(s, x)$
2. x is not a root variable in $Q_1 \rightarrow$ push $\exists x$:
   $Q_2(w, s) \leftarrow \text{Incriminates}(w, s) \wedge \exists x.\text{Alibi}(s, x)$
3. Apply independent-join 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)

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

- $Q_4 \leftarrow \pi^{\mathrm{i}}_{\mathrm{Suspect}}(\mathrm{Alibi})$
- $Q_2 \leftarrow \mathrm{Incriminates} \bowtie^{\mathrm{i}}_{\mathrm{Suspect}} Q_4$
- $Q \leftarrow \pi^{\mathrm{i}}_{\mathrm{Witness}}(Q_2)$




```
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
  $\rightarrow$ 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
  $$\pi^{\mathrm{i}}_{\emptyset}\left[R(x) \bowtie^{\mathrm{i}}_{x} (S(x,y) \bowtie_{y} T(y))\right]$$

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

Outline

Overview

Given a query $Q(\mathbf{x})$, a TI database $\mathcal{D}$; for each output tuple t

1. Compute the lineage $\Phi = \Phi^{\mathcal{D}}_{Q(t)}$
   - $|\Phi| = O(|\text{ADom}|^m)$, where m is the number of variables in Φ
   - 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)

Example

$$q(h) \leftarrow \exists n.\exists c.\mathrm{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\mathrm{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

Room (R)

RoomNo	Type	HotelNo	Price	
R1	Suite	H1	\$50	X_1
R2	Single	H1	\$600	X_2
R3	Double	H1	\$80	X_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge (X_1 \vee X_2)$

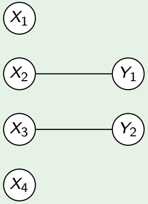
- $\Phi = X_4 \wedge (X_1 \vee X_2)$
- $\mathbb{P}(\Phi) = \mathbb{P}(X_4)[1 - (1 - \mathbb{P}(X_1))(1 - \mathbb{P}(X_2))]$
- E.g., $\mathbb{P}(X_i) = \frac{1}{2}$ for all $i \rightarrow \mathbb{P}(\Phi) = 0.375$

ExpensiveHotels

HotelNo	$\mathbb{P}$
H1	0.375

Example (intractable query)

Example



R

X_1	0.5
X_2	0.5
X_3	0.5
X_4	0.5

S

X_2	Y_1	1
X_3	Y_2	1

T

Y_1	0.5
Y_2	0.5

- $H_0 \leftarrow \exists x. \exists y. R(x), S(x,y), T(y)$
- $\Phi = X_2 Y_1 \vee X_3 Y_2$
- $\mathbb{P}(\Phi) = 1 - (1 - \mathbb{P}(X_2)\mathbb{P}(Y_1))(1 - \mathbb{P}(X_3)\mathbb{P}(Y_2)) = 0.4375$
- Model counting: $\#\Phi = 2^6 \, \mathbb{P}(\Phi) = 28$
- Bipartite vertex cover: $\#\Psi = 2^6 - \#\Phi = 36 = 2 \cdot 3 \cdot 3 \cdot 2$

Outline

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 $\rightarrow$ 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 Φ, denote by $V(\Phi)$ the set of variables that occur in Φ. Denote by $\mathrm{Var}(\Phi)$ the set of variables on which Φ depends; $\mathrm{Var}(\Phi)$ is called the *support* of Φ. $X \in \mathrm{Var}(\Phi)$ iff there exists an assignment θ 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

$\Phi = X \vee (Y \wedge Z)$

- $V(\Phi) = \{ X, Y, Z \}$
- $\mathrm{Var}(\Phi) = \{ X, Y, Z \}$

$\Phi = Y \vee (X \wedge Y) \equiv Y$

- $V(\Phi) = \{ X, Y \}$
- $\mathrm{Var}(\Phi) = \{ Y \}$

Syntactic independence

Definition

Φ_1 and Φ_2 are *syntactically independent* if they have disjoint support, i.e., $\mathrm{Var}(\Phi_1) \cap \mathrm{Var}(\Phi_2) = \emptyset$.

Example

$\Phi_1 = X \qquad \Phi_2 = Y \qquad \Phi_3 = \neg X \neg Y \vee XY$

- Φ_1 and Φ_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

$\Phi_1 = X \qquad \Phi_2 = Y \qquad \Phi_3 = \neg X \neg Y \vee XY$

- Φ_1 and Φ_2 are probabilistically independent
- Φ_1, Φ_2, Φ_3 are not pairwise syntactically independent

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

- Φ_1, Φ_2, Φ_3 are pairwise independent
- Φ_1, Φ_2, Φ_3 are not independent!

Outline

Rules 1 and 2: independent-and, independent-or

Definition

Let Φ_1 and Φ_2 be two syntactically independent propositional formulas:

$$
\begin{aligned}
\mathbb{P}(\,\Phi_1 \land \Phi_2\,) &= \mathbb{P}(\,\Phi_1\,) \cdot \mathbb{P}(\,\Phi_2\,) && (\textit{independent-and}) \\
\mathbb{P}(\,\Phi_1 \lor \Phi_2\,) &= 1 - (1 - \mathbb{P}(\,\Phi_1\,))(1 - \mathbb{P}(\,\Phi_2\,)) && (\textit{independent-or})
\end{aligned}
$$

Independent-and, independent-or (example)

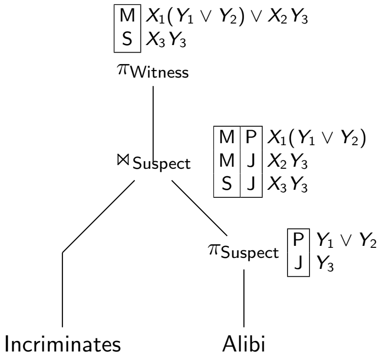
Incriminates

Witness	Suspect	
Mary	Paul	X_1 (p_1)
Mary	John	X_2 (p_2)
Susan	John	X_3 (p_3)

Alibi

Suspect	Claim	
Paul	Cinema	Y_1 (q_1)
Paul	Friend	Y_2 (q_2)
John	Bar	Y_3 (q_3)

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



- $\Phi_S = X_3 Y_3$
  1. Independent-and: $\mathbb{P}(\Phi_S) = p_3 q_3$
- $\Phi_M = X_1(Y_1 \vee Y_2) \vee X_2 Y_3$
  1. Independent-or:
     $\mathbb{P}(\Phi_M) = 1 - (1 - \mathbb{P}(X_1(Y_1 \vee Y_2)))(1 - \mathbb{P}(X_2 Y_3))$
  2. Independent-and: $\mathbb{P}(X_2 Y_3) = p_2 q_3$
  3. Independent-and:
     $\mathbb{P}(X_1(Y_1 \vee Y_2)) = p_1 \mathbb{P}(Y_1 \vee Y_2)$
  4. Independent-or:
     $\mathbb{P}(Y_1 \vee Y_2) = 1 - (1 - q_1)(1 - q_2)$
  5. $\mathbb{P}(\Phi_M) = 1 - [1 - p_1(1 - (1 - q_1)(1 - q_2))](1 - p_2 q_3)$

Rule 3: Disjoint-or

Definition

Two propositional formulas Φ_1 and Φ_2 are *disjoint* if $\Phi_1 \wedge \Phi_2$ is not satisfiable.

Definition

If Φ_1 and Φ_2 are disjoint:

$$\mathbb{P}(\Phi_1 \vee \Phi_2) = \mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2) \qquad \text{(\textit{disjoint-or})}$$

Example

- $\mathbb{P}(X) = 0.2; \quad \mathbb{P}(Y) = 0.7$
- $\Phi_1 = XY; \quad \mathbb{P}(XY) = \mathbb{P}(X)\mathbb{P}(Y) = 0.14$
- $\Phi_2 = \neg X; \quad \mathbb{P}(\neg X) = 0.8$
- $\mathbb{P}(\Phi_1 \vee \Phi_2) = \mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2) = 0.94$

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

Rule 4: Negation

Definition

$$\mathbb{P}(\neg\Phi) = 1 - \mathbb{P}(\Phi) \qquad \textit{(negation)}$$

Example

- $\mathbb{P}(X) = 0.2; \quad \mathbb{P}(Y) = 0.7$
- $\mathbb{P}(XY) = \mathbb{P}(X)\mathbb{P}(Y) = 0.14$
- $\mathbb{P}(\neg(XY)) = 1 - 0.14 = 0.86$

Shannon expansion

Definition

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

$$\Phi \equiv (\Phi[X \mapsto a_1] \wedge (X = a_1)) \vee \ldots \vee (\Phi[X \mapsto a_m] \wedge (X = a_m))$$

Example

- $\Phi = XY \vee XZ \vee YZ$
- $\Phi \equiv (\Phi[X \mapsto \texttt{TRUE}] \wedge X) \vee (\Phi[X \mapsto \texttt{FALSE}] \wedge \neg X)$
  $= (Y \vee Z)X \vee YZ\neg X$

In the Shannon expansion rule, every $\wedge$ is an independent-and; every $\vee$ is a disjoint-or.

Rule 5: Shannon expansion

Definition

Let Φ be a propositional formula and X be a variable:

$$\mathbb{P}(\Phi) = \sum_{a \in \mathrm{dom}(X)} \mathbb{P}(\Phi[X \mapsto a])\,\mathbb{P}(X = a) \qquad (\textit{Shannon expansion})$$

Example

- $\Phi = XY \vee XZ \vee YZ$
- $\mathbb{P}(\Phi) = \mathbb{P}(Y \vee Z)\,\mathbb{P}(X) + \mathbb{P}(YZ)\,\mathbb{P}(\neg X)$

- Can always be applied
- Effectively eliminates X from the formula
- But may lead to exponential blowup!

Shannon expansion (example)

Incriminates

Witness	Suspect	
Mary	Paul	X_1 (p_1)
Mary	John	X_2 (p_2)
Susan	John	X_3 (p_3)

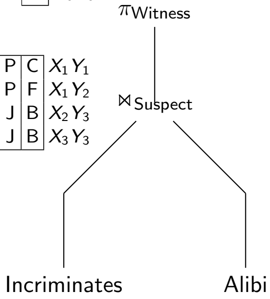
Alibi

Suspect	Claim	
Paul	Cinema	Y_1 (q_1)
Paul	Friend	Y_2 (q_2)
John	Bar	Y_3 (q_3)

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

M	$X_1Y_1 \lor X_1Y_2 \lor X_2Y_3$
S	X_3Y_3

M	P	C	X_1Y_1
M	P	F	X_1Y_2
M	J	B	X_2Y_3
S	J	B	X_3Y_3



$$\Phi_M = X_1Y_1 \lor X_1Y_2 \lor X_2Y_3$$

1. Independent-or:
   $\mathbb{P}(\Phi_M) = 1 - (1 - \mathbb{P}(X_1Y_1 \lor X_1Y_2))(1 - \mathbb{P}(X_2Y_3))$
2. Independent-and: $\mathbb{P}(X_2Y_3) = p_2q_3$
3. Shannon expansion: $\mathbb{P}((X_1Y_1 \lor X_1Y_2)) =$
   $\mathbb{P}(Y_1 \lor Y_2)\mathbb{P}(X_1) + \mathbb{P}(\texttt{FALSE})\mathbb{P}(\neg X_1)$
4. Independent-or:
   $\mathbb{P}(Y_1 \lor Y_2) = 1 - (1 - q_1)(1 - q_2)$
5. $\mathbb{P}(\Phi_M) = 1 - [1 - p_1(1 - (1 - q_1)(1 - q_2))](1 - p_2q_3)$

The intensional rules work on all plans!

A non-deterministic algorithm

if $\Phi = \Phi_1 \wedge \Phi_2$ and Φ_1, Φ_2 are syntactically independent **then**
    return $\mathbb{P}(\Phi_1) \cdot \mathbb{P}(\Phi_2)$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and Φ_1, Φ_2 are syntactically independent **then**
    return $1 - (1 - \mathbb{P}(\Phi_1))(1 - \mathbb{P}(\Phi_2))$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and Φ_1, Φ_2 are disjoint **then**
    return $\mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2)$
end if
if $\Phi = \neg\Phi_1$ **then**
    return $1 - \mathbb{P}(\Phi_1)$
end if
Choose $X \in \mathrm{Var}(\Phi)$
return $\sum_{a \in \mathrm{dom}(X)} \mathbb{P}(\Phi[X \mapsto a])\, \mathbb{P}(X = a)$

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

Outline

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)

$$q(h) \leftarrow \exists n.\exists c.\text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

Room (R)

RoomNo	Type	HotelNo	Price	
R1	Suite	H1	\$50	X_1
R2	Single	H1	\$600	X_2
R3	Double	H1	\$80	X_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

ExpensiveHotels

HotelNo	
H1	0.375

(2)

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge (X_1 \vee X_2)$

(3)

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge^{i} (X_1 \vee^{i} X_2)$

(4)

Materialized views in TID databases (2)

Example (Continued)

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

Consider the query

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

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

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge (X_1 \vee X_2)$

Yes, combine lineages

ExpensiveHotels

HotelNo	
H1	0.375

No, dependency between ExpensiveHotels and Hotels lost

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge^{\mathrm{i}} (X_1 \vee^{\mathrm{i}} X_2)$

Yes, combine "compiled lineages" $\rightarrow$ Need to be able to combine compiled lineages efficiently!

ExpensiveHiltons

HotelNo	
H1	$[X_4 \wedge (X_1 \vee X_2)] \wedge X_4$

ExpensiveHiltons

HotelNo	
H1	$X_4 \wedge^{\mathrm{i}} (X_1 \vee X_2)$

Query compilation

- "Compile" Φ 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 $\Phi_1 \wedge \Phi_2$ (even when not independent!)
  - Key application in PDBMS: Compile materialized views
- Tractable compilation = circuit of size polynomial in database
  $\rightarrow$ 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*) representents a propositional formula Ψ
- Let Ψ be represented by a gate with children representing $\Psi_1, \ldots, \Psi_n$; we consider the following gates & restrictions:
  - *Independent-and* ($\wedge^{\mathrm{i}}$): $\Psi_1, \ldots, \Psi_n$ are syntactically independent
  - *Independent-or* ($\vee^{\mathrm{i}}$): $\Psi_1, \ldots, \Psi_n$ syntactically independent
  - *Disjoint-or* ($\vee^{\mathrm{d}}$): $\Psi_1, \ldots, \Psi_n$ are disjoint
  - *Not* ($\neg$): single child, represents $\neg\Psi$
  - *Conditional gate* (X): two children representing $X \wedge \Psi_1$ and $\neg X \wedge \Psi_2$, where $X \notin \mathrm{Var}(\Psi_1)$ and $X \notin \mathrm{Var}(\Psi_2)$
  - *Leaf node* (0, 1, X): represents `FALSE`, `TRUE`, X

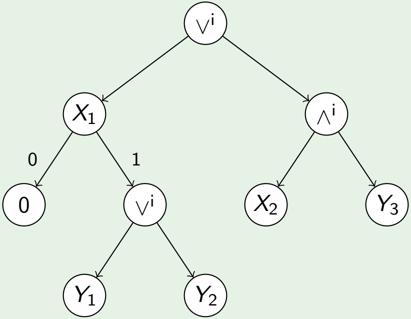
The different compilation targets restrict which and where gates may be used.

Restricted Boolean circuit (example)

Example

Who incriminates someone who has an alibi?

Lineage of unsafe plan: $\Phi_{\mathrm{M}} = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$



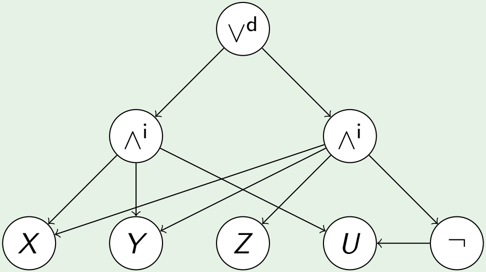
"Documents" the non-deterministic algorithm for intensional query evaluation.

Deterministic-decomposable normal form (d-DNF)

- Restricted to gates: $\wedge^{\mathsf{i}}$, $\vee^{\mathsf{d}}$, $\neg$
  - $\wedge^{\mathsf{i}}$-gates are called *decomposable* (D)
  - $\vee^{\mathsf{d}}$-gates are called *deterministic* (d)

Example

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



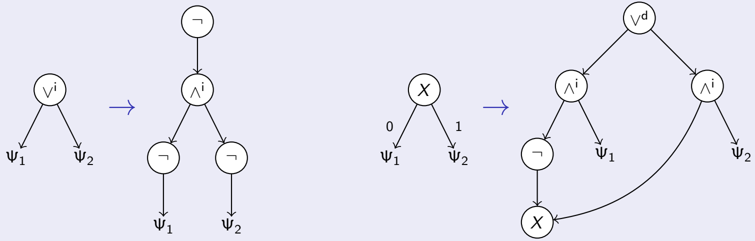
RBC and d-DNF

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 $\vee^{\mathrm{i}}$ and conditional nodes. Apply the transformations:



A $\vee^{\mathrm{i}}$-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

  $\rightarrow$ 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 ($\wedge$, $\vee$, $\neg$, 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

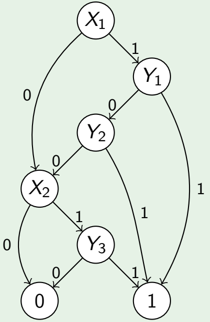
Free binary decision diagram (FBDD)

- 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: $\Phi_{\mathrm{M}} = X_1(Y_1 \vee Y_2) \vee X_2 Y_3$



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 Ψ_1 and Ψ_2 with a common variable order, we can compute an ODDB for $\Psi_1 \wedge \Psi_2$, $\Psi_1 \vee \Psi_2$, or $\neg\Psi_1$ in polynomial time. Note that Ψ_1 and Ψ_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 Φ is read-once (or *repetition-free*) if there exists a formula Φ' such that $\Phi \equiv \Phi'$ and every variable occurs at most once in Φ'.

Example

- $\Phi = X_1 \vee X_2 \vee X_3 \rightarrow$ read-once
- $\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3 \vee X_2 Y_4 \vee X_2 Y_5$
  - $\Phi' = X_1(Y_1 \vee Y_2) \vee X_2(Y_3 \vee Y_4 \vee Y_5) \rightarrow$ read-once
- $\Phi = XY \vee XU \vee YU \rightarrow$ not read-once

Theorem

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

Proof.

All $\wedge$'s and $\vee$'s are independent, and negation is easily handled. $\square$

When is a formula read-once? (1)

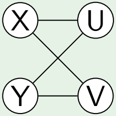
Definition

Let Φ be given in DNF such that no conjunct is a strict subset of some other conjunct. Φ 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 Φ and there is an edge $(X, Y) \in E$ if X and Y occur together in some conjunct.

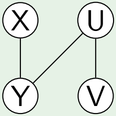
Example

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

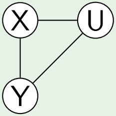
$XU \vee XV \vee YU \vee YV$



$XY \vee YU \vee UV$

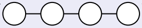


$XY \vee XU \vee YU$



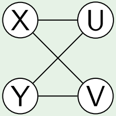
When is a formula read-once? (2)

Definition

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

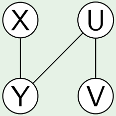
Example

$XU \vee XV \vee YU \vee YV$



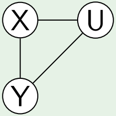
P_4-free
Normal
Read-once

$XY \vee YU \vee UV$



Not P_4-free
Normal
Not read-once

$XY \vee XU \vee YU$



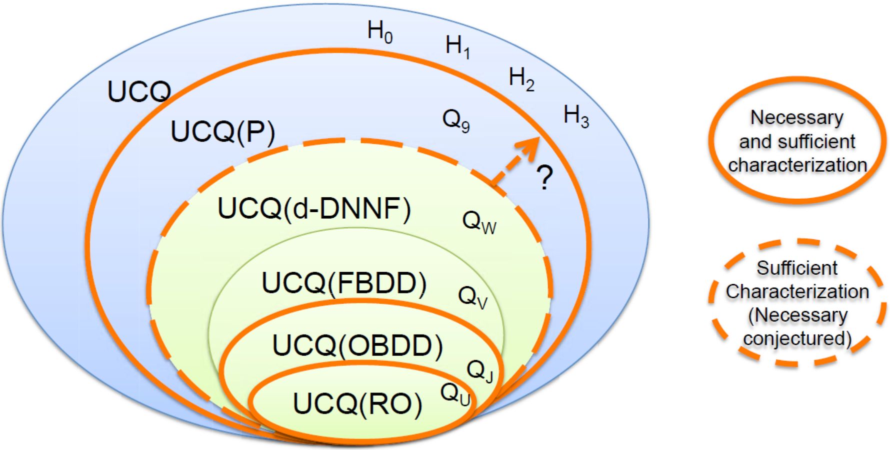
P_4-free
Not normal
Not read-once

Theorem

A unate formula is read-once iff it is P_4-free and normal.

Query compilation hierarchy

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



Outline

Why approximation?

- Exact inference may require exponential time $\rightarrow$ expensive
- Often absolute probability values of little interest; ranking desired
  $\rightarrow$ Good approximations of $\mathbb{P}(\Phi)$ 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 $\mathcal{RA}$-queries w.r.t. absolute error and for all $\mathcal{UCQ}$-queries w.r.t. relative error!

Probability bounds

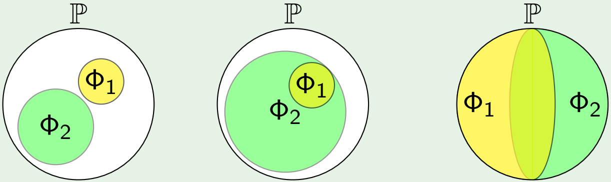
Theorem

Let Φ_1 and Φ_2 be propositional formulas. Then,

$$
\begin{array}{rcccl}
\max(\mathbb{P}(\Phi_1), \mathbb{P}(\Phi_2)) & \leq & \mathbb{P}(\Phi_1 \vee \Phi_2) & \leq & \overbrace{\min(\mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2), 1)}^{\textit{Boole's inequality / union bound}} \\
\underbrace{\max(0, \mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2) - 1)}_{\textit{via inclusion-exclusion}} & \leq & \mathbb{P}(\Phi_1 \wedge \Phi_2) & \leq & \min(\mathbb{P}(\Phi_1), \mathbb{P}(\Phi_2)).
\end{array}
$$

Example

Border cases:



$\mathbb{P}(\Phi_1 \vee \Phi_2)$	$\mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2)$	$\mathbb{P}(\Phi_2)$	1
$\mathbb{P}(\Phi_1 \wedge \Phi_2)$	0	$\mathbb{P}(\Phi_1)$	$\mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2) - 1$

Computation of probability intervals

Theorem

Let Φ_1 and Φ_2 be propositional formulas with bounds $[L_1, U_1]$ and $[L_2, U_2]$, respectively. Then,

$$\begin{aligned}
\Phi_1 \lor \Phi_2: &\quad [L, U] = [\max(L_1, L_2), \min(U_1 + U_2, 1)] \\
\Phi_1 \land \Phi_2: &\quad [L, U] = [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)] \\
\neg\Phi_1: &\quad [L, U] = [1 - U_1, 1 - L_1]
\end{aligned}$$

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

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

- $X_1 Y_1$: $[0.75, 0.85]$
- $X_1 Y_2$: $[0.65, 0.75]$
- $X_2 Y_3$: $[0.45, 0.65]$
- $X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$: $[0.75, 1]$

Incriminates

Witness	Suspect	$\mathbb{P}$	
Mary	Paul	0.9	X_1
Mary	John	0.8	X_2

Alibi

Suspect	Claim	$\mathbb{P}$	
Paul	Cinema	0.85	Y_1
Paul	Friend	0.75	Y_2
John	Bar	0.65	Y_3

Bounds can be computed in linear time in size of Φ.

Probability intervals and intensional query evaluation

if $\Phi = \Phi_1 \wedge \Phi_2$ and Φ_1, Φ_2 are syntactically independent **then**
    return $[L, U] = [L_1 \cdot L_2, U_1 \cdot U_2]$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and Φ_1, Φ_2 are syntactically independent **then**
    return $[L, U] = [L_1 \oplus L_2, U_1 \oplus U_2]$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and Φ_1, Φ_2 are disjoint **then**
    return $[L, U] = [L_1 + L_2, \min(U_1 + U_2, 1)]$
end if
if $\Phi = \neg\Phi_1$ **then**
    return $[L, U] = [1 - U_1, 1 - L_1]$
end if
Choose $X \in \mathrm{Var}(\Phi)$
Shannon expansion to $\Phi = \bigvee_i \Phi_i \wedge (X = a_i)$
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.

Probability intervals and intensional query evaluation (2)

Example

Incriminates

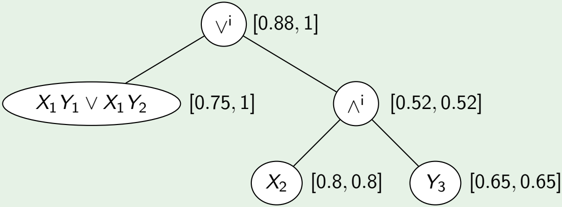
Witness	Suspect	$\mathbb{P}$	
Mary	Paul	0.9	X_1
Mary	John	0.8	X_2

Alibi

Suspect	Claim	$\mathbb{P}$	
Paul	Cinema	0.85	Y_1
Paul	Friend	0.75	Y_2
John	Bar	0.65	Y_3

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

- $X_1 Y_1$: $[0.75, 0.85]$
- $X_1 Y_2$: $[0.65, 0.75]$
- $X_2 Y_3$: $[0.45, 0.65]$
- Φ : $[0.75, 1]$



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 ϵ-approximation: $U - L \leq 2\epsilon \rightarrow$ choose $\hat{p} \in [U - \epsilon, L + \epsilon]$
  - Relative ϵ-approximation:
    $(1-\epsilon)U \leq (1+\epsilon)L \rightarrow$ choose $\hat{p} \in [(1-\epsilon)U, (1+\epsilon)L]$
- But: no apriori runtime bounds!

Definition

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

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

it is an *relative ϵ-approximation* of p if

$$(1-\epsilon)p \leq \hat{p} \leq (1+\epsilon)p.$$

Monte-Carlo approximation w/ naive estimator

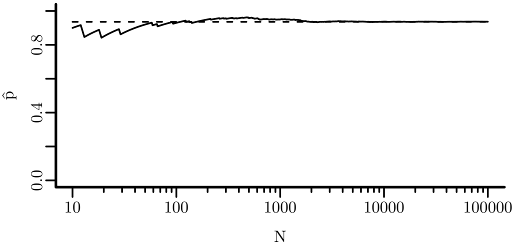
Let Φ be a propositional formula with $V(\Phi) = \{ X_1, \ldots, X_l \}$.

- Pick a value n and for $k \in \{ 1, 2, \ldots, n \}$, do
  1. Pick a random assignment θ_k by setting

$$X_i = \begin{cases} \texttt{TRUE} & \text{with probability } \mathbb{P}(X_i) \\ \texttt{FALSE} & \text{otherwise} \end{cases}$$

  2. Evaluate $Z_k = \Phi[\theta_k]$
- Return $\hat{p} = \frac{1}{n} \sum_k Z_k$

How good is this algorithm?



$\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$

X_1	X_2	Y_1	Y_2	Y_3		Z_k	$\hat{p}$
1	1	1	1	1		1	1.00
1	0	1	1	0		1	1.00
1	0	0	0	1		0	0.66
1	0	1	1	1		1	0.75
1	1	1	0	1		1	0.80
1	1	1	1	1		1	0.83
1	1	1	0	0		1	0.85
1	1	1	1	1		1	0.88
1	1	1	1	1		1	0.89
1	1	1	1	1		1	0.90

Naive estimator: expected value

Theorem

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

Proof.

$$
\begin{aligned}
\mathbb{E}\left[\,\hat{p}\,\right] &= \mathbb{E}\left[\,\frac{1}{n}\sum_{k=1}^{n} Z_k\,\right] = \frac{1}{n}\sum_{k=1}^{n}\mathbb{E}\left[\,Z_k\,\right] \\
&= \mathbb{E}\left[\,Z_1\,\right] \\
&= \sum_{\theta} \Phi[\theta]\,\mathbb{P}\left(\,\theta\,\right) \\
&= \mathbb{P}\left(\,\Phi\,\right).
\end{aligned}
$$

□

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 $\mathbb{E}\left[\, Z_1 \,\right] = p$ and set $\bar{Z} = \frac{1}{n} \sum_k Z_k$. Then,

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

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 $\bar{Z}$ of heads
- Estimate p using $\bar{Z}$
- Then: Probability that relative error larger than γ
    1. Decreases exponentially with increasing number of flips n
    2. Decreases with increasing error bound γ
    3. Decreases with increasing probability of heads p

Very important result with many applications!

Chernoff bound (2)

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,

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

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 $\mathbb{P}(|X| \geq a) \leq \mathbb{E}[|X|]/a$ to obtain

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

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. $\square$

Naive estimator: absolute (ϵ,δ)-approximation (1)

Theorem (sampling theorem)

To obtain an absolute ϵ-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.

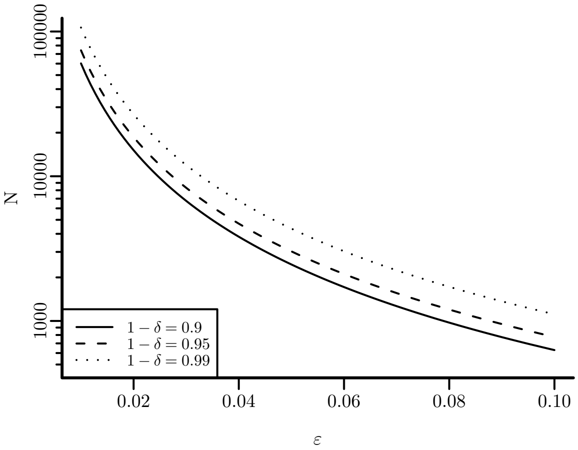
Take $\gamma = \epsilon/p$ and apply the Chernoff bound to obtain

$$\begin{aligned}\mathbb{P}\left(\left|\bar{Z} - p\right| \geq \epsilon\right) &\leq 2\exp\left(-\frac{\epsilon^2/p^2}{2+\epsilon/p} pn\right) = 2\exp\left(-\frac{\epsilon^2}{2p+\epsilon} n\right) \\ &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon} n\right)\end{aligned}$$

since $p \leq 1$. Now solve RHS $\leq \delta$ for n. □

Naive estimator: absolute (ϵ,δ)-approximation (2)

The number of sampling steps given by the sampling theorem is independent of Φ.



Naive estimator: relative (ϵ,δ)-approximation (1)

Theorem

To obtain a relative ϵ-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.

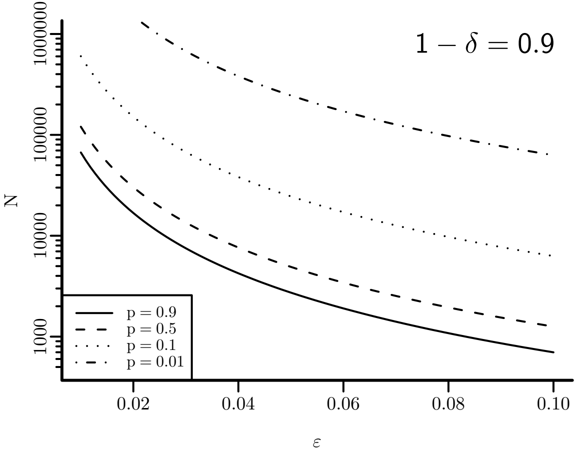
Take $\gamma = \epsilon$ and apply the Chernoff bound to obtain

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

Now solve RHS $\leq \delta$ for n. $\square$

Naive estimator: relative (ϵ,δ)-approximation (2)

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



Why care about relative ϵ-approximation?

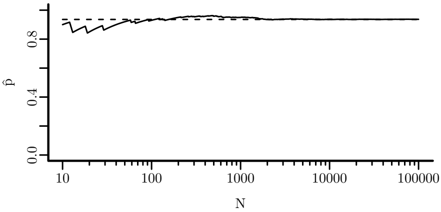
1. Absolute error ill-suited to compare estimates of small probabilities
   - $p_1 = 0.001$, $p_2 = 0.01$, $\epsilon = 0.1$
   - Absolute error: $I_1 = [0, 0.101]$, $I_2 = [0, 0.11]$
   - Relative error: $I_1 = [0.0009, 0.0011]$, $I_2 = [0.009, 0.011]$

   $\rightarrow$ Ranking of tuples more sensitive to absolute error
2. For $p \in [0, 1)$, relative error ϵ is *always tighter* than absolute error ϵ (esp. when probabilities are small)

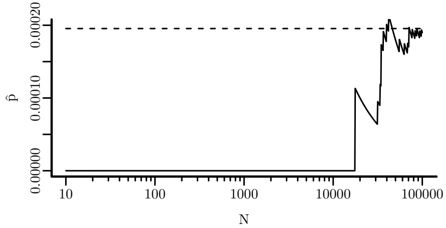
Can we get a relative ϵ-approximation in which the minimum number of sampling steps does not depend on $\mathbb{P}(\Phi)$?

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, Φ not satisfied on most samples
  $\rightarrow$ Slow convergence

> Idea: Change the sampling strategy so that Φ is satisfied on every sample.

Karp-Luby estimator (basic idea)

Let Φ be a propositional *DNF formula* with $V(\Phi) = \{ X_1, \dots, X_l \}$, 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).

$$\begin{aligned}
\mathbb{P}(\Phi) &= \mathbb{P}(C_1) + \mathbb{P}(\neg C_1 \wedge C_2) + \cdots \\
&\qquad + \mathbb{P}(\neg(C_1 \vee \cdots \vee C_{m-1}) \wedge C_m) \\
&= \mathbb{P}(\mathtt{TRUE} \mid C_1)\,\mathbb{P}(C_1) + \mathbb{P}(\neg C_1 \mid C_2)\,\mathbb{P}(C_2) + \cdots \\
&\qquad + \mathbb{P}(\neg(C_1 \vee \cdots \vee C_{m-1}) \mid C_m)\,\mathbb{P}(C_m) \\
&= Q\,[\mathbb{P}(\mathtt{TRUE} \mid C_1)\, q_1/Q + \mathbb{P}(\neg C_1 \mid C_2)\, q_2/Q + \cdots \\
&\qquad + \mathbb{P}(\neg(C_1 \vee \cdots \vee C_{m-1}) \mid C_m)\, q_m/Q]
\end{aligned}$$

Idea of Karp-Luby estimator:

1. q_i/Q is computed exactly (in linear time)
2. $\mathbb{P}(\neg(C_1 \vee \cdots \vee C_{i-1}) \mid C_i)$ are estimated
   - Impact of estimate proportional to $\mathbb{P}(C_i)$
     $\rightarrow$ 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 θ_k
     - ★ For a variable $X \in V(C_i)$
       $$X = \begin{cases} \texttt{TRUE} & \text{if } X \text{ is positive in } C_i \\ \texttt{FALSE} & \text{if } X \text{ is negative in } C_i \end{cases}$$
       $\rightarrow$ Clause C_i is satisfied (and thus Φ)
     - ★ For the other variables $X \notin V(C_i)$
       $$X = \begin{cases} \texttt{TRUE} & \text{with probability } \mathbb{P}(X) \\ \texttt{FALSE} & \text{otherwise} \end{cases}$$
       $\rightarrow$ All other variables take random values
  3. Evaluate
     $$Z_k = \begin{cases} 1 & \text{if } \neg(\bigvee_{1 \le j < i} C_j[\theta]) \\ 0 & \text{otherwise} \end{cases}$$
- Return $\hat{p} = \dfrac{Q}{n} \sum_{k=1}^{n} Z_k$

Example of KL estimator

$\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$

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

i		X_1	X_2	Y_1	Y_2	Y_3		C_1	C_2	C_3		Z_k	$\hat{p}$
1		1	1	1	1	0		1	1	0		1	0.196
3		0	1	1	1	1		0	0	1		1	0.196
2		1	1	1	1	0		1	1	0		0	0.131
1		1	1	1	1	0		1	1	0		1	0.147
1		1	1	1	1	0		1	1	0		1	0.157
2		1	0	1	1	0		1	1	0		0	0.131

KL estimator: expected value

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.

$$\begin{aligned}
\mathbb{E}[\,\hat{p}\,] &= \mathbb{E}\Big[\,\frac{Q}{n}\sum_{k=1}^{n} Z_k\,\Big] = Q\,\mathbb{E}[\,Z_1\,] = Q\,\mathbb{E}[\,\mathbb{E}[\,Z_1 \mid C_i \text{ picked}\,]\,] \\
&= Q\sum_{i=1}^{m}\frac{q_i}{Q}\,\mathbb{E}[\,Z_1 \mid C_i \text{ picked}\,] \\
&= \sum_{i=1}^{m}\mathbb{P}(\,C_i\,)\,\mathbb{P}(\neg \bigvee_{1\le j<i} C_j \mid C_i\,) \\
&= \mathbb{P}(\,\Phi\,)\,.
\end{aligned}$$

$\square$

KL estimator: relative (ϵ,δ)-approximation

Theorem

To obtain a relative ϵ-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.

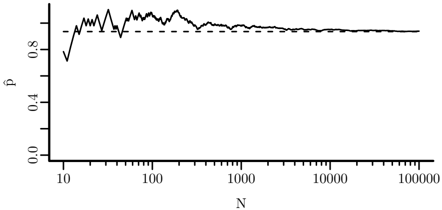
Use the Chernoff bound with $\gamma = \epsilon$ and $\mathbb{E}\left[\bar{Z}\right] = Q^{-1}p$.

$$\begin{aligned}
\mathbb{P}\left(\left|\bar{Z} - Q^{-1}p\right| \geq \epsilon Q^{-1}p\right) &\leq 2\exp\left(-\epsilon^2/(2+\epsilon)Q^{-1}pn\right) \\
\mathbb{P}\left(\left|Q^{-1}\hat{p} - Q^{-1}p\right| \geq \epsilon Q^{-1}p\right) &= \mathbb{P}\left(\left|\hat{p} - p\right| \geq \epsilon p\right) \\
&\leq 2\exp\left(-\epsilon^2/(2+\epsilon)m^{-1}n\right),
\end{aligned}$$

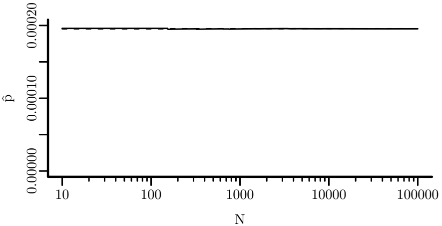
since $mp \geq Q$. Now solve RHS $\leq \delta$ for n. $\square$

KL estimator: discussion

- KL estimator provides relative (ϵ,δ)-approximation in polynomial time in *size* of Φ and $\frac{1}{\epsilon}$
  $\rightarrow$ *fully polynomial-time randomized approximation scheme* (*FPTRAS*)
- Example: $\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$



Large probabilities



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

- Requires DNF (=why-provenance; polynomial in DB size for $\mathcal{UCQ}$)

For ϵ, δ 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

Lessons learned

- Relational calculus is a great tool for query analysis & manipulation
- Query evaluation computes marginal probabilities $\mathbb{P}(\, t \in q(\mathcal{D})\,)$
- 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 $\mathbf{R}_6$-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