Uncertain Data and Probabilistic Data

MPRI 2.26.2: Web Data Management

Antoine Amarilli

TELECOM Paris

Uncertainty in the Real World

Uncertain data

Numerous sources of uncertain data:

- Measurement errors
- Data integration from contradicting sources
- Imprecise mappings between heterogeneous schemata
- Imprecise automatic process (information extraction, natural language processing, etc.)
- Imperfect human judgment
- Lies, opinions, rumors

Use case: Web information extraction

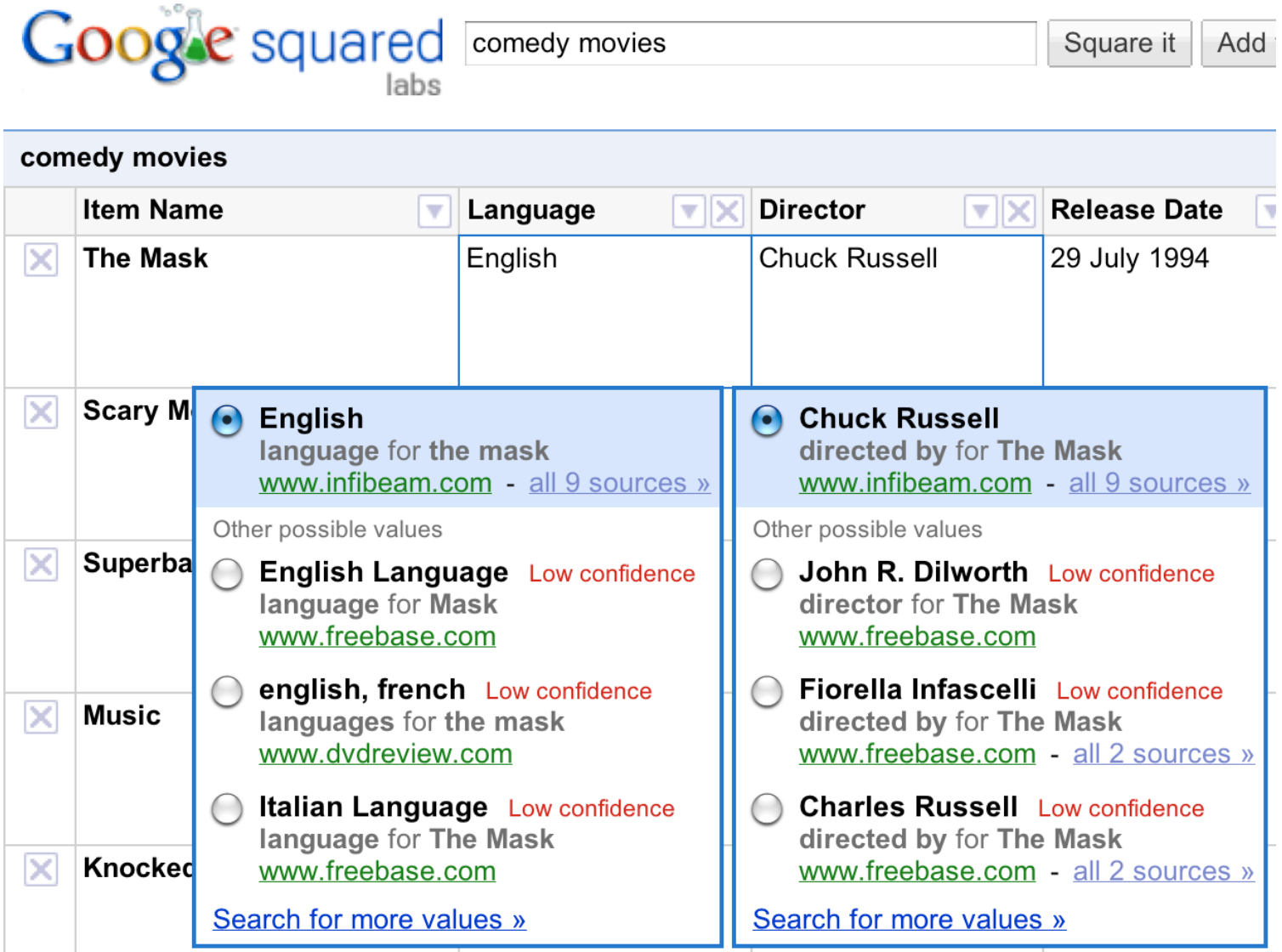
Recently-Learned Facts twitter Refresh

instance	iteration	date learned	confidence
oliguric_phase is a non-disease physiological condition	1111	06-jul-2018	97.5
alaska_airlines is an organization	1114	25-aug-2018	100.0
heating_insurance_policies is a physical action	1111	06-jul-2018	90.4
n98_12 is a term used by physicists	1111	06-jul-2018	94.2
dragonball_z__super_butoden_2 is software	1111	06-jul-2018	100.0
general_motors_corp_ is a company headquartered in the city detroit	1116	12-sep-2018	100.0
the companies herald and la compete with eachother	1111	06-jul-2018	99.6
stanford hired montgomery	1111	06-jul-2018	98.4
kimn is a radio station in the city denver	1116	12-sep-2018	100.0
radisson_sas_portman_hotel is a park in the city central_london	1116	12-sep-2018	100.0

Never-ending Language Learning (NELL, CMU),

http://rtw.ml.cmu.edu/rtw/kbbrowser/

Use case: Web information extraction



Google Squared (terminated), screenshot from [Fink et al., 2011]

Use case: Web information extraction

Subject	Predicate	Object	Confidence
Elvis Presley	diedOnDate	1977-08-16	97.91%
Elvis Presley	isMarriedTo	Priscilla Presley	97.29%
Elvis Presley	influences	Carlo Wolff	96.25%

YAGO, *https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/*

Other use case: Information extraction from scientific articles

Journal Articles

NAMURIAN (LOWER CARBONIFEROUS) GASTROPOD ASSEMBLAGES FROM NINGXIA, CHINA

PAN HUAZHANG

Nanjing Institute of Geology and Paleontology, Academia Sinica, Nanjing 210008, P. R. China

OCR

Text

… The Namurian Tsingyuan Formation from Ningxia, China, is divided into three members …

NLP

Relational Features

Entity1	Entity2	Feature
Namurian	Tsingyuan Fm.	nn
Silesian	Tsingyuan Fm.	SameRow

SQL+Python

Existing Tools

Table

Age	Formation
Silesian	Tsingyuan

Other use case: Crowdsourcing

All HITs

1-10 of 2751 Results

Sort by: HITs Available (most first) GO! Show all details | Hide all details 1 2 3 4 5 › Next » Last

Transcribe data — View a HIT in this group

Requester: p9r **HIT Expiration Date:** Nov 18, 2015 (23 hours 59 minutes) **Reward:** $0.03

Time Allotted: 45 minutes

Description: Please transcribe the data from the following images

Keywords: transcribe, handwriting, data entry

Qualifications Required:

HIT approval rate (%) is greater than 90

Classify Receipt — View a HIT in this group

Requester: Jon Brelig **HIT Expiration Date:** Nov 24, 2015 (6 days 23 hours) **Reward:** $0.02

Time Allotted: 20 minutes

Description: Looking at a receipt image, identify the business of the receipt

Keywords: image, receipt, categorize, transcribe, extract, data, entry, transcription, text, easy, qualification, jon, brelig, prod

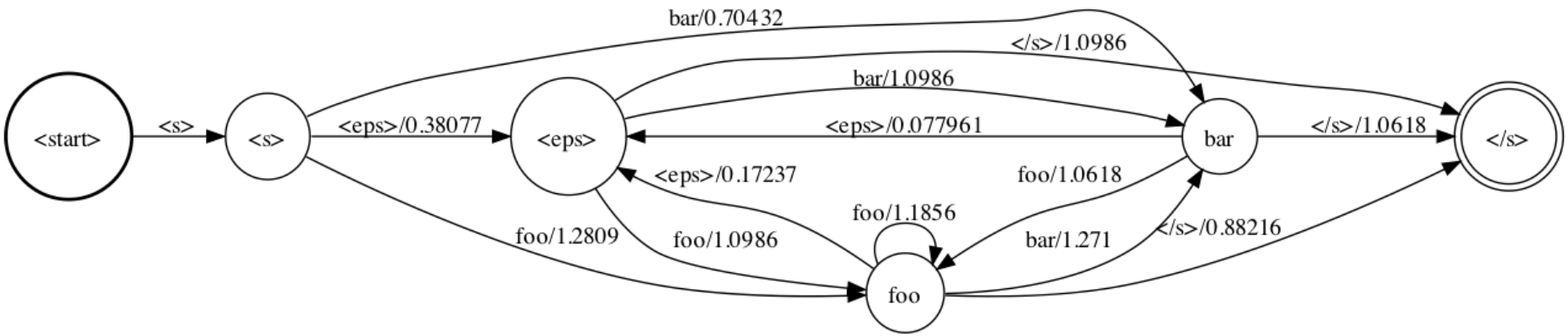
Qualifications Required:

Total approved HITs is not less than 1000

HIT approval rate (%) is not less than 97

Location is US

Other use case: Speech recognition and OCR



Uncertainty in Web information extraction

- The information extraction system is imprecise
- The system has some confidence in the information extracted, which can be:
  - a probability of the information being true (e.g., from a statistical or machine learning model)
  - an ad-hoc numeric confidence score
  - a discrete level of confidence (low, medium, high)
- What if this uncertain information is not seen as something final, but is used as a source of, e.g., a query answering system?

Different types of uncertainty

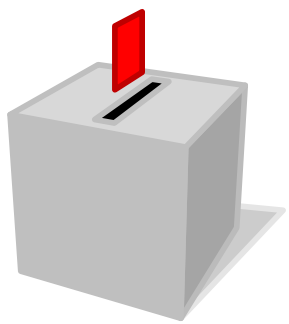
Two dimensions:

- Can be qualitative (NULL) or quantitative (95%, low-confidence, etc.) uncertainty
- Different types of uncertainty:
  - Unknown value: NULL in an RDBMS
  - Alternative between several possibilities: either A or B or C
  - Imprecision on a numeric value: a sensor gives a value that is an approximation of the actual value
  - Confidence in a fact as a whole: cf. information extraction
  - Structural uncertainty: the schema of the data itself is uncertain
  - Missing data: we know that some data is missing (open-world semantics)

Handling uncertainty

Recall the example of YAGO:

Subject	Predicate	Object	Confidence
Elvis Presley	diedOnDate	1977-08-16	97.91%
Elvis Presley	isMarriedTo	Priscilla Presley	97.29%
Elvis Presley	influences	Carlo Wolff	96.25%



What is the simplest way to manage uncertainty on this kind of data?

What happens to this uncertainty?

Naive solution

Forget about uncertainty, or apply a threshold after each computation step

What happens to this uncertainty?

Naive solution

Forget about uncertainty, or apply a threshold after each computation step

Ideal solution

Instead of neglecting uncertainty, let's manage it rigorously throughout the whole process of answering a query

How to manage uncertainty

- Represent all different forms of uncertainty
- Use probabilities to represent quantitative information on the confidence in the data
- Query data and retrieve uncertain results
- Allow adding, deleting, modifying data in an uncertain way
- Ideally: Also keep lineage/provenance information, so as to ensure traceability

Possible worlds semantics

Possible world: A regular (deterministic) relational database or XML tree

Uncertain database: (Compact) representation of a set of possible worlds

Probabilistic database: (Compact) representation of a probability distribution over possible worlds, either:

- **finite:** a set of possible worlds, each with their probability
- **continuous:** more complicated

Objective of this course

- Reminder about relational model and relational data
- Focus on probabilistic data and query evaluation on probabilistic data
  - Relational data (tuple-independent databases)
  - XML data
- Overview of nulls (missing values)
- Overview of open-world query answering (missing data)

Models of Uncertainty

- Reminder about relational data
- Probabilistic relational models
- Probabilistic XML
- Nulls and relational data
- Open-world query answering on relational data

Relational model by example

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

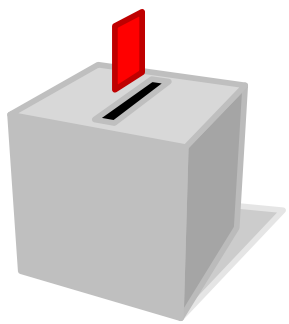
Relations and databases

Formally:

- A relation schema $\mathcal{R}$ is a finite sequence of attribute names
- A database schema $\mathcal{D}$ maps each relation name to a relation schema
- A tuple over relation schema $\mathcal{R}$ maps each attribute name of $\mathcal{R}$ to a data value
- A relation instance over $\mathcal{R}$ is a finite set of tuples over $\mathcal{R}$
- A database over database schema $\mathcal{D}$ maps each relation name R of $\mathcal{D}$ to a relation instance over the relation schema of R in $\mathcal{D}$

Positive relational algebra

Which positive relational algebra operators do you know about?



The positive relational algebra

- Algebraic language to express queries
- Each operator applies to 0, 1, or 2 subexpressions and produces a relation instance
- Main operators:
  - R: relation name
  - $\rho_{a \to b}$: rename attribute a to b
  - $\Pi_{a_1,\ldots,a_n}$: project on attributes $a_1, \ldots, a_n$
  - σ_ϕ: select all tuples satisfying condition ϕ
  - $\cup$: union of two relations (with same relation schema)
  - $\times$: cross product of two relations

Relation name

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Expression: *Guest*

Result:

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Renaming

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Expression: $\rho_{id \to guest}(Guest)$

Result:

guest	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Projection

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Expression: $\Pi_{email,id}(Guest)$

Result:

email	*id*
john.smith@gmail.com	1
alice@black.name	2
john.smith@ens.fr	3

Selection

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Expression: $\sigma_{\mathit{arrival} > 2022\text{-}01\text{-}12 \wedge \mathit{guest} = 2}(\mathit{Reservation})$

Result:

id	*guest*	*room*	*arrival*	*nights*
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

The formula used in the selection can be any Boolean combination of comparisons of attributes to attributes or constants

Cross product

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Expression: $\Pi_{id}(\mathit{Guest}) \times \Pi_{name}(\mathit{Guest})$

Result:

id	*name*
1	Alice Black
2	Alice Black
3	Alice Black
1	John Smith
2	John Smith
3	John Smith

Natural join

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Not a basic operator, but a useful shorthand!

Expression: $\mathit{Reservation} \bowtie \rho_{\mathit{id} \to \mathit{guest}}(\mathit{Guest})$

Result:

id	*guest*	*room*	*arrival*	*nights*	*name*	*email*
1	1	504	2022-01-01	5	John Smith	john.smith@gmail.com
2	2	107	2022-01-10	3	Alice Black	alice@black.name
3	3	302	2022-01-15	6	John Smith	john.smith@ens.fr
4	2	504	2022-01-15	2	Alice Black	alice@black.name
5	2	107	2022-01-30	1	Alice Black	alice@black.name

Equivalent to:

$$\Pi_{\mathit{id},\mathit{guest},\mathit{room},\mathit{arrival},\mathit{nights},\mathit{name},\mathit{email}}(\sigma_{\mathit{temp}=\mathit{guest}}(\rho_{\mathit{id} \to \mathit{temp}}(\mathit{Guest}) \times \mathit{Reservation}))$$

Union

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	*guest*	*room*	*arrival*	*nights*
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Expression: $\Pi_{room}(\sigma_{guest=2}(Reservation)) \cup \Pi_{room}(\sigma_{arrival=2022\text{-}01\text{-}15}(Reservation))$

Result:

room
107
302
504

Relational algebra vs relational calculus

Sometimes we write tuples as ground facts rather than tables

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Guest(1, John Smith, john.smith@gmail.com),
Guest(2, Alice Black, alice@black.name),
Guest(3, John Smith, john.smith@ens.fr)

Relational algebra vs relational calculus

Sometimes we write tuples as ground facts rather than tables

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Guest(1, John Smith, john.smith@gmail.com),
Guest(2, Alice Black, alice@black.name),
Guest(3, John Smith, john.smith@ens.fr)

Sometimes we write queries in relational calculus rather than algebra

$$\Pi_{id}(\mathit{Guest}) \times \Pi_{name}(\mathit{Guest})$$

$$Q(x, y') : \exists y\, z\, x'\, z'\ \ Guest(x, y, z) \wedge Guest(x', y', z')$$

Relational algebra vs relational calculus

Sometimes we write tuples as ground facts rather than tables

Guest

id	*name*	*email*
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Guest(1, John Smith, john.smith@gmail.com),
Guest(2, Alice Black, alice@black.name),
Guest(3, John Smith, john.smith@ens.fr)

Sometimes we write queries in relational calculus rather than algebra

$$\Pi_{id}(\mathit{Guest}) \times \Pi_{name}(\mathit{Guest})$$

$$Q(x, y') : \exists y\, z\, x'\, z'\ \ Guest(x, y, z) \wedge Guest(x', y', z')$$

$\rightarrow$ Relational algebra and calculus have the same expressive power!

Queries

- A query is an arbitrary function that maps databases over a fixed database schema $\mathcal{D}$ to relations over some relational schema $\mathcal{R}$
- Example of query languages:
  - Conjunctive queries (CQ), i.e., select-project-join, i.e., $\exists \cdots \bigwedge \cdots$
  - Unions of conjunctive queries (UCQ), i.e., select-project-join-union, i.e., $\bigcup \exists \cdots \bigwedge \cdots$
  - First-Order logic (FO) or the relational algebra
  - Monadic-Second Order logic (MSO)
  - etc.

Models of Uncertainty

- Reminder about relational data
- Probabilistic relational models
- Probabilistic XML
- Nulls and relational data
- Open-world query answering on relational data

Tuple-independent databases (TIDs)

[Lakshmanan et al., 1997, Dalvi and Suciu, 2007]

Patient	Examin. 1	Examin. 2	Diagnosis	Probability
A	23	12	α	0.9
B	10	23	β	0.8
C	2	4	γ	0.2
C	2	14	γ	0.4
D	15	15	α	0.6
D	15	15	β	0.4
E	15	17	β	0.7
E	15	17	α	0.3

- Allow representation of the confidence in each row of the table
- Impossible to express dependencies across rows
- Very simple model, well understood

Probabilistic c-tables [Green and Tannen, 2006]

Patient	Examin. 1	Examin. 2	Diagnosis	Condition
A	23	12	α	w_1
B	10	23	β	w_2
C	2	4	γ	w_3
C	2	14	γ	$\neg w_3 \wedge w_4$
D	15	15	β	w_5
D	15	15	α	$\neg w_5 \wedge w_6$
E	15	17	β	w_7
E	15	17	α	$\neg w_7$

- The w_i's are independent Boolean random variables
- Each w_i has a probability of being true (e.g., $\Pr(w_1) = 0.9$)
- Any finite probability distribution of tables can be represented using probabilistic c-tables

Query evaluation on probabilistic databases (PQE)

How can we evaluate a query Q over a probabilistic database?

Query evaluation on probabilistic databases (PQE)

How can we evaluate a query Q over a probabilistic database?

- Probability of a tuple for a query Q over D:

$$\Pr(t \in Q(D)) = \sum_{\substack{D' \subseteq D \\ t \in Q(D')}} \Pr(D')$$

- Intuitively: the probability of answer tuple t is the probability of drawing a possible world $D' \subseteq D$ where t is an answer

Query evaluation on probabilistic databases (PQE)

How can we evaluate a query Q over a probabilistic database?

- Probability of a tuple for a query Q over D:

$$\Pr(t \in Q(D)) = \sum_{\substack{D' \subseteq D \\ t \in Q(D')}} \Pr(D')$$

- Intuitively: the probability of answer tuple t is the probability of drawing a possible world $D' \subseteq D$ where t is an answer

Probabilistic query evaluation (PQE) problem for a query Q over tuple-independent databases: given a TID, compute the probability of each answer

Example of PQE

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value New York?

Example of PQE

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value New York?

- It is one minus the probability of not having such a tuple

Example of PQE

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value New York?

- It is one minus the probability of not having such a tuple
- Not having such a tuple is the independent AND of not having each tuple

Example of PQE

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value New York?

- It is one minus the probability of not having such a tuple
- Not having such a tuple is the independent AND of not having each tuple
- So the result is $1 - (1 - 0.5) \times (1 - 0.7) = 0.85$

Complexity of PQE

Formal question:

- We fix a Boolean relational calculus query, e.g., $\exists xy\, R(x), S(x, y), T(y)$

Complexity of PQE

Formal question:

- We fix a Boolean relational calculus query, e.g., $\exists xy\, R(x), S(x, y), T(y)$
- We are given a tuple-independent database, i.e., a relational database where facts are independent and have probabilities

Complexity of PQE

Formal question:

- We fix a Boolean relational calculus query, e.g., $\exists xy\, R(x), S(x, y), T(y)$
- We are given a tuple-independent database, i.e., a relational database where facts are independent and have probabilities
- Can we efficiently compute the probability that the query is true?

Complexity of PQE

Formal question:

- We fix a Boolean relational calculus query, e.g., $\exists xy\, R(x), S(x, y), T(y)$
- We are given a tuple-independent database, i.e., a relational database where facts are independent and have probabilities
- Can we efficiently compute the probability that the query is true?
  - This is the total probability of possible worlds where the query is true

Complexity of PQE

Formal question:

- We fix a Boolean relational calculus query, e.g., $\exists xy\, R(x), S(x, y), T(y)$
- We are given a tuple-independent database, i.e., a relational database where facts are independent and have probabilities
- Can we efficiently compute the probability that the query is true?
  - This is the total probability of possible worlds where the query is true
- Naive algorithm (exponential): consider all possible worlds and sum up their probabilities
- The naive algorithm is in fact in #P (nondeterministic PTIME counting problems)

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"

  $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"

  $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"

  $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"

  $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\, R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"

  $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\, R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_{R(a)} \left(1 - \Pr(R(a)) \left(1 - \prod_{S(a,b)}(1 - \Pr(S(a, b)))\right)\right)$

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"
  - $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\, R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_{R(a)} \left(1 - \Pr(R(a)) \left(1 - \prod_{S(a,b)}(1 - \Pr(S(a, b)))\right)\right)$
  - Make sure you understand why everything is independent in this case!

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"
  - $\rightarrow$ It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\, R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_{R(a)} \left(1 - \Pr(R(a)) \left(1 - \prod_{S(a,b)} (1 - \Pr(S(a, b)))\right)\right)$
  - Make sure you understand why everything is independent in this case!

- What is the probability of the query: $\exists xy\, R(x), S(x, y), T(y)$?

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "do we have an R-fact?"
  - $\rightarrow$ It is: $1 - \prod_{R(a)}(1 - \Pr(R(a)))$
- What is the probability of the query: $\exists xy\, R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_{R(a)} \left(1 - \Pr(R(a)) \left(1 - \prod_{S(a,b)}(1 - \Pr(S(a, b)))\right)\right)$
  - Make sure you understand why everything is independent in this case!
- What is the probability of the query: $\exists xy\, R(x), S(x, y), T(y)$?
  - It is #P-hard!

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has...

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has... 3 satisfying valuations

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has... 3 satisfying valuations
- $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has... 3 satisfying valuations
- $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

$$a'_1 \xrightarrow{1/2} a_1$$

$$a'_2 \xrightarrow{1/2} a_2$$

$$a'_3 \xrightarrow{1/2} a_3$$

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has... 3 satisfying valuations
- $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

$$a'_1 \xrightarrow{1/2} a_1 \qquad b_1 \xrightarrow{1/2} b'_1$$

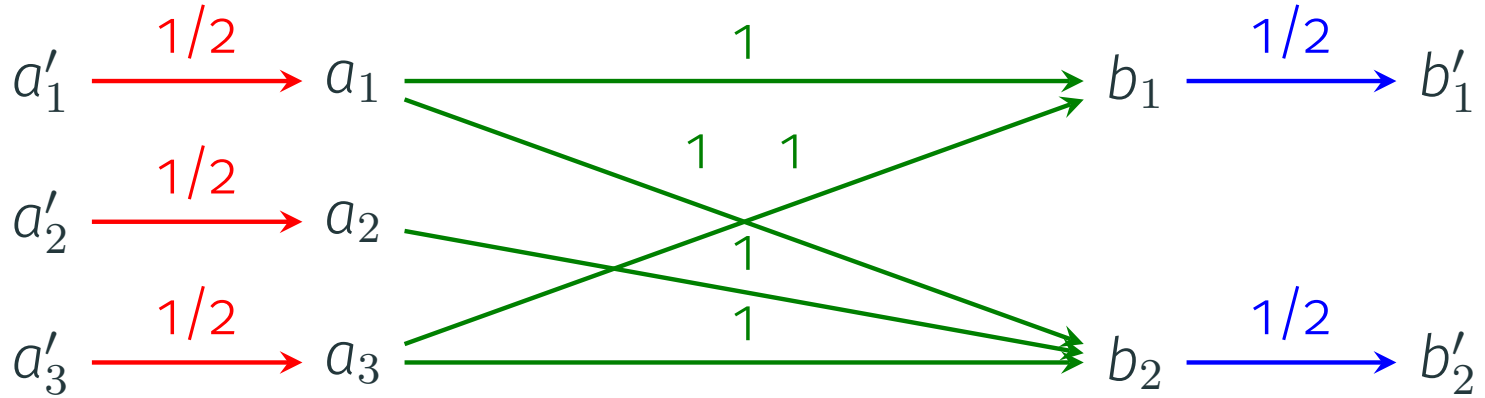
$$a'_2 \xrightarrow{1/2} a_2$$

$$a'_3 \xrightarrow{1/2} a_3 \qquad b_2 \xrightarrow{1/2} b'_2$$

An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

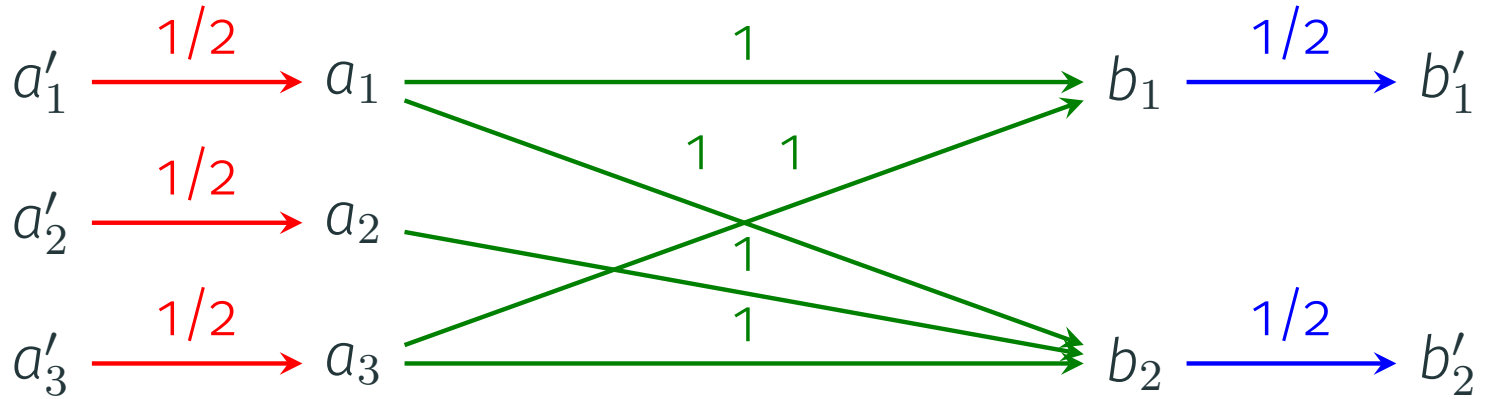
- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has... 3 satisfying valuations
- $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$



An intractable example

How to show the #P-hardness of PQE for Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from counting satisfying valuations of propositional formulas in PP2DNF:
  - Positive (no negation)
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$
- E.g., compute that $(x \wedge y) \vee (x \wedge y')$ has... 3 satisfying valuations
- $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$



Idea: Satisfying valuations of ϕ are possible worlds satisfying Q

Precise characterization of PQE

- In general, PQE is intractable (#P-hard)

Precise characterization of PQE

- In general, PQE is intractable (#P-hard)
- For CQs without self-joins:
  - Either the query is in a class (called hierarchical) and PQE is tractable
    - Algorithm: maintain probabilities while evaluating the query, everything is independent
    - Can also be explained via provenance (see later)
  - Or PQE is #P-hard for the query [Dalvi and Suciu, 2007, Olteanu and Huang, 2008]

Precise characterization of PQE

- In general, PQE is intractable (#P-hard)
- For CQs without self-joins:
  - Either the query is in a class (called hierarchical) and PQE is tractable
    - Algorithm: maintain probabilities while evaluating the query, everything is independent
    - Can also be explained via provenance (see later)
  - Or PQE is #P-hard for the query [Dalvi and Suciu, 2007, Olteanu and Huang, 2008]
- For UCQs:
  - Dichotomy between tractable (safe) and unsafe queries [Dalvi and Suciu, 2012] with more complicated criteria
  - Open problem: can this be explained via provenance? (see later)

Bleeding-edge results on PQE

- With İsmail İlkan Ceylan, for expressive queries:

Theorem ([Amarilli and Ceylan, 2020])

For any query Q closed under homomorphisms on graphs, $\mathrm{PQE}(Q)$ *is #P-hard unless Q is equivalent to a safe UCQ*

Open problems: is this also true for hypergraphs?

Bleeding-edge results on PQE

- With İsmail İlkan Ceylan, for expressive queries:

Theorem ([Amarilli and Ceylan, 2020])

For any query Q closed under homomorphisms on graphs, $\mathrm{PQE}(Q)$ is #P-hard unless Q is equivalent to a safe UCQ

Open problems: is this also true for hypergraphs?

- With Benny Kimelfeld, in the unweighted case:

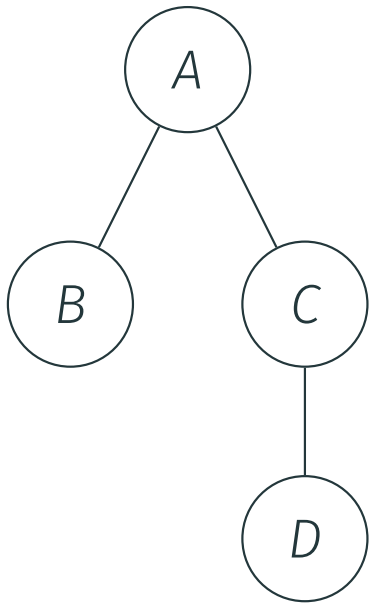
Theorem ([Amarilli and Kimelfeld, 2021])

For any CQ Q without self-joins (each relation used only once), if Q is unsafe then $\mathrm{PQE}(Q)$ is #P-hard even if all probabilities are $1/2$

Models of Uncertainty

- Reminder about relational data
- Probabilistic relational models
- Probabilistic XML
- Nulls and relational data
- Open-world query answering on relational data

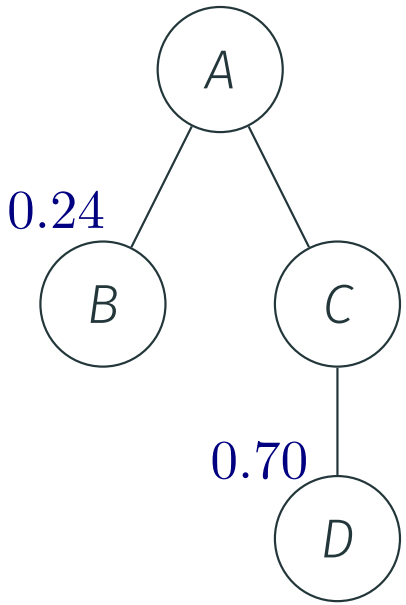
Reminder: The semistructured model and XML




```
<a>
   <b>...</b>
   <c>
      <d>...</d>
   </c>
</a>
```

- Tree-like structuring of data
- No (or less) schema constraints
- Allow mixing tags (structured data) and text (unstructured content)
- Particularly adapted to tagged or heterogeneous content

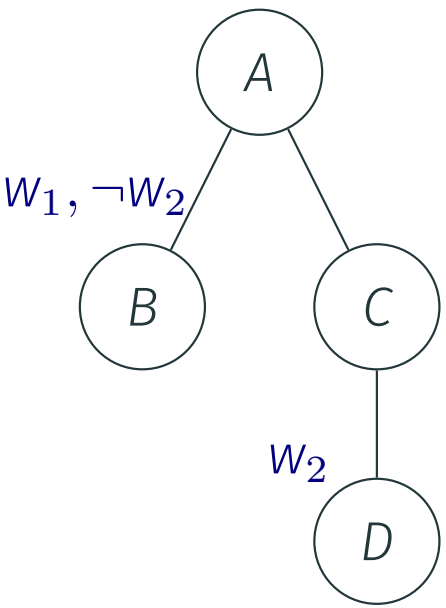
Simple probabilistic annotations



- Probabilities associated to tree nodes
- Express parent/child dependencies
- Impossible to express more complex dependencies
- $\Rightarrow$ some sets of possible worlds are not expressible this way!

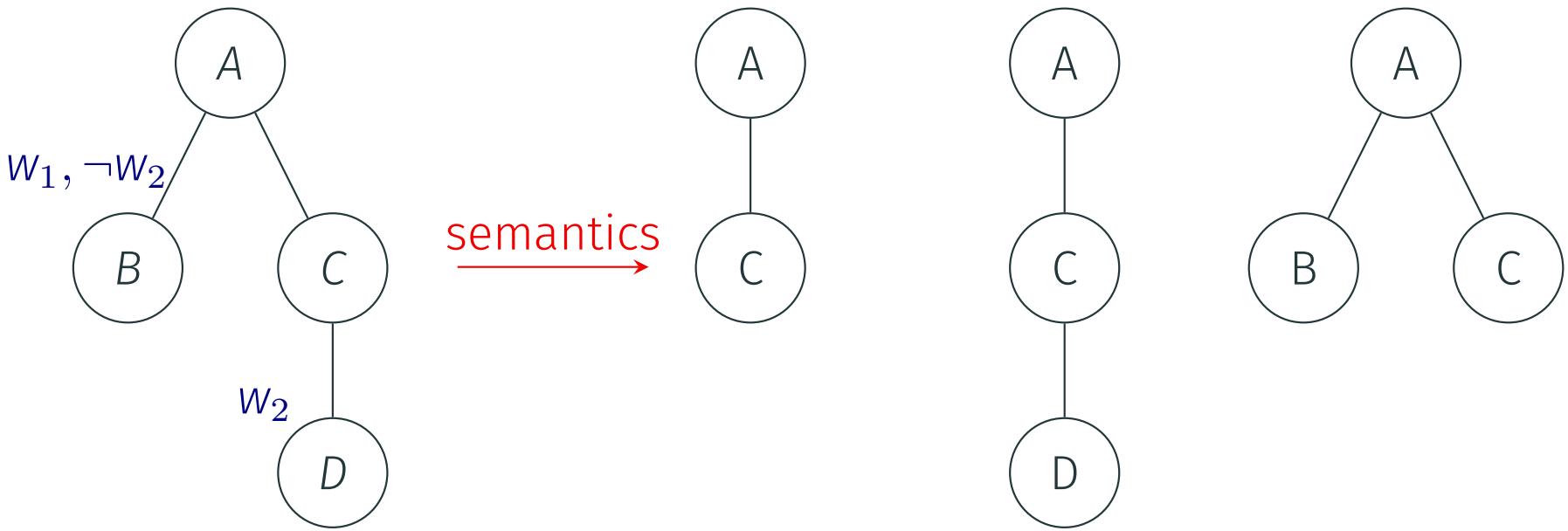
Annotations with event variables

Event	Prob.
w_1	0.8
w_2	0.7



Annotations with event variables

Event	Prob.
w_1	0.8
w_2	0.7



- Expresses arbitrarily complex dependencies
- Obviously, analogous to probabilstic c-tables

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) tree automaton accepts?

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) tree automaton accepts?
- Can be computed bottom-up in the simple model [Cohen et al., 2009]

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) tree automaton accepts?
- Can be computed bottom-up in the simple model [Cohen et al., 2009]
- #P-hard in the general model

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) tree automaton accepts?
- Can be computed bottom-up in the simple model [Cohen et al., 2009]
- #P-hard in the general model
- This generalizes to PQE for MSO on relational databases (TID) when assuming that the treewidth is bounded [Amarilli et al., 2015]

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) tree automaton accepts?
- Can be computed bottom-up in the simple model [Cohen et al., 2009]
- #P-hard in the general model
- This generalizes to PQE for MSO on relational databases (TID) when assuming that the treewidth is bounded [Amarilli et al., 2015]
- Bounding the treewidth is necessary for tractability in a certain sense [Amarilli et al., 2016]

Models of Uncertainty

- Reminder about relational data
- Probabilistic relational models
- Probabilistic XML
- Nulls and relational data
- Open-world query answering on relational data

Codd tables, a.k.a. SQL NULLs

Patient	Examin. 1	Examin. 2	Diagnosis
A	23	12	α
B	10	23	$\bot_1$
C	2	4	γ
D	15	15	$\bot_2$
E	$\bot_3$	17	β

- Most simple form of incomplete database
- Widely used in practice, in DBMS since the mid-1970s!
- All NULLs ($\bot$) are considered distinct
- Possible world semantics: all possible completions of the table (infinitely many)
- In SQL, three-valued logic, weird semantics:

```
SELECT * FROM Tel WHERE tel_nr = '333' OR tel_nr <> '333'
```

Problem: Codd tables and query evaluation

Appointment	
Doctor	Patient
D1	A
D2	A

Illness	
Patient	Diagnosis
A	$\perp$

Let's join the two tables...

Problem: Codd tables and query evaluation

Appointment

Doctor	Patient
D1	A
D2	A

Illness

Patient	Diagnosis
A	$\perp$

Let's join the two tables...

Appointment $\bowtie$ Illness

Doctor	Patient	Diagnosis

Problem: Codd tables and query evaluation

Appointment	
Doctor	Patient
D1	A
D2	A

Illness	
Patient	Diagnosis
A	$\perp$

Let's join the two tables...

Appointment $\bowtie$ Illness		
Doctor	Patient	Diagnosis
D1	A	$\perp_1$
D2	A	$\perp_2$

Problem: Codd tables and query evaluation

Appointment	
Doctor	Patient
D1	A
D2	A

Illness	
Patient	Diagnosis
A	$\perp$

Let's join the two tables...

Appointment $\bowtie$ Illness		
Doctor	Patient	Diagnosis
D1	A	$\perp_1$
D2	A	$\perp_2$

- We know that $\perp_1 = \perp_2$, but we cannot represent it
- Simple solution: named nulls aka v-tables
- More expressive solution: c-tables

C-tables [Imielinski and Lipski, 1984]

Patient	Examin. 1	Examin. 2	Diagnosis	Condition
A	23	12	α	
B	10	23	$\perp_1$	
C	2	4	γ	
D	$\perp_2$	15	$\perp_1$	
E	$\perp_3$	17	β	$18 < \perp_3 < \perp_2$

- NULLs are labeled, and can be reused inside and across tuples
- Arbitrary correlations across tuples
- Closed under the relational algebra
- Every set of possible worlds can be represented as a database with c-tables

Models of Uncertainty

- Reminder about relational data
- Probabilistic relational models
- Probabilistic XML
- Nulls and relational data
- Open-world query answering on relational data

Open-world query answering

- Most data sources are incomplete, e.g., Wikidata
- Idea: see an incomplete data source as representing all possible completions
- A query result is certain if it is true on every possible completion

Open-world query answering

- Most data sources are incomplete, e.g., Wikidata
- Idea: see an incomplete data source as representing all possible completions
- A query result is certain if it is true on every possible completion
  - → Does this seem sensible to you?

Open-world query answering

- Most data sources are incomplete, e.g., Wikidata
- Idea: see an incomplete data source as representing all possible completions
- A query result is certain if it is true on every possible completion
  - $\rightarrow$ Does this seem sensible to you?
- We also assume constraints to restrict the possible completions

Open-world query answering problem

Definition of the open-world query answering problem (OWQA):

- Given:
  - An incomplete database D
  - Logical constraints Σ on the true state of the world
  - A query Q
- Determine if Q is true in every completion of D that satisfies Q

Open-world query answering problem

Definition of the open-world query answering problem (OWQA):

- Given:
  - An incomplete database D
  - Logical constraints Σ on the true state of the world
  - A query Q
- Determine if Q is true in every completion of D that satisfies Q
- Equivalently: satisfiability of $D \wedge \Sigma \wedge \neg Q$

Open-world query answering problem

Definition of the open-world query answering problem (OWQA):

- Given:
  - An incomplete database D
  - Logical constraints Σ on the true state of the world
  - A query Q
- Determine if Q is true in every completion of D that satisfies Q
- Equivalently: satisfiability of $D \wedge \Sigma \wedge \neg Q$

Note: We assume that the incomplete database D satisfies the constraints. (Otherwise we need to repair it — a different problem.)

Results on OWQA

- The OWQA problem can be undecidable if we allow arbitrary first-order logic for Σ
- It is also undecidable for common database constraint languages, e.g., tuple-generating dependencies
- It is decidable for better-behaved logical fragments, e.g., the guarded fragment

Results on OWQA

- The OWQA problem can be undecidable if we allow arbitrary first-order logic for Σ
- It is also undecidable for common database constraint languages, e.g., tuple-generating dependencies
- It is decidable for better-behaved logical fragments, e.g., the guarded fragment
- Two main techniques:
  - Forward chaining, aka the “chase”: add data to satisfy the constraints:

Results on OWQA

- The OWQA problem can be undecidable if we allow arbitrary first-order logic for Σ
- It is also undecidable for common database constraint languages, e.g., tuple-generating dependencies
- It is decidable for better-behaved logical fragments, e.g., the guarded fragment
- Two main techniques:
  - Forward chaining, aka the "chase": add data to satisfy the constraints:
    - If the process terminates, use the result to satisfy the query

Results on OWQA

- The OWQA problem can be undecidable if we allow arbitrary first-order logic for Σ
- It is also undecidable for common database constraint languages, e.g., tuple-generating dependencies
- It is decidable for better-behaved logical fragments, e.g., the guarded fragment
- Two main techniques:
  - Forward chaining, aka the "chase": add data to satisfy the constraints:
    - If the process terminates, use the result to satisfy the query
    - If it is infinite but has bounded treewidth, reason over it, e.g., with automata

Results on OWQA

- The OWQA problem can be undecidable if we allow arbitrary first-order logic for Σ
- It is also undecidable for common database constraint languages, e.g., tuple-generating dependencies
- It is decidable for better-behaved logical fragments, e.g., the guarded fragment
- Two main techniques:
  - Forward chaining, aka the "chase": add data to satisfy the constraints:
    - If the process terminates, use the result to satisfy the query
    - If it is infinite but has bounded treewidth, reason over it, e.g., with automata
  - Backward chaining, aka "query rewriting": change the query to reflect the constraints

References i

Antoine Amarilli and İsmail İlkan Ceylan. A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs. In *ICDT*, 2020.

Antoine Amarilli and Benny Kimelfeld. Uniform Reliability of Self-Join-Free Conjunctive Queries. In *ICDT*, 2021.

Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Provenance Circuits for Trees and Treelike Instances. In *ICALP*, 2015. doi: 10.1007/978-3-662-47666-6_5.

Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Tractable Lineages on Treelike Instances: Limits and Extensions. In *PODS*, 2016. doi: 10.1145/2902251.2902301.

References ii

Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Running tree automata on probabilistic xml. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 227–236, 2009.

Nilesh Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6), 2012.

Nilesh Dalvi, Chrisopher Ré, and Dan Suciu. Probabilistic databases: Diamonds in the dirt. *Communications of the ACM*, 52(7), 2009.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB Journal*, 16(4), 2007.

References iii

Robert Fink, Andrew Hogue, Dan Olteanu, and Swaroop Rath. SPROUT^2: a squared query engine for uncertain web data. In *SIGMOD*, 2011.

Todd J. Green and Val Tannen. Models for incomplete and probabilistic information. In *Proc. EDBT Workshops, IIDB*, Munich, Germany, March 2006.

Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

Laks V. S. Lakshmanan, Nicola Leone, Robert B. Ross, and V. S. Subrahmanian. ProbView: A flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3), 1997.

Dan Olteanu and Jiewen Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *Proc. SUM*, 2008.

References iv

Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. CIDR*, Asilomar, CA, USA, January 2005.

Credits

Adapted from class material by Pierre Senellart