Uncertain Data Management
Querying Probabilistic Databases

Antoine Amarilli[1], **Silviu Maniu**[2]

[1]Télécom ParisTech

[2]Université Paris-Sud

January 9th, 2017

Credits

Stucture, flow, and examples are based on the book Probabilistic Databases by D. Suciu, D. Olteanu, C. Ré, C. Koch (Morgan&Claypool, 2011)

PDFs of the slides available at http://silviu.maniu.info/teaching/

Introduction

Reminder: we work with *relational queries*

- in the following: queries in FO or relational calculus

Introduction

Reminder: we work with *relational queries*

- in the following: queries in FO or relational calculus

Booking

date	teacher	room	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	equipment	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Introduction

Booking

date	**teacher**	**room**	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	**equipment**	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Query: List dates when courses are taught in rooms with projectors

Introduction

Booking

date	**teacher**	**room**	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	**equipment**	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Query: List dates when courses are taught in rooms with projectors

- SQL: `SELECT B.date FROM B, R WHERE B.room=R.room AND R.equipment='projector'`

Introduction

Booking

date	**teacher**	**room**	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	**equipment**	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Query: List dates when courses are taught in rooms with projectors

- SQL: `SELECT B.date FROM B, R WHERE B.room=R.room AND R.equipment='projector'`
- calculus: $Q(d)\ :\ \exists t, r. B(d, t, r) \wedge R(r, \text{'projector'})$

Introduction

Booking

date	**teacher**	**room**	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	**equipment**	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Query: Is any course taught in a room without a projector?
(**Boolean query**)

Introduction

Booking

date	**teacher**	**room**	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	**equipment**	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Query: Is any course taught in a room without a projector? (**Boolean query**)

- SQL: `SELECT DISTINCT 1 FROM B, R WHERE B.room=R.room AND R.equipment='none'`

Introduction

Booking

date	**teacher**	**room**	
3011	Silviu	C42	x_1
0712	Antoine	C42	x_2
1412	Silviu	C017	x_3
0401	Antoine	C017	x_4

Room

room	**equipment**	
C42	projector	y_1
C42	none	$\neg y_1$
C017	projector	y_2
C017	none	$\neg y_2$

Query: Is any course taught in a room without a projector? (**Boolean query**)

- SQL: `SELECT DISTINCT 1 FROM B, R WHERE B.room=R.room AND R.equipment='none'`
- calculus: $Q() : \exists d, t, r. B(d, t, r) \wedge R(r, \text{'none'})$

Query Evaluation Problem

Definition

For a fixed query Q, a database $\mathcal{D}$, and a possible answer tuple a, compute its marginal probability $\mathrm{P}(a \in Q)$.

Lineage

How can we answer such queries on probabilistic databases?

Lineage

How can we answer such queries on probabilistic databases?

- *identify* the conditions when the query is true, or a tuple is an answer to a query

Lineage

How can we answer such queries on probabilistic databases?

- *identify* the conditions when the query is true, or a tuple is an answer to a query
- *estimate* the probability of each possible output tuple in the query

Lineage

$$Q(): \ \exists d, t, r. B(d,t,r) \land R(r, \text{'none'})$$

Lineage

$$Q(): \ \exists d, t, r. B(d, t, r) \wedge R(r, \text{'none'})$$

Booking ⋈ Room

date	teacher	room	equipment
3011	Silviu	C42	projector
3011	Silviu	C42	none
0712	Antoine	C42	projector
0712	Antoine	C42	none
1412	Silviu	C017	projector
1412	Silviu	C017	none
0401	Antoine	C017	projector
0401	Antoine	C017	none

Lineage

$$Q() : \ \exists d, t, r. B(d, t, r) \land R(r, \text{'none'})$$

Booking ⋈ Room

date	teacher	room	equipment	
3011	Silviu	C42	projector	$x_1 y_1$
3011	Silviu	C42	none	$x_1 \neg y_1$
0712	Antoine	C42	projector	$x_2 y_1$
0712	Antoine	C42	none	$x_2 \neg y_1$
1412	Silviu	C017	projector	$x_3 y_2$
1412	Silviu	C017	none	$x_3 \neg y_2$
0401	Antoine	C017	projector	$x_4 y_2$
0401	Antoine	C017	none	$x_4 \neg y_2$

Lineage

$$Q() : \exists d, t, r. B(d,t,r) \land R(r, \text{'none'})$$

Booking ⋈ Room

date	teacher	room	equipment	
3011	Silviu	C42	projector	$x_1 y_1$
3011	Silviu	C42	none	$x_1 \neg y_1$
0712	Antoine	C42	projector	$x_2 y_1$
0712	Antoine	C42	none	$x_2 \neg y_1$
1412	Silviu	C017	projector	$x_3 y_2$
1412	Silviu	C017	none	$x_3 \neg y_2$
0401	Antoine	C017	projector	$x_4 y_2$
0402	Antoine	C017	none	$x_4 \neg y_2$

Lineage

$$Q() \iff x_1 \neg y_1 \lor x_2 \neg y_1 \lor x_3 \neg y_2 \lor x_4 \neg y_2$$

Booking ⋈ Room

date	teacher	room	equipment	
3011	Silviu	C42	projector	$x_1 y_1$
3011	Silviu	C42	none	$x_1 \neg y_1$
0712	Antoine	C42	projector	$x_2 y_1$
0712	Antoine	C42	none	$x_2 \neg y_1$
1412	Silviu	C017	projector	$x_3 y_2$
1412	Silviu	C017	none	$x_3 \neg y_2$
0401	Antoine	C017	projector	$x_4 y_2$
0402	Antoine	C017	none	$x_4 \neg y_2$

Lineage

The propositional formula $x_1 \neg y_1 \lor x_2 \neg y_1 \lor x_3 \neg y_2 \lor x_4 \neg y_2$ is called the lineage of $Q()$

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;
- for a Boolean query Q;

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;
- for a Boolean query Q;

The lineage Φ_Q is defined inductively as follows:

- if t is a ground tuple, then Φ_t is its propositional formula

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;
- for a Boolean query Q;

The lineage Φ_Q is defined inductively as follows:

- if t is a ground tuple, then Φ_t is its propositional formula
- $\Phi_{Q_1 \wedge Q_2} = \Phi_{Q_1} \wedge \Phi_{Q_2}$

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* ADom$(\mathcal{D})$;
- for a Boolean query Q;

The lineage Φ_Q is defined inductively as follows:

- if t is a ground tuple, then Φ_t is its propositional formula
- $\Phi_{Q_1 \wedge Q_2} = \Phi_{Q_1} \wedge \Phi_{Q_2}$
- $\Phi_{Q_1 \vee Q_2} = \Phi_{Q_1} \vee \Phi_{Q_2}$

Lineage

The propositional formula $x_1\neg y_1 \vee x_2\neg y_1 \vee x_3\neg y_2 \vee x_4\neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;
- for a Boolean query Q;

The lineage Φ_Q is defined inductively as follows:

- if t is a ground tuple, then Φ_t is its propositional formula
- $\Phi_{Q_1 \wedge Q_2} = \Phi_{Q_1} \wedge \Phi_{Q_2}$
- $\Phi_{Q_1 \vee Q_2} = \Phi_{Q_1} \vee \Phi_{Q_2}$
- $\Phi_{\neg Q} = \neg \Phi_Q$

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;
- for a Boolean query Q;

The lineage Φ_Q is defined inductively as follows:

- if t is a ground tuple, then Φ_t is its propositional formula
- $\Phi_{Q_1 \wedge Q_2} = \Phi_{Q_1} \wedge \Phi_{Q_2}$
- $\Phi_{Q_1 \vee Q_2} = \Phi_{Q_1} \vee \Phi_{Q_2}$
- $\Phi_{\neg Q} = \neg \Phi_Q$
- $\Phi_{\exists x.Q} = \bigvee_{a \in \mathrm{ADom}(\mathcal{D})} \Phi_{Q(a/x)}$

Lineage

The propositional formula $x_1 \neg y_1 \vee x_2 \neg y_1 \vee x_3 \neg y_2 \vee x_4 \neg y_2$ is called the lineage of $Q()$

Formally:

- for a database $\mathcal{D}$ having *active domain* $\mathrm{ADom}(\mathcal{D})$;
- for a Boolean query Q;

The lineage Φ_Q is defined inductively as follows:

- if t is a ground tuple, then Φ_t is its propositional formula
- $\Phi_{Q_1 \wedge Q_2} = \Phi_{Q_1} \wedge \Phi_{Q_2}$
- $\Phi_{Q_1 \vee Q_2} = \Phi_{Q_1} \vee \Phi_{Q_2}$
- $\Phi_{\neg Q} = \neg \Phi_Q$
- $\Phi_{\exists x.Q} = \bigvee_{a \in \mathrm{ADom}(\mathcal{D})} \Phi_{Q(a/x)}$
- $\Phi_{\mathrm{true}} = \mathrm{true}$, $\Phi_{\mathrm{false}} = \mathrm{false}$

For non-Boolean queries with head variables $\bar{x}$, and for each possible answer $\bar{a}$, its lineage is defined as the lineage of $Q(\bar{a}/\bar{x})$

For non-Boolean queries with head variables $\bar{x}$, and for each possible answer $\bar{a}$, its lineage is defined as the lineage of $Q(\bar{a}/\bar{x})$

- the query evaluation problem reduces to computing the probability of its lineage

Proposition

For $Q(\bar{x})$ and $\mathcal{D}$ a pc-database, the probability of a possible answer $\bar{a}$ to Q is equal to the probability of its lineage formula:

$$\mathrm{P}(\bar{a} \in Q) = \mathrm{P}(\Phi_{Q(\bar{a}/\bar{x})})$$

Table of contents

Possible Worlds Semantics

Recall that $\mathrm{P}(a \in Q) = \mathrm{P}(\Phi_{Q(a/x)})$

Possible Worlds Semantics

Recall that $\mathrm{P}(a \in Q) = \mathrm{P}(\Phi_{Q(a/x)})$

- we know $\mathrm{P}(\Phi) = \sum_{\theta \in w(\Phi)} \mathrm{P}(\theta)$, where $w(\Phi)$ is the set of all possible assignments of Φ (possible worlds)

Possible Worlds Semantics

Recall that $\mathrm{P}(a \in Q) = \mathrm{P}(\Phi_{Q(a/x)})$

- we know $\mathrm{P}(\Phi) = \sum_{\theta \in w(\Phi)} \mathrm{P}(\theta)$, where $w(\Phi)$ is the set of all possible assignments of Φ (possible worlds)
- iterating through each $\theta \in w(\Phi)$ is exponential (if booleans, 2^n possible assignments)

Possible Worlds Semantics

Recall that $\mathrm{P}(a \in Q) = \mathrm{P}(\Phi_{Q(a/x)})$

- we know $\mathrm{P}(\Phi) = \sum_{\theta \in w(\Phi)} \mathrm{P}(\theta)$, where $w(\Phi)$ is the set of all possible assignments of Φ (possible worlds)
- iterating through each $\theta \in w(\Phi)$ is exponential (if booleans, 2^n possible assignments)
- inefficient even prohibitive; can we do better?

Possible Worlds Semantics

Recall that $\mathrm{P}(a \in Q) = \mathrm{P}(\Phi_{Q(a/x)})$

- we know $\mathrm{P}(\Phi) = \sum_{\theta \in w(\Phi)} \mathrm{P}(\theta)$, where $w(\Phi)$ is the set of all possible assignments of Φ (possible worlds)
- iterating through each $\theta \in w(\Phi)$ is exponential (if booleans, 2^n possible assignments)
- inefficient even prohibitive; can we do better?
- not in the general case!

Model Counting

Definition

The **Model Counting Problem**: given a formula Φ, count the number of satisfying assignments Φ, $\#\Phi = |w(\Phi)|$

Model Counting

Definition

The **Model Counting Problem**: given a formula Φ, count the number of satisfying assignments Φ, $\#\Phi = |w(\Phi)|$

- special case of probability computation: any algorithm for computing $\mathrm{P}(\Phi)$ can be used to compute $\#\Phi$ (if $p = 0.5$ everywhere, then $\#\Phi = \mathrm{P}(\Phi) \cdot 2^n$)

Model Counting

Definition

The **Model Counting Problem**: given a formula Φ, count the number of satisfying assignments Φ, $\#\Phi = |w(\Phi)|$

- special case of probability computation: any algorithm for computing $\mathrm{P}(\Phi)$ can be used to compute $\#\Phi$ (if $p = 0.5$ everywhere, then $\#\Phi = \mathrm{P}(\Phi) \cdot 2^n$)
- problem known as #SAT, in complexity class #P (given a polynomial-time, non-deterministic Turing machine, compute the number of accepting computations)

Probability Computation

Definition
The **Probability Computation Problem**: given a formula Φ and a probability $\mathrm{P}(X)$ for each variable X, compute $\mathrm{P}(\Phi) = \sum_{\theta \in w(Q)} \mathrm{P}(\theta)$

Probability Computation

Definition

The **Probability Computation Problem**: given a formula Φ and a probability $\mathrm{P}(X)$ for each variable X, compute $\mathrm{P}(\Phi) = \sum_{\theta \in w(Q)} \mathrm{P}(\theta)$

- probability computation is hard for #P, but not in #P (not a counting problem)

Probability Computation

Definition

The **Probability Computation Problem**: given a formula Φ and a probability $\mathrm{P}(X)$ for each variable X, compute $\mathrm{P}(\Phi) = \sum_{\theta \in w(Q)} \mathrm{P}(\theta)$

- probability computation is hard for $\#$P, but not in $\#$P (not a counting problem)
- if we assume $\mathrm{P}(X_i) = m_i/n_i$ (rational number), $\sum n_i = N$, then $N \cdot \mathrm{P}(\Phi)$ is an integer

Probability Computation

Definition
The **Probability Computation Problem**: given a formula Φ and a probability $\mathrm{P}(X)$ for each variable X, compute $\mathrm{P}(\Phi) = \sum_{\theta \in w(Q)} \mathrm{P}(\theta)$

- probability computation is hard for #P, but not in #P (not a counting problem)
- if we assume $\mathrm{P}(X_i) = m_i/n_i$ (rational number), $\sum n_i = N$, then $N \cdot \mathrm{P}(\Phi)$ is an integer
- hence, computing $N \cdot \mathrm{P}(\Phi)$ is in #P

Probability Computation In Probabilistic Databases

What about probability computation in probabilistic databases?

- interested in the *data complexity* of the query evaluation problem
- tractable if data complexity is polynomial

Probability Computation In Probabilistic Databases

What about probability computation in probabilistic databases?

- interested in the *data complexity* of the query evaluation problem
- tractable if data complexity is polynomial

Example class of intractable (or unsafe) queries:

$$H_0 = R(x), S(x, y), T(y)$$
$$H_1 = R(x_0), S(x_0, y_0) \vee S(x_1, y_1), T(y_1)$$
$$\dots$$

Probability Computation In Probabilistic Databases

R

x_1	0.5
x_2	0.5
...	

S

x_1	y_1	1
x_1	y_2	1
...		
x_2	y_1	1
x_2	y_2	1
...		

T

y_1	0.5
y_2	0.5
...	

Probability Computation In Probabilistic Databases

R

x_1	0.5
x_2	0.5
$\dots$	

S

x_1	y_1	1
x_1	y_2	1
$\dots$		
x_2	y_1	1
x_2	y_2	1
$\dots$		

T

y_1	0.5
y_2	0.5
$\dots$	

Let us analyze $H_0 = R(x), S(x, y), T(y)$ on a tuple-independent database

Probability Computation In Probabilistic Databases

R

x_1	0.5
x_2	0.5
...	

S

x_1	y_1	1
x_1	y_2	1
...		
x_2	y_1	1
x_2	y_2	1
...		

T

y_1	0.5
y_2	0.5
...	

- each possible tuple is of the form $W = \langle R^W, S, T^W \rangle$, $\Phi(X_i) = \text{true} \iff X_i \in R^W$, similarly for $\Phi(Y_i)$

Probability Computation In Probabilistic Databases

R

x_1	0.5
x_2	0.5
...	

S

x_1	y_1	1
x_1	y_2	1
...		
x_2	y_1	1
x_2	y_2	1
...		

T

y_1	0.5
y_2	0.5
...	

- each possible tuple is of the form $W = \langle R^W, S, T^W \rangle$, $\Phi(X_i) = \text{true} \iff X_i \in R^W$, similarly for $\Phi(Y_i)$
- H_0 true iff $\exists x_i, x_j . R^W(x_i) S(x_i, y_j) T^W(y_j) = \text{true}$; 1-1 correspondence with possible worlds

Probability Computation In Probabilistic Databases

R

x_1	0.5
x_2	0.5
$\dots$	

S

x_1	y_1	1
x_1	y_2	1
$\dots$		
x_2	y_1	1
x_2	y_2	1
$\dots$		

T

y_1	0.5
y_2	0.5
$\dots$	

- each possible tuple is of the form $W = \langle R^W, S, T^W \rangle$, $\Phi(X_i) = \text{true} \iff X_i \in R^W$, similarly for $\Phi(Y_i)$
- H_0 true iff $\exists x_i, x_j . R^W(x_i) S(x_i, y_j) T^W(y_j) = \text{true}$; 1-1 correspondence with possible worlds
- $\#\Phi = 2^n \mathrm{P}(H_0)$ – an oracle for computing $\mathrm{P}(H_0)$ can be used to compute $\#\Phi \rightsquigarrow \mathrm{P}(H_0)$ is hard for $\#$P

Probability Computation In Probabilistic Databases

Generally, H_k for $k \geqslant 0$ are hard for $\#$P

Probability Computation In Probabilistic Databases

Generally, H_k for $k \geqslant 0$ are hard for $\#$P

H_k are used as primitives for finding intractable queries

Probability Computation In Probabilistic Databases

Generally, H_k for $k \geqslant 0$ are hard for #P

H_k are used as primitives for finding intractable queries

H_k are not the only queries that are intractable, and there exist safe (tractable) queries

Probability Computation In Probabilistic Databases

Generally, H_k for $k \geqslant 0$ are hard for #P

H_k are used as primitives for finding intractable queries

H_k are not the only queries that are intractable, and there exist safe (tractable) queries

Next:

- extensional query evaluation: reasoning on the query Q directly
- intensional query evaluation: reasoning on the lineage of the query Φ_Q