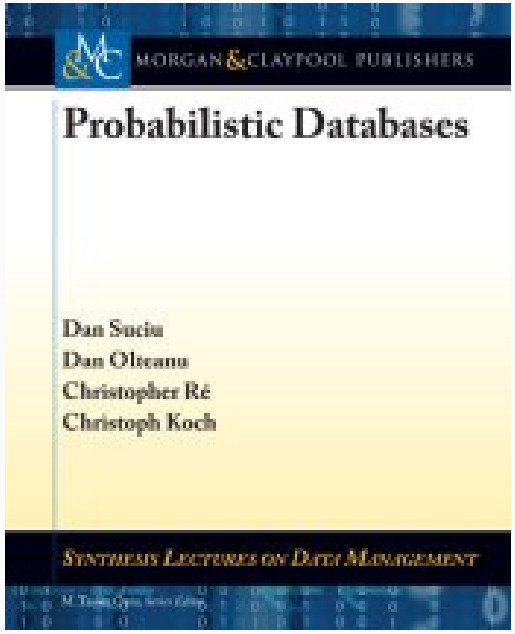
Probabilistic Databases

Dan Olteanu (Oxford)

London, November 2017

The National Archives



Probabilistic Databases

For the purpose of this introductory talk:

Probabilistic data =

- Relational data
  +
- Probabilities that measure the degree of uncertainty in the data.

Long-term research challenges:

- Models for probabilistic data to capture data and its uncertainty.
- Query evaluation = Probabilistic inference
  Query answers are annotated with output probabilities.

Research Development Map

Early work (80s and 90s):

- Basic data models and query processing

  Wong'82, Shoshani'82, Cavallo & Pittarelli'87, Barbara'92, Lakshmanan'97,'01, Fuhr& Roellke'97, Zimanyi'97.

Recent wave (2004 - now):

- Computational complexity of query evaluation
- Probabilistic database systems, *just a few examples*:
  - UW (MystiQ)
  - Stanford (Trio)
  - **Cornell & Oxford (MayBMS/SPROUT)**
  - IBM Almaden & Rice (MCDB)
  - Maryland, Waterloo, UBC, Florida, Purdue, Wisconsin
  - **LogicBlox & Technion & Oxford (PPDL)**

Talk Outline

Why Do We Care About Probabilistic Databases?

Probabilistic relational data is commonplace. It accommodates several possible interpretations of the data weighted by probabilities.

- **Information extraction**: Probabilistic data inferred from unstructured data (e.g., web) text using statistical models
  Google Knowledge Vault, DeepDive, NELL

- **Manually entered data**
  Represent several possible readings with MayBMS [Antova'07]
  Infer missing data with meta-rule semi-lattices [Stoyanovich'11]
  Manage OCR data with Staccato/Google OCRopus [Kumar'12]

- **Data cleaning**
  Represent several possible data repairs [Beskales'09]

- **Data integration**
  Google Squared and SPROUT^2 [Fink'11]

- **Risk management** (Decision support queries, hypothetical queries); ...

Possible segmentations of unstructured text [Sarawagi'06]

`52-A Goregaon West Mumbai 400 076`

ID	HouseNo	Area	City	PinCode	**P**
1	52	Goregaon West	Mumbai	400 062	**0.1**
1	52-A	Goregaon	West Mumbai	400 062	**0.2**
1	52-A	Goregaon West	Mumbai	400 062	**0.4**
1	52	Goregaon	West Mumbai	400 062	**0.2**
...	...	...	...	...	

- Probabilities obtained using probabilistic extraction models (e.g., CRF)
  The probabilities correlate with the precision of the extraction.
- The output is a ranked list of possible extractions
- Several segmentations are required to cover most of the probability mass and improve recall
  Avoid empty answer to queries such as *Find areas in 'West Mumbai'*

Continuously-Improving Information Extraction

Never-Ending Language Learner (NELL) database [Mitchell'15]

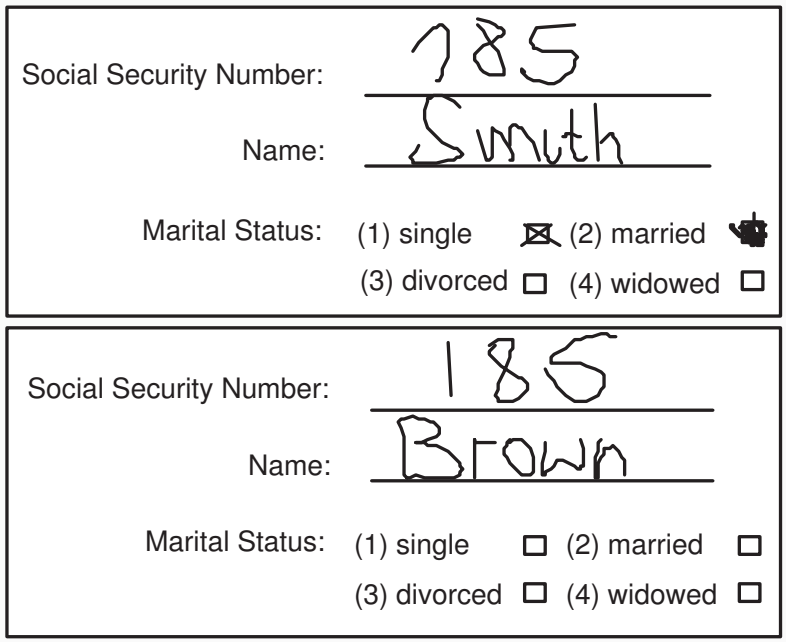
Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
biscutate_swift is an animal	211	18-feb-2011	100.0
pedigree_animals is a mammal	210	17-feb-2011	99.5
poppy_seed_holiday_bread is a baked good	212	20-feb-2011	100.0
manuel_criado_de_val is a South American person	210	17-feb-2011	99.5
dillon_county_airport is an airport	210	17-feb-2011	93.8
the sports team toronto_blue_jays was the winner of n1993_world_series	212	20-feb-2011	96.9
mozart is a person who died at the age of 35	210	17-feb-2011	96.9
peoria and arizona are proxies for eachother	210	17-feb-2011	99.9
wutv_tv is a TV affiliate of the network fox	210	17-feb-2011	96.9
white_stripes collaborates with jack_white	210	17-feb-2011	93.8

Manu e a l?y-enter d census data

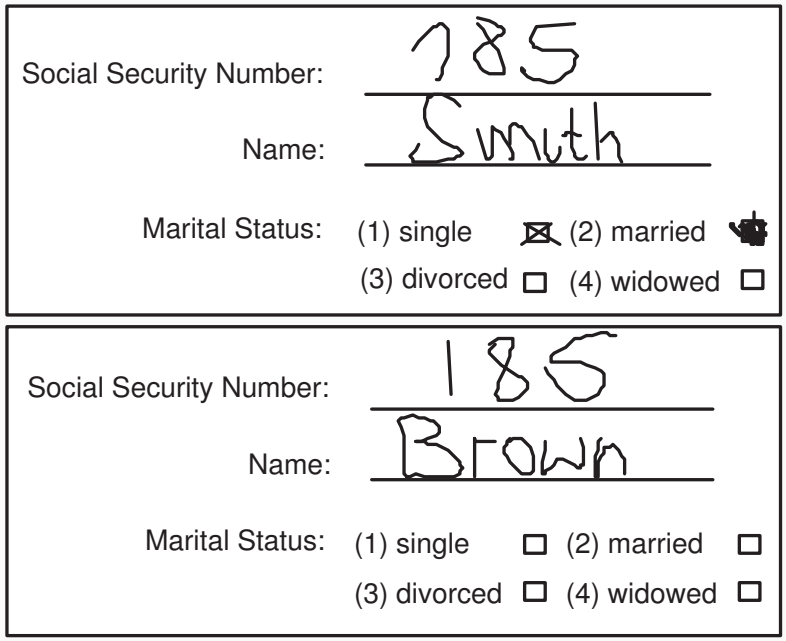
MayBMS manages 10^{10^6} possible readings of census data [Antova'07]



We want to enter the information from forms like these into a database.

- What is the marital status of the first resp. the second person?
- What are the social security numbers? 185? 186? 785?

Manu e a l?y-enter d census data



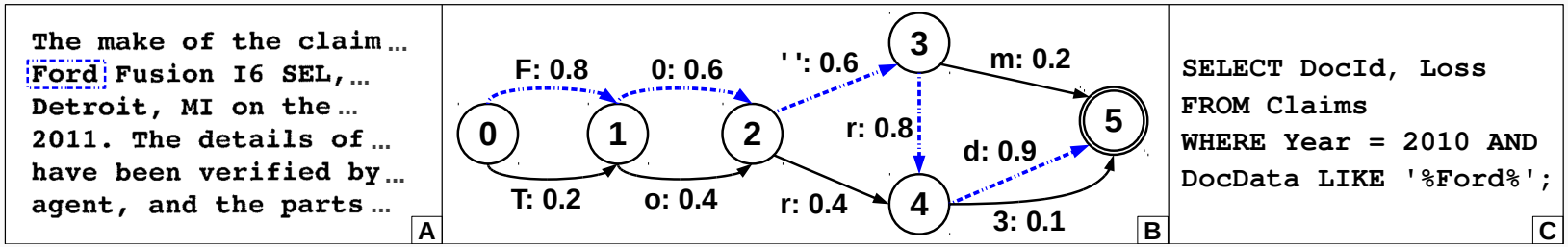
(TID)	SSN	N	M
t_1	NULL	Smith	NULL
t_2	NULL	Brown	NULL

Much of the available information cannot be represented and is lost, e.g.

- Smith's SSN is either 185 or 785; Brown's SSN is either 185 or 186.
- Data cleaning: No two distinct persons can have the same SSN.

OCR on manually-entered data

Staccato [Kumar'12]

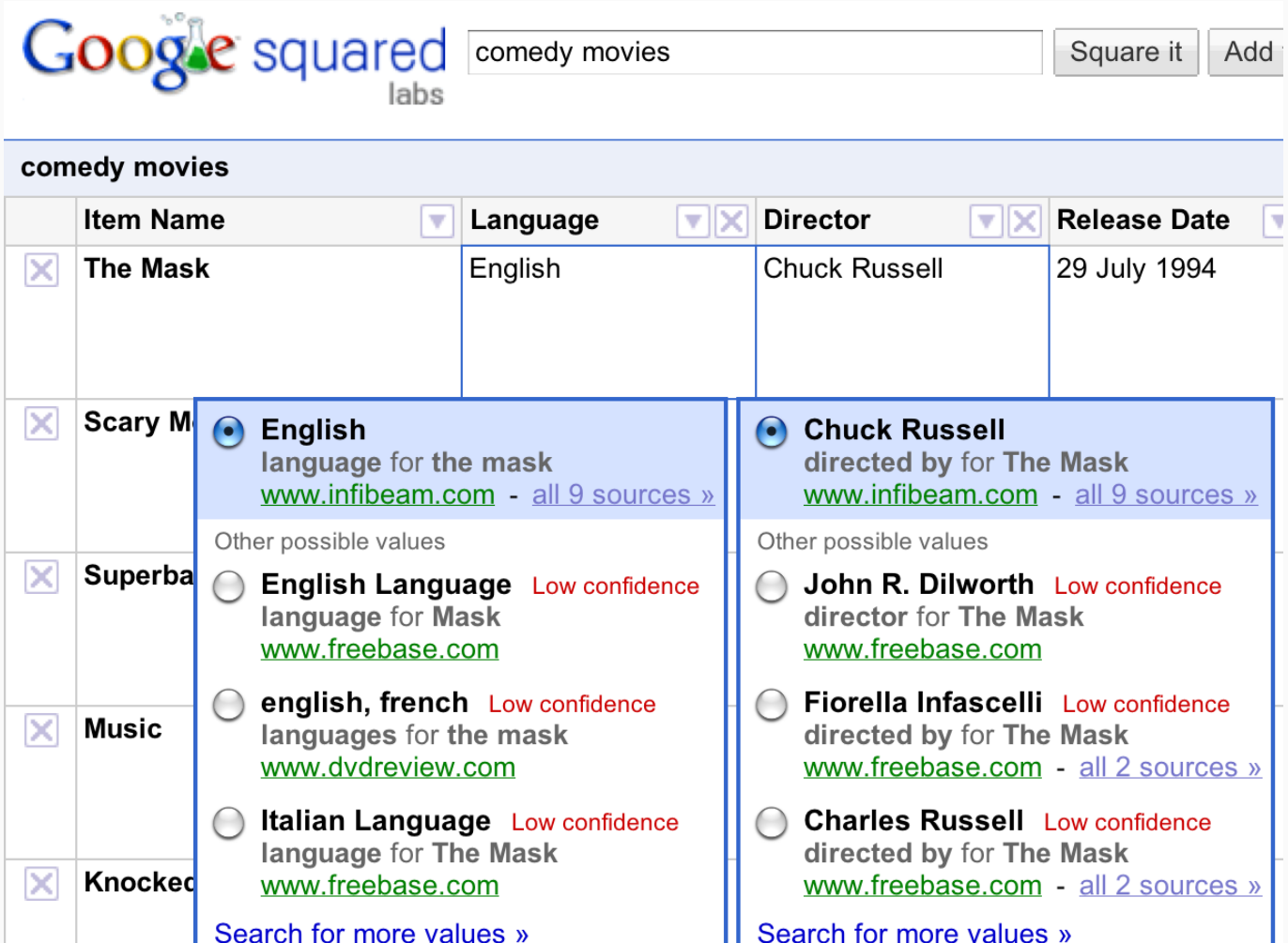


- Stochastic automaton constructed from text using Google OCRopus.
- String *F0 rd* has the highest probability (0.21).
- String *Ford* has lower probability (0.12).

Staccato accommodates several possible readings of the text to increase recall.

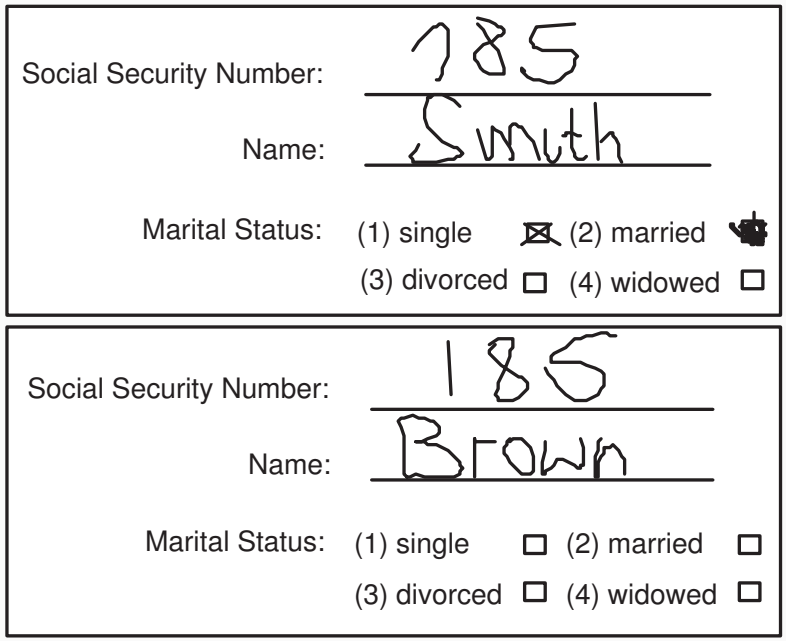
Web Data Integration with Google Squared

- Tables instead of lists of page links as answers to Google queries
- Integration of data sources with contradicting information or different schemas, degrees of trust, and degrees of completion
- Confidence values mapped to [0,1]



Talk Outline

Revisiting the Census Data Example



RID	SSN	N	M
t_1	NULL	Smith	NULL
t_2	NULL	Brown	NULL

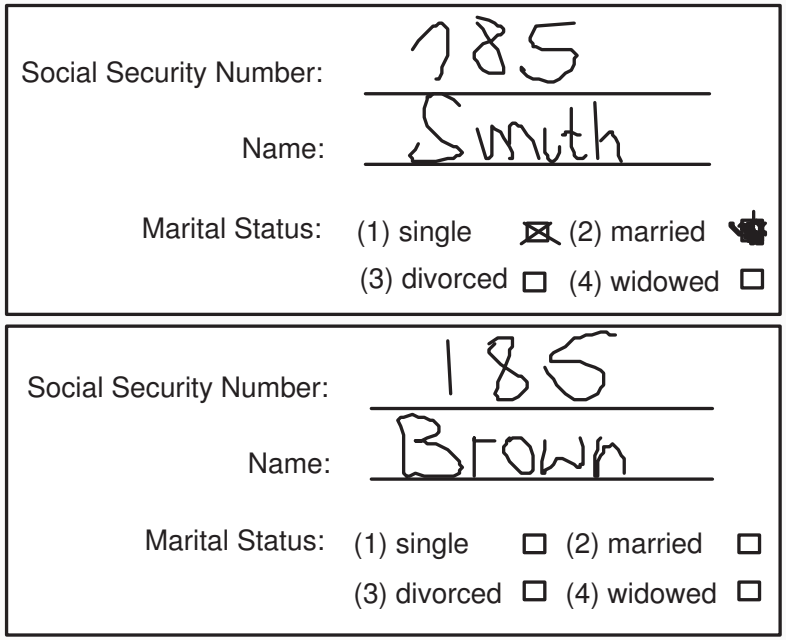
NULL values are too uninformative.

We could instead incorporate all available possibilities:

- Smith's SSN is either 185 or 785; Brown's SSN is either 185 or 186.
- Smith's M is either 1 or 2; Brown's M is either 1, 2, 3, or 4.

Revisiting the Census Data Example

There are $2 \times 2 \times 2 \times 4 = 32$ possible readings of our two census entries.



SSN	N	M
185	Smith	1
185	Brown	1

SSN	N	M
185	Smith	1
185	Brown	2

SSN	N	M
185	Smith	1
185	Brown	3

SSN	N	M
185	Smith	1
185	Brown	4

SSN	N	M
185	Smith	1
186	Brown	1

SSN	N	M
185	Smith	1
186	Brown	2

SSN	N	M
185	Smith	1
186	Brown	3

SSN	N	M
185	Smith	1
186	Brown	4

. . .

. . .

Incomplete Databases

An Incomplete Database is a finite set of database instances $\mathbf{W} = (W_1, \ldots, W_n)$.

W_1		
SSN	N	M
185	Smith	1
185	Brown	**1**

W_2		
SSN	N	M
185	Smith	1
185	Brown	**2**

Each W_i is a *possible world*.

W_3		
SSN	N	M
185	Smith	1
185	Brown	**3**

W_4		
SSN	N	M
185	Smith	1
185	Brown	**4**

W_5		
SSN	N	M
185	Smith	1
186	Brown	**1**

...

W_6		
SSN	N	M
185	Smith	1
186	Brown	**2**

...

Incomplete Databases

An Incomplete Database is a finite set of database instances $\mathbf{W} = (W_1, \ldots, W_n)$.

W_1		
SSN	N	M
185	Smith	1
185	Brown	**1**

W_2		
SSN	N	M
185	Smith	1
185	Brown	**2**

W_3		
SSN	N	M
185	Smith	1
185	Brown	**3**

W_4		
SSN	N	M
185	Smith	1
185	Brown	**4**

W_5		
SSN	N	M
185	Smith	1
186	Brown	**1**

...

W_6		
SSN	N	M
185	Smith	1
186	Brown	**2**

...

Each W_i is a *possible world*.

Typical scenario: 200M people (2/3 US census), 50 questions, 1 in 10000 ambiguous (2 options)

- 2^{10^6} possible worlds
- A world is a table with 50 columns and 200M rows!

[Antova'07]

$\rightarrow$ Key challenge: How to succinctly represent incomplete databases?

Probabilistic Databases

A Probabilistic Database is a pair $(\mathbf{W}, P)$, where $\mathbf{W}$ is an incomplete database and $P : \mathbf{W} \to [0, 1]$ is a probability distribution: $\sum_{W_i \in \mathbf{W}} P(W_i) = 1$.

$W_1 : P(W_1) = 0.1$

SSN	N	M
185	Smith	1
185	Brown	**1**

$W_2 : P(W_2) = 0.1$

SSN	N	M
185	Smith	1
185	Brown	**2**

$W_3 : P(W_3) = 0.1$

SSN	N	M
185	Smith	1
185	Brown	**3**

$W_4 : P(W_4) = 0.1$

SSN	N	M
185	Smith	1
185	Brown	**4**

$W_5 : P(W_5) = 0.3$

SSN	N	M
185	Smith	1
186	Brown	**1**

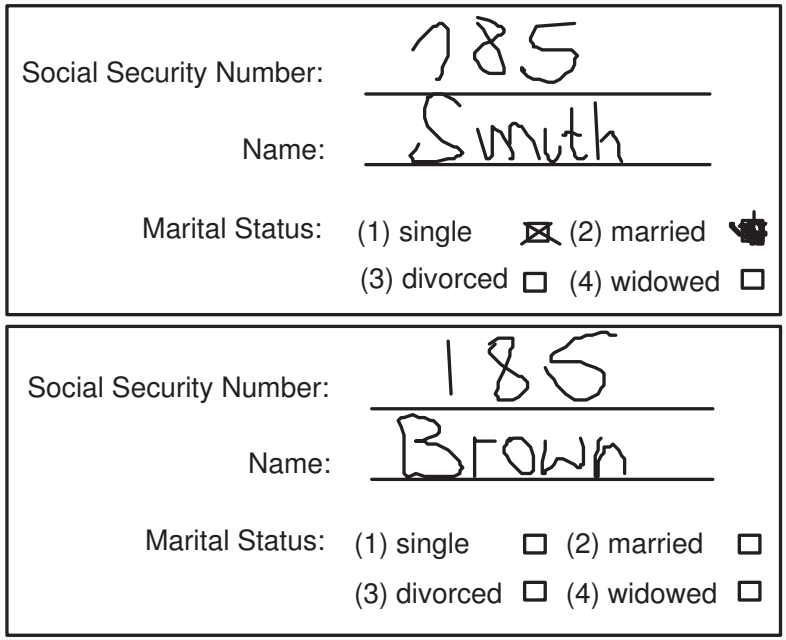
$W_6 : P(W_6) = 0.3$

SSN	N	M
185	Smith	1
186	Brown	**2**

For $\mathbf{W} = \{W_1, \ldots, W_6\}$,

$\sum_{W_i \in \mathbf{W}} P(W_i) = 1$.

Succinct Representations of Incomplete/Probabilistic Data



Succinct *or-set* representation: [Imielinski'91]

SSN	N	M
{ 185,785 }	Smith	{ 1,2 }
{ 185, 186 }	Brown	{ 1,2,3,4 }

It exploits independence of different fields:

- Choice for Smith's SSN independent of choice of for Brown's SSN.
- The probability distributions associated with these choices are independent.

BID: Alternative Representation of Our Or-Set

RID	SSN	P
t_1	185	0.7
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_1	2	0.1
t_2	1	0.25
t_2	2	0.25
t_2	3	0.25
t_2	4	0.25

BID: Alternative Representation of Our Or-Set

RID	SSN	P
t_1	185	0.7
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_1	2	0.1
t_2	1	0.25
t_2	2	0.25
t_2	3	0.25
t_2	4	0.25

Interpretation:

- The tuples within each block with the same key RID are *disjoint*
  Each world contains one tuple per block, so the tuples within a block are mutually exclusive.

BID: Alternative Representation of Our Or-Set

RID	SSN	P
t_1	185	0.7
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_1	2	0.1
t_2	1	0.25
t_2	2	0.25
t_2	3	0.25
t_2	4	0.25

Interpretation:

- The tuples within each block with the same key RID are *disjoint*
  Each world contains one tuple per block, so the tuples within a block are mutually exclusive.
- Blocks are *independent* of each other.
  The choices of tuples within different blocks are independent.
  The aggregated probability of the worlds taking the first tuple of the first block in each relation is $0.7 \times 1 \times 0.9 = 0.63$.

These *block-independent disjoint* (BID) relations are sometimes called x-relations or x-tables. Google squares are prime examples.

More on BID Databases

BIDs also allow blocks with probabilities less than 1:

RID	SSN	P
t_1	185	0.6
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

RID	N	P
t_1	Smith	0.9
t_2	Brown	1

RID	M	P
t_1	1	0.8
t_1	2	0.1
t_2	1	0.25
t_2	2	0.25
t_2	3	0.25
t_2	4	0.25

Interpretation:

- There are worlds where the first block of each of the three relations is empty, e.g., the following world:

RID	SSN	P
t_2	186	0.2

RID	N	P
t_2	Brown	1

RID	M	P
t_2	4	0.25

The probability of this world is
$0.2 \times 1 \times 0.25 \times (\mathbf{1 - 0.6 - 0.3}) \times (\mathbf{1 - 0.9}) \times (\mathbf{1 - 0.8 - 0.1}) = 5 \times 10^{-5}$.

TI: Tuple-Independent Databases

TI databases are BID databases where each block has exactly one tuple.

TI databases are the simplest and most common probabilistic data model.

RID	SSN	P
t_1	185	0.7
t_2	185	0.8

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_2	2	0.25

Interpretation:

- Each tuple t is in a random world with its probability $p(t)$.
- A relation with n tuples, whose probabilities are less than 1, has 2^n possible worlds, since each tuple may be in or out.
- Our TI example has 2^4 worlds: Any subset of the first and third relation and the entire second relation.

Are BID Databases Enough?

BIDs (and TIs) are good at capturing independence and local choice.

What about correlations across blocks?

- Enforce the key dependency on SSN in each world.
  That is: Discard the worlds where both t_1 and t_2 have SSN $= 185$.

RID	SSN	P
t_1	185	0.6
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

Are BID Databases Enough?

BIDs (and TIs) are good at capturing independence and local choice.

What about correlations across blocks?

- Enforce the key dependency on SSN in each world.
  That is: Discard the worlds where both t_1 and t_2 have SSN $= 185$.

RID	SSN	P
t_1	185	0.6
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

$\Rightarrow$

RID	SSN	Φ
t_1	185	$X = 1$
t_1	785	$X = 2$
t_2	185	$Y = 1 \wedge X \neq 1$
t_2	186	$Y = 2$

This constraint is supported by a probabilistic version of *conditional databases*.

[Imielinski'84]

Idea: Use random variables to encode correlations between tuples.

- Exclude the world where t_1 and t_2 have the same SSN 185 by using contradicting assignments for variable X.
- Transfer probabilities of tuples to probability distributions of variables.

PC: Probabilistic Conditional Databases

A *PC database* is $(\mathbf{D}, \mathbf{X}, \Phi)$, where $\mathbf{D}$ is a relational database, $\mathbf{X}$ is a set of independent random variables, and Φ is a function mapping each tuple in $\mathbf{D}$ to a propositional formula over $\mathbf{X}$.

RID	SSN	Φ
t_1	185	$X = 1$
t_1	785	$X = 2$
t_2	185	$Y = 1 \wedge X \neq 1$
t_2	186	$Y = 2$

VAR	Dom	P
X	1	0.6
X	2	0.3
Y	1	0.8
Y	2	0.2

Interpretation:

- The *world table* (right) lists the probability distribution for each independent random variable in $\mathbf{X}$.
- Each total valuation of variables in $\mathbf{X}$ defines a world whose probability is the product of probabilities of the variable assignments.
- Each tuple t is *conditional* on the satisfiability of the formula $\Phi(t)$ and is contained in those worlds defined by valuations that satisfy $\Phi(t)$.

TIs and BIDs are Special Cases of PCs

Recall our previous TI database example:

RID	SSN	P
t_1	185	0.7
t_2	185	0.8

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_2	2	0.25

Here is a PC encoding of the above TI database:

RID	SSN	Φ	P
t_1	185	s_1	0.7
t_2	185	s_2	0.8

RID	N	Φ	P
t_1	Smith	n_1	1
t_2	Brown	n_2	1

RID	M	Φ	P
t_1	1	m_1	0.9
t_2	2	m_2	0.25

Idea:

- Consider a set of Boolean random variables
- Associate each tuple in the TI database with exactly one of them
- For instance, s_1 annotates $(t_1, 185)$ and $P(s_1) = 0.7$
- World table with variable assignments may be stored explicitly

Takeaways

Various representations for probabilistic databases of increasing expressiveness.

- Most complex: probabilistic conditioned databases. [Imielinski'84]
  - Trio's ULDBs [Benjelloun'06] and MayBMS's U-relations [Antova'07].
  - Completeness: They can represent any probabilistic database.

- Mid-level: block-independent disjoint databases. [Barbará'92]
  - MystiQ, Trio, MayBMS, SPROUT^2.
  - Prime examples of BIDs: Google squares.
  - Not complete, but achieve completeness via conjunctive queries over BIDs. [Poole'93]

- Simplest: tuple-independent databases. [Zimanyi'97]
  - *The* norm in real-world repositories like Google's, DeepDive, and NELL.
  - Not complete even via unions of conjunctive queries.
  - Most theoretical work on complexity of query evaluation done for them.

Talk Outline

Why Probabilistic Databases?

Probabilistic Data Models

The Query Evaluation Problem

Query Evaluation: Complexity and Algorithms

Live Demo with MayBMS

Possible Worlds Semantics

The underlying semantics of query evaluation in probabilistic databases:

Possible worlds semantics: Given a database $\mathbf{W} = \{W_1, \ldots, W_n\}$ and a query Q, the query answer is $Q(\mathbf{W}) = \{Q(W_1), \ldots, Q(W_n)\}$.

Possible Worlds Semantics

The underlying semantics of query evaluation in probabilistic databases:

> *Possible worlds semantics*: Given a database $\mathbf{W} = \{W_1, \ldots, W_n\}$ and a query Q, the query answer is $Q(\mathbf{W}) = \{Q(W_1), \ldots, Q(W_n)\}$.

Investigations so far followed three main directions:

1. Possible and certain query answers for incomplete databases.
2. Probabilities of query answers for probabilistic databases.
3. Succinct representation of $Q(\mathbf{W})$ for query languages and data models.

Approaches 1 & 2 close the possible worlds semantics: They compute one relation with answer tuples and possibly their probabilities.

Queries on Incomplete Databases

Given query Q and incomplete database $\mathbf{W}$:

- An answer t is certain, if $\forall W_i \in \mathbf{W} : t \in Q(W_i)$
- An answer t is possible if $\exists W_i \in \mathbf{W} : t \in Q(W_i)$

W_1

SSN	N	M
185	Smith	1
185	Brown	**1**

W_2

SSN	N	M
185	Smith	1
185	Brown	**2**

W_3

SSN	N	M
185	Smith	1
185	Brown	**3**

W_4

SSN	N	M
185	Smith	1
185	Brown	**4**

W_5

SSN	N	M
185	Smith	1
186	Brown	**1**

W_6

SSN	N	M
185	Smith	1
186	Brown	**2**

Queries on Incomplete Databases

Given query Q and incomplete database $\mathbf{W}$:

- An answer t is certain, if $\forall W_i \in \mathbf{W} : t \in Q(W_i)$
- An answer t is possible if $\exists W_i \in \mathbf{W} : t \in Q(W_i)$

W_1

SSN	N	M
185	Smith	1
185	Brown	**1**

W_2

SSN	N	M
185	Smith	1
185	Brown	**2**

W_3

SSN	N	M
185	Smith	1
185	Brown	**3**

W_4

SSN	N	M
185	Smith	1
185	Brown	**4**

W_5

SSN	N	M
185	Smith	1
186	Brown	**1**

W_6

SSN	N	M
185	Smith	1
186	Brown	**2**

Let $\mathbf{W} = \{W_1, \ldots, W_6\}$.

- Query $Q_1(s) = Census(s, n, m)$ has certain answer (185) and possible answers (185) and (186).
- Query $Q_2(n) = Census(s, n, m)$ has the same possible and certain answers (Smith) and (Brown).

Queries on Incomplete Databases

Several studies on this date back to 90s for various models, in particular conditional databases. [Abiteboul'91, Grahne'91, O.'08]

Hard tasks already for positive relational algebra:

- Tuple possibility is NP-complete
- Tuple certainty is coNP-complete

We next focus on probabilistic databases.

Queries on Probabilistic Databases

Given query Q and probabilistic database $(\mathbf{W}, P)$: The Marginal Probability of an answer t is: $P(t) = \sum\{P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i)\}$.

$W_1 : P(W_1) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**1**

$W_2 : P(W_2) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**2**

$W_3 : P(W_3) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**3**

$W_4 : P(W_4) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**4**

$W_5 : P(W_5) = 0.3$		
SSN	N	M
185	Smith	1
186	Brown	**1**

$W_6 : P(W_6) = 0.3$		
SSN	N	M
185	Smith	1
186	Brown	**2**

Queries on Probabilistic Databases

Given query Q and probabilistic database $(\mathbf{W}, P)$: The Marginal Probability of an answer t is: $P(t) = \sum\{P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i)\}$.

$W_1 : P(W_1) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**1**

$W_2 : P(W_2) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**2**

$W_3 : P(W_3) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**3**

$W_4 : P(W_4) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**4**

$W_5 : P(W_5) = 0.3$		
SSN	N	M
185	Smith	1
186	Brown	**1**

$W_6 : P(W_6) = 0.3$		
SSN	N	M
185	Smith	1
186	Brown	**2**

Let $\mathbf{W} = \{W_1, \ldots, W_6\}$.

- $Q_1(s) = Census(s, n, m)$:
  $P(185) = 1$ and
  $P(186) = 0.6$.

- $Q_2(n) = Census(s, n, m)$:

$P(\text{Smith}) = P(\text{Brown}) = 1$.

Q_1 and Q_2 are trivial queries! Computing the marginal probability is hard in general!

Queries on Probabilistic Databases

Given query Q and probabilistic database $(\mathbf{W}, P)$: The Marginal Probability of an answer t is: $P(t) = \sum\{P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i)\}$.

$W_1 : P(W_1) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**1**

$W_2 : P(W_2) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**2**

$W_3 : P(W_3) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**3**

$W_4 : P(W_4) = 0.1$		
SSN	N	M
185	Smith	1
185	Brown	**4**

$W_5 : P(W_5) = 0.3$		
SSN	N	M
185	Smith	1
186	Brown	**1**

$W_6 : P(W_6) = 0.3$		
SSN	N	M
185	Smith	1
186	Brown	**2**

Let $\mathbf{W} = \{W_1, \ldots, W_6\}$.

- $Q_1(s) = Census(s, n, m)$:
  $P(185) = 1$ and
  $P(186) = 0.6$.

- $Q_2(n) = Census(s, n, m)$:

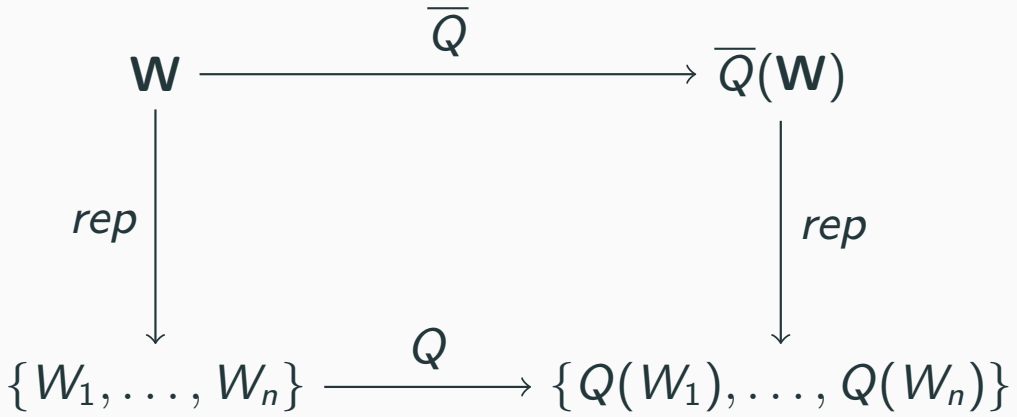
$P(\text{Smith}) = P(\text{Brown}) = 1$.

Q_1 and Q_2 are trivial queries! Computing the marginal probability is hard in general!

$\rightarrow$ Key challenge: Which queries admit efficient (polynomial time) computation of marginal probabilities for their answers?

Representability of Query Answers

For a given query language $\mathcal{Q}$ and data model $\mathcal{W}$:
For any query $Q \in \mathcal{Q}$ and database $\mathbf{W} \in \mathcal{W}$, is there $\overline{Q} \in \mathcal{Q}$ such that $\overline{Q}(\mathbf{W}) = \{Q(W_i) \mid W_i \in \mathbf{W}\}$ and can be represented in $\mathcal{W}$?



- This holds for relational algebra and PC databases: [Imielinski'84]
  $\overline{Q}(T)$ is an extension of Q to also compute the *query lineage*.
- This does not hold for BIDs and TIs, but query lineage still useful for computing marginal probabilities of query answers on BIDs and TIs.
- This idea is also used by Trio and MayBMS. [Benjelloun'06, Antova'09]

Query Lineage by Example

Customer

ckey	name	Φ
1	Joe	x_1
2	Dan	x_2

Orders

okey	ckey	date	Φ
1	1	1995-01-10	y_1
2	1	1996-01-09	y_2
3	2	1994-11-11	y_3

Lineitem

okey	disc	ckey	Φ
1	0.1	1	z_1
1	0.2	1	z_2
3	0.4	2	z_3
3	0.1	2	z_4

Query asking for the dates of discounted orders shipped to customer 'Joe':

Customer(*ckey*, Joe), Orders(*okey*, *ckey*, *date*), Lineitem(*okey*, *disc*, *ckey*)

Query answer and lineage

odate	Φ
1995-01-10	$x_1 y_1 z_1 + x_1 y_1 z_2$

$\overline{Q}$ does Q and propagates the input conditions Φ to the answers:

- join of tuples leads to conjunction of their conditions
- union/disjunction of tuples leads to disjunction of their conditions.

Query lineage traces the computation of an answer back to its input.

Marginal Probabilities via Query Lineage

The marginal probability of a query answer is the probability of its lineage.

How to compute the lineage probability?

x_1	y_1	z_1	z_2	$x_1y_1z_1 + x_1y_1z_2$	Probability
0	*	*	*	0	0
1	0	*	*	0	0
1	1	0	0	0	0
1	1	0	1	1	$P(x_1) \cdot P(y_1) \cdot (1 - P(z_1)) \cdot P(z_2)$
1	1	1	0	1	$P(x_1) \cdot P(y_1) \cdot P(z_1) \cdot (1 - P(z_2))$
1	1	1	1	1	$P(x_1) \cdot P(y_1) \cdot P(z_1) \cdot P(z_2)$

$$P(x_1y_1z_1 + x_1y_1z_2) = P(x_1) \cdot P(y_1) \cdot [1 - (1 - P(z_1)(1 - P(z_2))].$$

- The truth table is exponential in the number of variables.

Two ideas: [O.'08+]

- Read-once lineage factorization: $x_1y_1z_1 + x_1y_1z_2 = x_1y_1(z_1 + z_2)$
- Lineage compilation into polysize decision diagrams.

Where Are We Now?

- We know how to compute the query answers using a simple query extension that also computes the query lineage.
- We do not know yet how to compute the marginal probabilities of query answers efficiently.

Next, more technical part of the tutorial (not covered here):

- Analyze the complexity of computing marginal probabilities as a function of database size and query structure.

Talk Outline

Short Recap on Complexity Class #P (Sharp P)

#P = Class of functions $f(x)$ for which there exists a PTIME non-deterministic Turing machine M such that $f(x)$ = number of accepting computations of M on input x. [Valiant'79]

Class of **counting problems** associated with decision problems in NP:

- SAT (given formula ϕ, is ϕ satisfiable?) is NP-complete
- #SAT (given formula ϕ, count # of satisfying assignments) is #P-complete

A PTIME machine with a #P oracle can solve any problem in polynomial hierarchy with one #P query. [Toda'91]

#SAT is #P-complete already for bipartite positive DNFs! [Provan'83]

- .. yet SAT is trivially PTIME for DNFs.

Dichotomies for Queries on Probabilistic Databases

The following property has been observed for several classes $\mathcal{Q}$ of relational queries on TI databases:

> The data complexity of every query in $\mathcal{Q}$ is either **polynomial time** or **#P-hard**.

Dichotomies for Queries on Probabilistic Databases

The following property has been observed for several classes $\mathcal{Q}$ of relational queries on TI databases:

> The data complexity of every query in $\mathcal{Q}$ is either **polynomial time** or **#P-hard**.

Examples of such classes $\mathcal{Q}$ of relational queries:

- NCQ: non-repeating conjunctive queries [Dalvi&Suciu'04]
- Quantified queries (division, set comparisons) [Fink&O.'11]
- UCQ: unions of conjunctive queries [Dalvi&Suciu'12]
- RNCQ: ranking NCQ [O.&Wen'12]
- $1RA^-$: NCQ's relational algebra counterpart extended with negation [Fink&O.'14]

Syntactic Characterizations of Tractable Queries

The tractable queries in (R)NCQ and $1RA^{-}$ admit an efficient syntactic characterization via the *hierarchical* property.

A (Boolean) NCQ or $1RA^{-}$ query Q is *hierarchical* if:

For every pair of distinct variables A and B in Q,

there is no triple of relation symbols R, S, and T in Q such that:

- $R^{A\neg B}$ has query variable A and not B,
- S^{AB} has both query variables A and B, and
- $T^{\neg AB}$ has query variable B and not in A.

Examples

Hierarchical queries:

- $\exists_A \exists_B \left[\left(R(A) \wedge S(A, B) \right) \wedge \neg T(A, B) \right]$
- $\exists_A \exists_B \left[\left(R(A) \wedge T(B) \right) \wedge \neg \left(U(A) \wedge V(B) \right) \right]$
- $\exists_A \exists_B \left[\left(M(A) \wedge N(B) \right) \wedge \neg \left[\left(R(A) \wedge T(B) \right) \wedge \neg \left(U(A) \wedge V(B) \right) \right] \right]$

Examples

Hierarchical queries:

- $\exists_A \exists_B \left[\left(R(A) \wedge S(A,B) \right) \wedge \neg T(A,B) \right]$
- $\exists_A \exists_B \left[\left(R(A) \wedge T(B) \right) \wedge \neg \left(U(A) \wedge V(B) \right) \right]$
- $\exists_A \exists_B \Big[\left(M(A) \wedge N(B) \right) \wedge \neg \left[\left(R(A) \wedge T(B) \right) \wedge \neg \left(U(A) \wedge V(B) \right) \right] \Big]$

Non-hierarchical queries:

- $\exists_A \exists_B \left[R(A) \wedge S(A,B) \wedge T(B) \right]$
- $\exists_B \Big[\exists_A \left(R(A) \wedge S(A,B) \right) \wedge \neg T(B) \Big]$
- $\exists_B \Big[T(B) \wedge \neg \exists_A \left(R(A) \wedge S(A,B) \right) \Big]$

Hardness Proof Idea

Reduction from #P-hard model counting problem for positive 2DNF:

- Given a non-hierarchical $1\mathrm{RA}^{-}$ query Q and
- Any positive bipartite DNF formula Ψ over disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ of random variables.
- $\#\Psi$ can be computed using linearly (in most cases constantly) many calls to an oracle for $P(Q)$, where Q is evaluated on tuple-independent databases with sizes polynomial in the size of Ψ.

Simplest Example of Hardness Reduction

[Grädel'98, Dalvi& Suciu'04]

Input formula and query:

- $\Psi = x_1y_1 \vee x_1y_2 \vee x_2y_1$ over sets $\mathbf{X} = \{x_1, x_2\}, \mathbf{Y} = \{y_1, y_2\}$
- $Q = \exists_A \exists_B \Big[R(A) \wedge S(A, B) \wedge T(B) \Big]$

Construct a TI database $\mathbf{D}$ such that Ψ annotates $Q(\mathbf{D})$:

- Column Φ holds random variables in Ψ.
  - Notation: $\top$ (true)
- Variables also used as constants for A and B.

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

R		T		S			$R \wedge S \wedge T$			Q
A	Φ	B	Φ	A	B	Φ	A	B	Φ	Φ
x_1	$\mathbf{x_1}$	y_1	$\mathbf{y_1}$	x_1	y_1	$\top$	x_1	y_1	$\mathbf{x_1y_1}$	() $\mathbf{\Psi}$
x_2	$\mathbf{x_2}$	y_2	$\mathbf{y_2}$	x_1	y_2	$\top$	x_1	y_2	$\mathbf{x_1y_2}$	
				x_2	y_1	$\top$	x_2	y_1	$\mathbf{x_2y_1}$	

Simplest Example of Hardness Reduction

[Grädel'98, Dalvi& Suciu'04]

Input formula and query:

- $\Psi = x_1y_1 \vee x_1y_2 \vee x_2y_1$ over sets $\mathbf{X} = \{x_1, x_2\}, \mathbf{Y} = \{y_1, y_2\}$
- $Q = \exists_A \exists_B \Big[R(A) \wedge S(A, B) \wedge T(B) \Big]$

Construct a TI database $\mathbf{D}$ such that Ψ annotates $Q(\mathbf{D})$:

- Column Φ holds random variables in Ψ.
  - Notation: $\top$ (true)
- Variables also used as constants for A and B.

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

R		T		S			$R \wedge S \wedge T$			Q
A	Φ	B	Φ	A	B	Φ	A	B	Φ	Φ
x_1	$\mathbf{x_1}$	y_1	$\mathbf{y_1}$	x_1	y_1	$\top$	x_1	y_1	$\mathbf{x_1y_1}$	() $\mathbf{\Psi}$
x_2	$\mathbf{x_2}$	y_2	$\mathbf{y_2}$	x_1	y_2	$\top$	x_1	y_2	$\mathbf{x_1y_2}$	
				x_2	y_1	$\top$	x_2	y_1	$\mathbf{x_2y_1}$	

Query Q is the only minimal hard pattern in case of queries without negation!

A Surprising Example of Hardness Reduction

Input formula and query:

- $\Psi = x_1 y_1 \vee x_1 y_2$ over sets $\mathbf{X} = \{x_1\}, \mathbf{Y} = \{y_1, y_2\}$
- $Q = \exists_A \Big[R(A) \wedge \neg \exists_B \big(T(B) \wedge S(A, B) \big) \Big]$

Construct a TI database $\mathbf{D}$ such that Ψ annotates $Q(\mathbf{D})$:

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

R	
A	Φ
1	$\top$
2	$\top$

T	
B	Φ
x_1	$\neg \mathbf{x_1}$
y_1	$\neg \mathbf{y_1}$
y_2	$\neg \mathbf{y_2}$

S		
A	B	Φ
1	x_1	$\top$
1	y_1	$\top$
2	x_1	$\top$
2	y_2	$\top$

$T \wedge S$		
A	B	Φ
1	x_1	$\neg \mathbf{x_1}$
1	y_1	$\neg \mathbf{y_1}$
2	x_1	$\neg \mathbf{x_1}$
2	y_2	$\neg \mathbf{y_2}$

$\exists_B(T \wedge S)$	
A	Φ
1	$\neg \mathbf{x_1} \vee \neg \mathbf{y_1}$
2	$\neg \mathbf{x_1} \vee \neg \mathbf{y_2}$

$R \wedge \neg \exists_B(T \wedge S)$	
A	Φ
1	$\mathbf{x_1 y_1}$
2	$\mathbf{x_1 y_2}$

A Surprising Example of Hardness Reduction

Input formula and query:

- $\Psi = x_1 y_1 \vee x_1 y_2$ over sets $\mathbf{X} = \{x_1\}, \mathbf{Y} = \{y_1, y_2\}$
- $Q = \exists_A \Big[R(A) \wedge \neg \exists_B \big(T(B) \wedge S(A, B) \big) \Big]$

Construct a TI database $\mathbf{D}$ such that Ψ annotates $Q(\mathbf{D})$:

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

R	
A	Φ
1	$\top$
2	$\top$

T	
B	Φ
x_1	$\neg \mathbf{x_1}$
y_1	$\neg \mathbf{y_1}$
y_2	$\neg \mathbf{y_2}$

S		
A	B	Φ
1	x_1	$\top$
1	y_1	$\top$
2	x_1	$\top$
2	y_2	$\top$

$T \wedge S$		
A	B	Φ
1	x_1	$\neg \mathbf{x_1}$
1	y_1	$\neg \mathbf{y_1}$
2	x_1	$\neg \mathbf{x_1}$
2	y_2	$\neg \mathbf{y_2}$

$\exists_B(T \wedge S)$	
A	Φ
1	$\neg \mathbf{x_1} \vee \neg \mathbf{y_1}$
2	$\neg \mathbf{x_1} \vee \neg \mathbf{y_2}$

$R \wedge \neg \exists_B(T \wedge S)$	
A	Φ
1	$\mathbf{x_1 y_1}$
2	$\mathbf{x_1 y_2}$

Query Q is already hard when T is the only uncertain input relation!

Evaluation of Hierarchical 1RA$^-$ Queries

Approach based on knowledge compilation

- For any TI database $\mathbf{D}$, the probability $P_{Q(\mathbf{D})}$ of a 1RA$^-$ query Q is the probability P_Ψ of the query lineage Ψ.
- Compile Ψ into poly-size OBDD(Ψ).
- Compute probability of OBDD(Ψ) in time linear in its size.

Evaluation of Hierarchical 1RA$^-$ Queries

Approach based on knowledge compilation

- For any TI database $\mathbf{D}$, the probability $P_{Q(\mathbf{D})}$ of a 1RA$^-$ query Q is the probability P_Ψ of the query lineage Ψ.
- Compile Ψ into poly-size OBDD(Ψ).
- Compute probability of OBDD(Ψ) in time linear in its size.

Lineage of tractable 1RA$^-$ queries:

- Read-once for queries without negation (so NCQ) [O. & Huang'08]
  It admits linear-size OBBDs.

- **Not** read-once for queries with negation [Fink& O'14]
  - It admits OBBDs of size linear in the database size but exponential in the query size.

The Inner Workings

From hierarchical $1RA^-$ to RC-hierarchical $\exists$-consistent $RC^{\exists}$:

- Translate query Q into an equivalent disjunction of disjunction-free existential relational calculus queries $Q_1 \vee \cdots \vee Q_k$.

- **RC-hierarchical**:
  For each $\exists_X(Q')$, every relation symbol in Q' has variable X.
  - Each of the disjuncts gives rise to a poly-size OBDD.

- $\exists$-**consistent**:
  The nesting order of the quantifiers is the same in $Q_1, \cdots, Q_k$.
  - All OBDDs have compatible variable orders and their disjunction is a poly-size OBDD.

- The OBDD width grows exponentially with k,
  its height stays linear in the size of the database.
  - Width = maximum number of edges crossing the section between any two consecutive levels.

Similar ideas used for the evaluation of inversion-free UCQs. [Jha& Suciu'12]

Query Evaluation Example (1/3)

Consider the following query and TI database:

$$Q = \exists_A \exists_B \Big[\big(R(A) \wedge T(B)\big) \wedge \neg \big(U(A) \wedge V(B)\big) \Big]$$

R		T		U		V	
A	Φ	B	Φ	A	Φ	B	Φ
1	$\mathbf{r_1}$	1	$\mathbf{t_1}$	1	$\mathbf{u_1}$	1	$\mathbf{v_1}$
2	$\mathbf{r_2}$	2	$\mathbf{t_2}$	2	$\mathbf{u_2}$	2	$\mathbf{v_2}$

$R \wedge T$			
A	B		Φ
1	1	$\mathbf{r_1 t_1}$	
1	2	$\mathbf{r_1 t_2}$	
2	1	$\mathbf{r_2 t_1}$	
2	2	$\mathbf{r_2 t_2}$	

$R \wedge T \wedge \neg(U \wedge V)$			
A	B		Φ
1	1	$\mathbf{r_1 t_1} \neg(\mathbf{u_1 v_1})$	
1	2	$\mathbf{r_1 t_2} \neg(\mathbf{u_1 v_2})$	
2	1	$\mathbf{r_2 t_1} \neg(\mathbf{u_2 v_1})$	
2	2	$\mathbf{r_2 t_2} \neg(\mathbf{u_2 v_2})$	

Query Evaluation Example (1/3)

Consider the following query and TI database:

$$Q = \exists_A \exists_B \Big[\big(R(A) \wedge T(B)\big) \wedge \neg \big(U(A) \wedge V(B)\big) \Big]$$

R	
A	Φ
1	$\mathbf{r_1}$
2	$\mathbf{r_2}$

T	
B	Φ
1	$\mathbf{t_1}$
2	$\mathbf{t_2}$

U	
A	Φ
1	$\mathbf{u_1}$
2	$\mathbf{u_2}$

V	
B	Φ
1	$\mathbf{v_1}$
2	$\mathbf{v_2}$

$R \wedge T$		
A	B	Φ
1	1	$\mathbf{r_1 t_1}$
1	2	$\mathbf{r_1 t_2}$
2	1	$\mathbf{r_2 t_1}$
2	2	$\mathbf{r_2 t_2}$

$R \wedge T \wedge \neg(U \wedge V)$		
A	B	Φ
1	1	$\mathbf{r_1 t_1} \neg(\mathbf{u_1 v_1})$
1	2	$\mathbf{r_1 t_2} \neg(\mathbf{u_1 v_2})$
2	1	$\mathbf{r_2 t_1} \neg(\mathbf{u_2 v_1})$
2	2	$\mathbf{r_2 t_2} \neg(\mathbf{u_2 v_2})$

The lineage of Q is:

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

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

Query Evaluation Example (2/3)

Translate $Q = \exists_A \exists_B \Big[\big(R(A) \wedge T(B)\big) \wedge \neg \big(U(A) \wedge V(B)\big) \Big]$ into $\mathrm{RC}^{\exists}$:

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

- Both Q_1 and Q_2 are RC-hierarchical.
- $Q_1 \vee Q_2$ is $\exists$-consistent: Same order $\exists_A \exists_B$ for Q_1 and Q_2.

Query annotation:

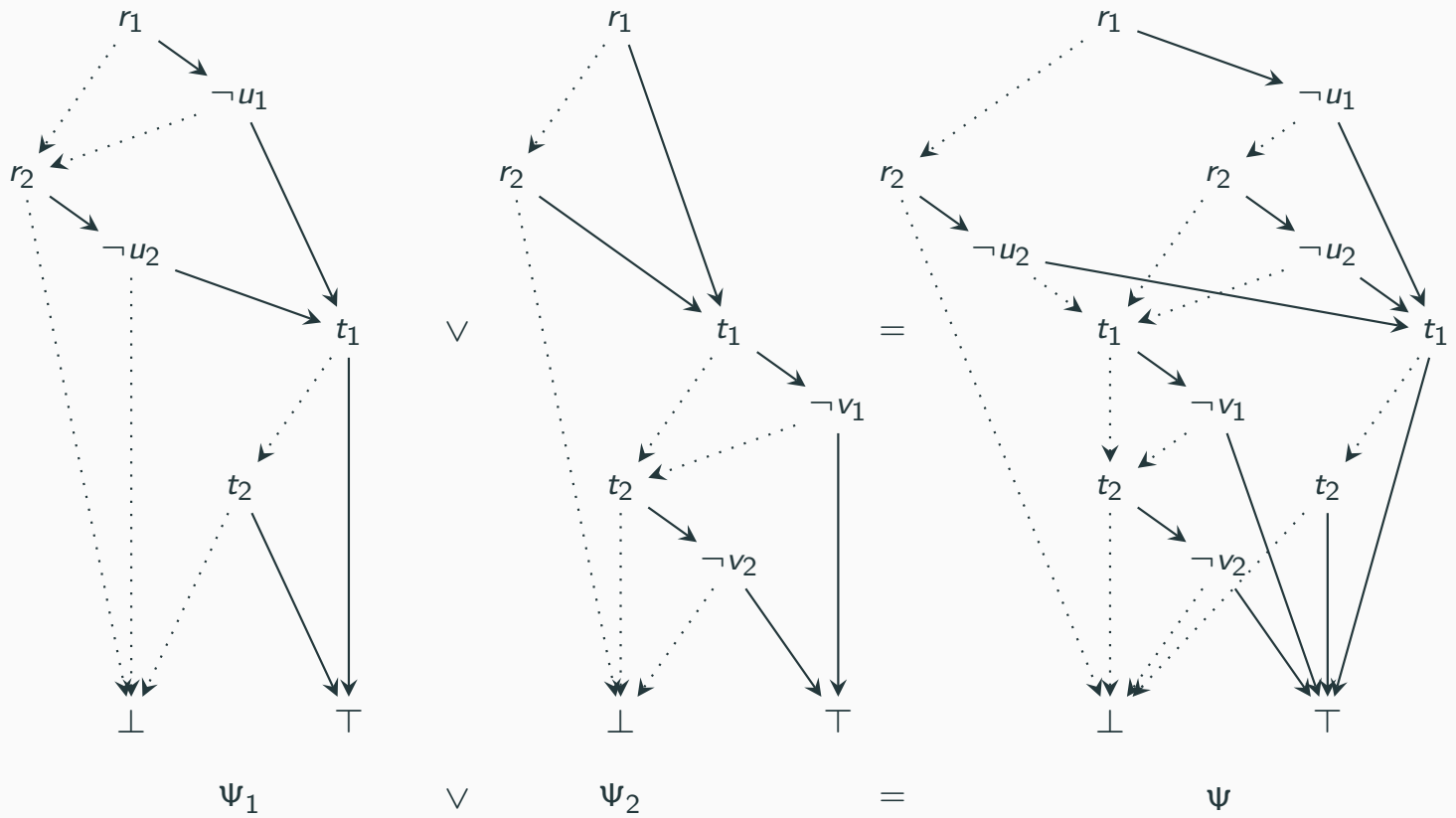
$$\Psi = \underbrace{(r_1 \neg u_1 \vee r_2 \neg u_2) \wedge (t_1 \vee t_2)}_{\Psi_1} \quad \vee \quad \underbrace{(r_1 \vee r_2) \wedge (t_1 \neg v_1 \vee t_2 \neg v_2)}_{\Psi_2}.$$

- Both Ψ_1 and Ψ_2 admit linear-size OBDDs.
- The two OBDDs have compatible orders and their disjunction is an OBDD whose width is the product of the widths of the two OBDDs.

Query Evaluation Example (3/3)

Compile query annotation into OBDD:

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



Talk Outline

Why Probabilistic Databases?

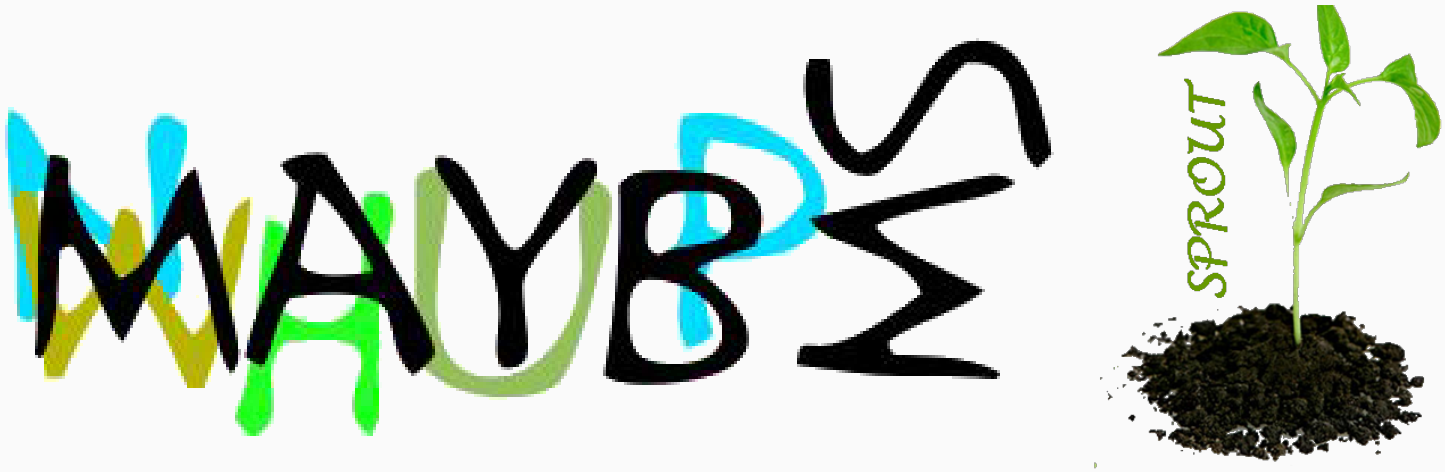
Probabilistic Data Models

The Query Evaluation Problem

Query Evaluation: Complexity and Algorithms

Live Demo with MayBMS

The MayBMS/SPROUT Probabilistic Database Management System



`http://maybms.sourceforge.net`

- Extension of open-source PostgreSQL relational DBMS
- Available manual, worked-out examples, and publications
- 3-min video demo: `http://www.cs.ox.ac.uk/dan.olteanu/maybms.mp4`

We now go over a few simple examples from the manual.

Thank you!