Soutenance d’évaluation à mi-parcours

Uncertainty over Structured and Intensional Data

Antoine Amarilli

Télécom ParisTech; Institut Mines–Télécom; CNRS LTCI

December 4th, 2014
Background

- Lots of raw information on the Web
- Leverage it to answer complex queries
  - Extract structure
  - Integrate various sources
  - Manage possible errors

→ Where can I get a pizza?
→ Find an affordable flat near Télécom with $\geq 20 \text{ m}^2$?
Intensionality

- We cannot collect all information:
  - Storage space
  - Bandwidth
  - Access restrictions

- Need to access remote data sparingly
- Choose relevant accesses dynamically

- Web crawling
- Web APIs
- Crowdsourcing
- Deep Web
- Expensive processing
- Rule consequences
Structure

- Need to leverage existing structure
- Structure can be heterogeneous
  - Avoid focusing only on one framework

- XML/JSON
- Views
- Web graph
- RDF triples
- Relational DBs
- Parse trees
Uncertainty

- Data is imprecise
- Data is wrong
- Processing induces uncertainty
- Represent priors on remote data

→ Fuzzy rules → Crowdsourcing → Data integration
→ NLP → Annotations → Information extraction
Use cases

- Extracting **structured facts** from an **open** set of news sources
  - Start with an initial **knowledge** about the world
  - Locate promising **articles**
  - Run expensive **processing** on the articles
  - **Uncertainty** when accessing, disambiguating
  - Use **crowdsourcing** to validate the facts
  - Using **logical rules** to constrain them
Our vision of a general approach

- Unsuccessfully submitted to VLDB 2014 [Amarilli and Senellart, 2014a]
- Submitted as a tutorial proposal to ICDT 2015 [Amarilli and Senellart, 2014b]
- Reviews due in 8 days

UnSAID: Uncertainty and Structure in the Access to Intensional Data
Antoine Amarilli
Institut Mines-Télécom; Télécom ParisTech; CNRS LTCI
Paris, France
firstname.lastname@telecom-paristech.fr

ABSTRACT
In this vision paper, we first briefly describe the high-level challenges of intensional data management in the context of the extensional view. Then, we introduce our proposal, called UnSAID (UnCertainty and Structure in the Access to Intensional Data), which allows users to access complex data sets in a way that is as transparent as possible. We also discuss the advantages and disadvantages of the approach, and provide an overview of the current state of the art in the field.

1. INTRODUCTION

Intensional Data Management. Many data-centric applications involve data that is not directly available in extension. In the past, such applications were limited to the use of a few specific data sources, such as the Web or the database. However, today, with the advent of the Web of Data, it is possible to access a wide range of data sources, including social media, sensors, and other sources of information. This has opened up new possibilities for data integration and analysis, but it also poses new challenges, such as the need to handle uncertainty and the need to manage the complexity of the data.

As a first example of the approach, consider the application of mobility in smart cities, i.e., a system integrating information about transportation options, travel habits, traffic, etc., and around a city. All resources mentioned in the previous paragraph can be used to collect and enrich data related to this application: the Web, deep Web sources, social networking sites, the Semantic Web, and wrapper induction systems, crowdsourcing platforms, etc. Moreover, in such a setting, domain-specific resources, not necessarily public, contribute to the available data: street cameras, red light sensors, air pollution monitoring systems, etc.

Users of the system, namely, transport engineers, ordinary citizens, etc., may have many kinds of knowledge acquisition needs. They can be simple queries expressed in a classical query language (e.g., “How many cars went through this road during that day?”), certain patterns to mine from the data (“Find an association rule of the form $X \Rightarrow Y$ that holds among people commuting to this district”), or higher-level business intelligence queries (“Find anything interesting about the use of the local bike rental system in the past week.”).

As a second example, consider the problem of personal information management, namely, integrating user data across services that manage the user’s emails, calendar, social network, travel information, etc. To answer a knowledge acquisition need such as “find the people I need to warn about my upcoming trips”, the system would have to orchestrate queries to the various services: extract the trips, identify the meetings that conflict with them, and determine their likely participants.

As a third example, consider socially-driven Web archived [26]: their goal is to build semantically annotated Web archives on specific topics or events (investment for growth in the 2014 Winter Olympics, etc.) We can use this system to enrich data from the social Web as to which documents are relevant. These archives can then be semantically connected by means of, for instance, topic or background databases [28]. In the same way, in intensional data management, we study how to perform query optimization and other data management tasks when only the schema (and access methods) to some of the data is directly available, not the facts.

Intensional data management applications share a number of distinguishing features. At every point in time, one has an uncertain view of the world, that includes all the data that has already been accessed, together with the schema, access methods, and some prior about what data remain to be accessed. Given a user’s query, the central question in intensional data management is: “What is the best thing to do next?”

Once an access is chosen and performed, some data is retrieved, and the knowledge of the world is revised. This tutorial is an introduction to intensional data management, with a review of the solutions brought in various areas of data management and machine learning, and of some challenging open problems.

Use Case. To illustrate, let us give some concrete examples of complex use cases involving intensional data management.

Consider the application of mobility in smart cities, i.e., a system integrating information about transportation options, travel habits, traffic, etc., and around a city. Various public resources can be used to collect and enrich data related to this application: the Web, deep Web sources, social networking sites, the Semantic Web, and wrapper induction systems, crowdsourcing platforms, etc. Moreover, in such a setting, domain-specific resources, not necessarily public, contribute to the available data: street cameras, red light sensors, air pollution monitoring systems, etc. Users of the system, namely, transport engineers, ordinary citizens, etc., may have many kinds of knowledge acquisition needs. They can be simple queries expressed in a classical query language (e.g., “How many cars went through this road during that day?”), certain patterns to mine from the data (“Find an association rule of the form $X \Rightarrow Y$ that holds among people commuting to this district”), or higher-level business intelligence queries (“Find anything interesting about the use of the local bike rental system in the past week.”).

As a second example, consider the problem of personal information management, namely, integrating user data across services that manage the user’s emails, calendar, social network, travel information, etc. To answer a knowledge acquisition need such as “find the people I need to warn about my upcoming trips”, the system would have to orchestrate queries to the various services: extract the trips, identify the meetings that conflict with them, and determine their likely participants.

As a third example, consider socially-driven Web archived [26]: their goal is to build semantically annotated Web archives on specific topics or events (investment for growth in the 2014 Winter Olympics, etc.) We can use this system to enrich data from the social Web as to which documents are relevant. These archives can then be semantically connected by means of, for instance, topic or background databases [28]. In the same way, in intensional data management, we study how to perform query optimization and other data management tasks when only the schema (and access methods) to some of the data is directly available, not the facts.

Intensional data management applications share a number of distinguishing features. At every point in time, one has an uncertain view of the world, that includes all the data that has already been accessed, together with the schema, access methods, and some prior about what data remain to be accessed. Given a user’s query, the central question in intensional data management is: “What is the best thing to do next?”

Once an access is chosen and performed, some data is retrieved, and the knowledge of the world is revised. This tutorial is an introduction to intensional data management, with a review of the solutions brought in various areas of data management and machine learning, and of some challenging open problems.
Down to Earth

- Mere query evaluation on probabilistic data: \#P-hard
- Interaction of rules and probabilistic data poorly understood
- No good notions of reasoning with probabilistic rules
- Query answering with rules often undecidable
- Conditioning probabilistic data wildly intractable
Down to Earth

- Mere query evaluation on probabilistic data: \#P-hard
- Interaction of rules and probabilistic data poorly understood
- No good notions of reasoning with probabilistic rules
- Query answering with rules often undecidable
- Conditioning probabilistic data wildly intractable

→ Let us focus on more manageable problems!
Table of contents

1. Research Topic
2. Tractability for Treelike Probabilistic Data
3. Open-World Query Answering
4. Crowd Data Mining
5. Other Topics
6. Conclusion
General presentation

- Joint work with Pierre Bourhis (CNRS Lille) and Pierre Senellart (my advisor)
- Restrict probabilistic instances and correlations to be treelike
- Show tractability of query evaluation on them
Background: Instances and queries

- Given a **relational instance with probabilities**:

<table>
<thead>
<tr>
<th>Paper</th>
<th>Conference</th>
<th>Proba</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PODS</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>ICDT</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>PODS</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>ICDT</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Given a **conjunctive query** (CQ) (existentially quantified)

\[ q : \exists p_1 p_2 c \text{ Accepted}(p_1, c) \land \text{Accepted}(p_2, c) \land p_1 \neq p_2 \]
Background: Instances and queries

- Given a relational instance with probabilities:

<table>
<thead>
<tr>
<th>Paper</th>
<th>Conference</th>
<th>Proba</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PODS</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>ICDT</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>PODS</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>ICDT</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Given a conjunctive query (CQ) (existentially quantified)
  \[ q : \exists p_1 p_2 c \text{ Accepted}(p_1, c) \land \text{Accepted}(p_2, c) \land p_1 \neq p_2 \]

  → Query evaluation: probability that \( q \) holds?
Background: Instances and queries

- Given a relational instance with probabilities:

<table>
<thead>
<tr>
<th>Paper</th>
<th>Conference</th>
<th>Proba</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PODS</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>ICDT</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>PODS</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>ICDT</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Given a conjunctive query (CQ) (existentially quantified)
  \[ q : \exists p_1 p_2 c \text{ Accepted}(p_1, c) \land \text{Accepted}(p_2, c) \land p_1 \neq p_2 \]
  → Query evaluation: probability that \( q \) holds?
  → Data complexity: \( q \) is fixed
Background: Instances and queries

Given a relational instance with probabilities:

<table>
<thead>
<tr>
<th>Paper</th>
<th>Conference</th>
<th>Proba</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PODS</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>ICDT</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>PODS</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>ICDT</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Given a conjunctive query (CQ) (existentially quantified)

$$q : \exists p_1 p_2 c \text{ Accepted}(p_1, c) \land \text{Accepted}(p_2, c) \land p_1 \neq p_2$$

→ Query evaluation: probability that $q$ holds?

→ Data complexity: $q$ is fixed

→ Assume independent events (for now)
Hardness and tractability

→ Query evaluation is \#P-hard on arbitrary instances! :-(

Existing work:

- Show dichotomy between \#P-hard and PTIME queries

Our approach:

- Impose a restriction on the instance and correlations
- Show that many queries are tractable in this case
Hardness and tractability

→ Query evaluation is \( \#P\)-hard on arbitrary instances! :-(

- Existing work:
  → Show dichotomy between \( \#P\)-hard and PTIME queries
Hardness and tractability

→ Query evaluation is \( \#P \)-hard on arbitrary instances! :-(

- Existing work:
  → Show dichotomy between \( \#P \)-hard and PTIME queries

- Our approach:
  → Impose a restriction on the instance and correlations
  → Show that many queries are tractable in this case
Bounded treewidth

An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...
Bounded treewidth

An idea from instances without probabilities...

- If an instance has \textit{low treewidth} then it is almost a tree
- Assume that the instance treewidth is \textit{constant}...

\textit{instance } I
\[ R(a, b) \ R(b, c) \ S(c) \]
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[
\text{instance } I \xrightarrow{\text{tree encoding } T_I} \text{tree decomposition}
\]

\[
R(a, b) \; R(b, c) \; S(c)
\]

\[
O(|I|) \text{ for fixed width}
\]
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[ \text{instance } I \xrightarrow{\text{tree encoding}} T_I \]

\[ R(a, b) \ R(b, c) \ S(c) \]

\[ \text{tree decomposition} \]

\[ O(|I|) \text{ for fixed width} \]

\[ \exists xy \ R(x, y) \land S(y) \]

query \( q \)
Bounded treewidth

An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

instance $I$ $\rightarrow$ tree encoding $T_I$

\[
R(a, b) \ R(b, c)\ \ S(c)
\]

tree decomposition

O($|I|$) for fixed width

rewriting

O(1) data complexity

$\exists xy \ R(x, y) \land S(y)$

query $q$ $\rightarrow$ deterministic

tree automaton $A_q$
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[
\text{instance } I \xrightarrow{R(a, b) \ R(b, c) \ S(c)} \text{tree encoding } T_I
\]

- \(O(|I|)\) for fixed width
- rewriting \(O(1)\) data complexity

\[
\exists xy \ R(x, y) \land S(y) \quad \text{query } q \xrightarrow{\text{tree automaton } A_q}\]

- \(\exists xy \ R(x, y) \land S(y)\) tree automaton \(A_q\)
- **evaluation** linear time
- **query answer**
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[
\begin{align*}
\text{instance } I & \quad \text{tree encoding } T_I \\
R(a, b) & \quad R(b, c) & \quad S(c)
\end{align*}
\]

- tree decomposition
- \(O(|I|)\) for fixed width
- rewriting
- \(O(1)\) data complexity
- \(\exists xy \ R(x, y) \land S(y)\)
- query \(q\)
- deterministic tree automaton \(A_q\)
- evaluation
- linear time
- query answer
Bounded treewidth

An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

instance $I$ \(\rightarrow\) tree encoding $T_I$

- tree decomposition
- $O(|I|)$ for fixed width

rewriting

$O(1)$ data complexity

query $q$ \(\rightarrow\) deterministic tree automaton $A_q$

evaluation

linear time

query answer

$\rightarrow$ Linear time data complexity
Our idea

- Consider tree-like instances
- Represent probabilistic events with a circuit
- Compute a joint tree decomposition of them
- Compile the query to a tree automaton on encodings
- Instrument an automaton run on the uncertain instance
- Use existing message-passing inference on the result
Our idea

- Consider tree-like instances
- Represent probabilistic events with a circuit
- Compute a joint tree decomposition of them
- Compile the query to a tree automaton on encodings
- Instrument an automaton run on the uncertain instance
- Use existing message-passing inference on the result
  → Compute query probability in linear time
    (assuming fixed-cost arithmetics)
Main result in pictures

instance $I$

$s_1 = 1/2$ $s_2 = 1/2$ $s_3 = 1/2$

$R(a, b)$ $R(b, c)$ $R(c, d)$
Main result in pictures

instance $I$

$\land$

$\land$

$\land$

1/2 1/2 1/2

$R(a, b)$

$R(b, c)$

$R(c, d)$

tree encoding $T_i$

$\land$

$\land$

$\land$

1/2

$R(a, b)$

$R(b, c)$

$R(c, d)$

tree decomposition

$O(|I|)$ for fixed width
Main result in pictures

instance $I$

\[ \exists xy \, R(x, y) \land S(y) \]

query $q$
Main result in pictures

instance $I$

\[
\begin{align*}
&1/2 \quad 1/2 \quad 1/2 \\
&\land \\
&\land \\
&\land
\end{align*}
\]

\[R(a, b)\]

\[R(b, c)\]

\[R(c, d)\]

\(\land\) tree decomposition

\(O(|I|)\) for fixed width

\[
\begin{align*}
&1/2 \\
&\land \\
&\land \\
&\land
\end{align*}
\]

\[R(a, b)\]

\[R(b, c)\]

\[R(c, d)\]

\(\land\) tree encoding

\(T_I\)

rewriting

\(O(1)\) data complexity

\[\exists xy \ R(x, y) \land S(y)\]

query \(q\)

deterministic

\(\land\) tree automaton \(A_q\)
instance $I$  
\begin{align*}
1/2 & 1/2 1/2 \\
\land & \land \\
R(a, b) & R(b, c) \\
\land & R(c, d)
\end{align*}

\text{tree decomposition} \quad O(|I|) \text{ for fixed width}

\text{tree encoding} \quad T_I

\text{rewriting} \quad O(1) \text{ data complexity}

\exists xy \ R(x, y) \land S(y)

\text{query} \ q

\text{deterministic} \quad \text{tree automaton} \ A_q

\text{bounded treewidth}\n
\text{circuit} \ C

\text{instrumentation} \quad \text{linear time}
Main result in pictures

instance $I$

$\exists xy \ R(x, y) \land S(y)$

query $q$

rewriting

$O(1)$ data complexity

$O(|I|)$ for fixed width

tree decomposition

tree encoding $T_I$

bounded treewidth circuit $C$

instrumentation linear time

probabilistic inference $O(|C|)$ for fixed width

probability $p$
Main result in pictures

instance $I$

$\exists y \, R(x, y) \land S(y)$

query $q$

rewriting

$O(1)$ data complexity

$T_I$

tree encoding

$O(|I|)$ for fixed width

$deterministic$

tree automaton $A_q$

bounded treewidth
circuit $C$

instrumentation

linear time

probabilistic inference

$O(|C|)$ for fixed width

$0.42$

probability $p$
Specific consequences

- For queries representable as deterministic automata ...
  - CQs
  - Monadic second-order
  - Guarded second-order
Specific consequences

- For queries \textit{representable as deterministic automata} ...
  - \textit{CQs}
  - Monadic second-order
  - Guarded second-order

- ... on various \textit{probabilistic models} ...
  - Tuple-independent tables (presented before)
  - Block-independent disjoint tables
  - \textit{pc}-tables
  - Probabilistic XML
Specific consequences

- For queries **representable as deterministic automata** ...
  - CQs
  - Monadic second-order
  - Guarded second-order

- ... on various **probabilistic models** ...
  - Tuple-independent tables (presented before)
  - Block-independent disjoint tables
  - pc-tables
  - Probabilistic XML

- ... assuming **bounded treewidth** (for reasonable definitions) ...
Specific consequences

- For queries **representable as deterministic automata** ...
  - \( \rightarrow \) CQs
  - \( \rightarrow \) Monadic second-order
  - \( \rightarrow \) Guarded second-order

- ... on various **probabilistic models** ...
  - \( \rightarrow \) Tuple-independent tables (presented before)
  - \( \rightarrow \) Block-independent disjoint tables
  - \( \rightarrow \) pc-tables
  - \( \rightarrow \) Probabilistic XML

- ... assuming **bounded treewidth** (for reasonable definitions) ...
  - \( \rightarrow \) ... probability of fixed \( q \) can be computed in \( O(l) \)!
Specific consequences

- For queries representable as deterministic automata ...
  - CQs
  - Monadic second-order
  - Guarded second-order

- ... on various probabilistic models ...
  - Tuple-independent tables (presented before)
  - Block-independent disjoint tables
  - pc-tables
  - Probabilistic XML

- ... assuming bounded treewidth (for reasonable definitions) ...
  - ... probability of fixed $q$ can be computed in $O(l)!$

Also: link with semiring provenance
Conference submission

- Preliminary presentation at the AMW School 2014
- Informal presentation at Highlights 2014
- Submitted to PODS 2015 [Amarilli et al., 2014c]
- Reviews due in 15 days

Probabilities and Provenance via Tree Decompositions

Antoine Amarilli
Institut Mines–Télécom
Télécom ParisTech
CNRS LTCI
Paris, France

Pierre Bourhis
CNRS LIFL
Université Lille 1
INRIA Lille
Lille, France

Pierre Senellart
Institut Mines–Télécom
Télécom ParisTech; CNRS LTCI
& NUS; CNRS IPAL
Paris, France & Singapore

ABSTRACT

Query evaluation is hard on probabilistic databases, even on very simple probabilistic data frameworks and fairly simple queries, except for limited classes of safe queries. We study the problem from a different angle: rather than restricting the queries, at which conditions on the data can we tractably evaluate expressive queries on probabilistic instances? More specifically, we restrict the data tree-width, which we define on a circuit-based generalization of c-tables, in a natural way that restricts both the underlying instance and the annotations. We then leverage known tree-automata constructions to evaluate queries on bounded-tree-width instances, for such logical fragments as monadic second-order logic or frontier-guarded Datalog. We prove that we can compute in linear time a bounded-treewidth lineage circuit for automaton runs on tree decompositions of bounded-tree-width instances, so that the probability of the query can then be evaluated in linear-time data complexity (assuming unit-
Possible extensions

- **Practical implementation**: connect to [Maniu et al., 2014]
- Connect to *rule mining* on ontologies [Galárraga et al., 2013]
- Extend to *probabilistic rules* (original focus)
- MPRI internship proposal
# Table of contents

1. Research Topic
2. Tractability for Treelike Probabilistic Data
3. Open-World Query Answering
4. Crowd Data Mining
5. Other Topics
6. Conclusion
General presentation

- Joint work with Michael Benedikt (University of Oxford)
- Impose logical rules on databases
- Reason on the certain consequences of an instance
- Show decidability of the problem for rule languages
Background

- Database instance $I$ which is correct but incomplete
- Query $q$: is it certain that $q$ holds on completions of $I$?
- Restrict to completions satisfying some constraints $\Sigma$

→ Is $q$ a logical consequence of $I$ and $\Sigma$?
Background

- Database instance $I$ which is correct but incomplete
- Query $q$: is it certain that $q$ holds on completions of $I$?
- Restrict to completions satisfying some constraints $\Sigma$
  $\Rightarrow$ Is $q$ a logical consequence of $I$ and $\Sigma$?
- Constraints:
  - **Unary inclusion dependencies (UID)**
    Example: $\forall xy \, \text{Reviews}(x, y) \Rightarrow \exists z \, \text{Reviews}(y, z)$
  - **Functional dependencies (FD)**
    Example: $\forall xyz \, \text{Reviews}(x, z) \land \text{Reviews}(y, z) \Rightarrow x = y$
Finite vs unrestricted query answering

- **Unrestricted QA:**
  \[ l, \Sigma \models q \text{ if } J \models q \text{ for all } J \supseteq l \text{ s.t. } J \models \Sigma \]

- **Finite QA:**
  \[ l, \Sigma \models q \text{ if } J \models q \text{ for all finite } J \supseteq l \text{ s.t. } J \models \Sigma \]

They do not always coincide!
Finite vs unrestricted query answering

- **Unrestricted QA:**
  \[ I, \Sigma \models q \text{ if } J \models q \text{ for all } J \supseteq I \text{ s.t. } J \models \Sigma \]

- **Finite QA:**
  \[ I, \Sigma \models q \text{ if } J \models q \text{ for all finite } J \supseteq I \text{ s.t. } J \models \Sigma \]

- They do not always coincide!

  **Instance:** List of employees
  **Constraint 1:** Each employee reviews some employee (UID)
  **Constraint 2:** At most one reviewer per employee (FD)
  **Query:** Are all employees reviewed?
Finite vs unrestricted query answering

- **Unrestricted QA:**
  \[ I, \Sigma \models q \text{ if } J \models q \text{ for all } J \supseteq I \text{ s.t. } J \models \Sigma \]

- **Finite QA:**
  \[ I, \Sigma \models q \text{ if } J \models q \text{ for all finite } J \supseteq I \text{ s.t. } J \models \Sigma \]

They do not always coincide!

**Instance:** List of employees
- **Constraint 1:** Each employee reviews some employee (UID)
- **Constraint 2:** At most one reviewer per employee (FD)
- **Query:** Are all employees reviewed?
  → If they coincide, we say we are finitely controllable (FC)
Implication

- The constraints $\Sigma$ entail constraint $\tau$: every instance satisfying $\Sigma$ also satisfies $\tau$
- Again, finite or unrestricted
- For general inclusion dependencies and FDs: undecidable [Mitchell, 1983]
- Fortunately, PTIME for UIDs and FDs
  $\rightarrow$ Possible reason why not FC: not closed under implication
  $\rightarrow$ Is this the only reason?
Our result

- This is the **only** reason why UIDs/FDs are not FC

→ UIDs/FDs are *finitely controllable* modulo finite closure
Our result

This is the **only** reason why UIDs/FDs are not FC

→ **UIDs/FDs are finitely controllable** modulo finite closure

Why is it interesting?

- UIDs and FDs are **common** database constraints
- These problems are often **undecidable**
- Existing techniques were **limited**:
  - To **infinite** QA (separability)
  - To cases with **no FDs** [Barany et al., 2010]
  - To **restricted cases** with forced FC [Rosati, 2006]
  - To **arity-two** signatures
    [Pratt-Hartmann, 2009, Ibáñez-García et al., 2014]
Our result

- This is the **only** reason why UIDs/FDs are not FC

→ UIDs/FDs are **finitely controllable** modulo finite closure

- Why is it interesting?
  - UIDs and FDs are **common** database constraints
  - These problems are often **undecidable**
  - Existing techniques were **limited**:
    - To infinite QA (separability)
    - To cases with **no FDs** [Barany et al., 2010]
    - To **restricted cases** with forced FC [Rosati, 2006]
    - To **arity-two** signatures
      [Pratt-Hartmann, 2009, Ibáñez-García et al., 2014]

- Other result: decidable unrestricted QA for GC² and frontier-one acyclic dependencies
Unsuccessfully submitted to PODS 2014 [Amarilli, 2014a]

Presented at Dahu working group at ENS Cachan, 2014

Presented at Dagstuhl seminar “Querying and Reasoning under Expressive Constraints”
Open-World Query Answering Under Number Restrictions

Antoine Amarilli
Institut Mines–Télécom
Télécom ParisTech; CNRS LTCI
Paris, France
antoine.amarilli@telecom-paristech.fr

ABSTRACT

Open-world query answering (QA) is the problem of deciding, given a database instance, a set of constraints and a query, whether the query holds over all possible completions of the instance satisfying the constraints. It is used to reason over incomplete information and find out if a query is entailed by constraints given non-exhaustive data. Though QA is in general undecidable under expressive constraint languages, decidable cases are known: the guarded fragment, which cannot express number restrictions such as functional dependencies, or the guarded fragment with number restrictions but on a signature of arity two. In this paper, we combine both settings by showing the decidability of QA with number restrictions for arbitrary signatures, with expressive constraints on the binary part of the signature and less expressive constraints overall. Turning to QA over finite completions of the instance, we show in decidability under unary inclusion dependencies and functional dependencies, by establishing finite controllability up to a finite closure operation. This provides, to our knowledge, the first decidability result for QA has also been studied in the context of classical database theory, first as a query containment problem [23] and then in its full right [10, 4]. In this setting, the signature is arbitrary and number restrictions, such as the well-known functional dependencies (FDs), often make QA undecidable [33]; the decidable fragments [10, 8, 6] usually limit the interaction between number restrictions and the other constraints.

Contribution 1. Our first main contribution (Theorem 5.5) is to prove that we can get the best of both worlds, namely decidable QA on arbitrary arity signatures for a fragment including both GC^2 constraints on arity-two predicates, arbitrary FDs, and frontier-one dependencies [3] exporting only one variable. We prove this result through an unwrapping argument inspired by [24], to show that we can force models to be acyclic and respect FDs, obtaining as a by-product the tree model property for this fragment. We then present the refinement reduction to the arity-two case [25], rewriting our fragment to GC^2 constraints and proving decidability. In comparison with extensions of description logics to higher arity [12], we support arbitrary FDs and expressive GC^2 constraints.

Contributions 2 and 3.

Finite Open-World Query Answering with Number Restrictions

Antoine Amarilli
Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI
Email: antoine.amarilli@telecom-paristech.fr

Abstract—Open-world query answering is the problem of deciding, given a database instance set of constraints and query, whether the query holds over all possible completions of the instance satisfying the constraints. There are two variations, depending on whether the completions considered are finite (denoted here as FQA) or are unrestricted in cardinality (UQA). Open-world query answering is used to reason over incomplete information and find out if a query is entailed by constraints given non-exhaustive data. The major known decidable cases of UQA and FQA derive from the following: the guarded fragment of first-order logic, which can express referential constraints (data in one place points to data in another) but not number restrictions such as functional dependencies; and the guarded fragment with number restrictions but on a signature of arity only two. In this paper, we give the first decidability results for FQA that combine both referential constraints and number restrictions for arbitrary signatures. Our results rely on new techniques for constructing finite models respecting number restrictions and referential constraints.

[TODO: no hold in preprint] [TODO: resatle these in appendix] 1. Introduction

A longstanding goal in computational logic is to get logical that, in fact, coincide. These results have been generalized by Bárány et al. [2] to a much richer class of constraints, the guarded fragment of first-order logic.

A second class of constraints that has long been known to be decidable for many problems of interest are functional dependencies (FDs) – constraints of the form \( \forall x_1 \ldots x_n (R(x_1 \ldots x_n) \land R(x_1 \ldots x_n) / x_k \rightarrow y) \). Indeed, the implication problem (does one FD follow from a set of others) is decidable, and coincides with implication restricted to finite instances. Trivially FQA and UQA are decidable as well, and co-inside.

This paper considers to what extent these classes, FDs and IDs, can be combined while retaining decidable FQA. It is well-known that for arbitrary FDs and IDs, both unrestricted and finite query answering are undecidable [4]. Unrestricted query answering is known to be decidable when the FDs and the IDs are “non-conflicting” [12], [4]. We will formally define this later, but it is a condition that is sufficient to guarantee that the FDs can be ignored, as long as they hold on the initial instance \( I \), and one can then solve the query answering problem by considering the IDs alone. The non-conflicting condition is
<table>
<thead>
<tr>
<th></th>
<th>Research Topic</th>
<th>2 Tractability for Treelike Probabilistic Data</th>
<th>3 Open-World Query Answering</th>
<th>4 Crowd Data Mining</th>
<th>5 Other Topics</th>
<th>6 Conclusion</th>
</tr>
</thead>
</table>
General presentation

- Joint work with Yael Amsterdamer and Tova Milo (Tel Aviv University) and Pierre Senellart
- Crowd sourcing: asking queries to human users
- Crowd data sourcing: extract data from humans in this way
- Crowd data mining: perform data mining tasks on the crowd
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{
\{\text{beer, diapers}\},
\{\text{beer, bread, butter}\},
\{\text{beer, bread, diapers}\},
\{\text{salad, tomato}\}
\} \]

- Itemset is frequent if it occurs in \( \geq \Theta = 50\% \) of transactions
Frequent itemset mining

**Data mining** – discovering interesting patterns in large databases

**Database** – a (multi)set of transactions

**Transaction** – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{\text{beer, diapers}\}, \{\text{beer, bread, butter}\}, \{\text{beer, bread, diapers}\}, \{\text{salad, tomato}\} \} \]

- Itemset is frequent if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{salad\} not frequent
Frequent itemset mining

**Data mining** – discovering interesting patterns in large databases

**Database** – a (multi)set of **transactions**

**Transaction** – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

\[ D = \{
\{ \text{beer, diapers} \},
\{ \text{beer, bread, butter} \},
\{ \text{beer, bread, diapers} \},
\{ \text{salad, tomato} \}
\} \]

- Itemset is **frequent** if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{salad\} not frequent
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{\text{beer, diapers}\}, \{\text{beer, bread, butter}\}, \{\text{beer, bread, diapers}\}, \{\text{salad, tomato}\} \} \]

- Itemset is frequent if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{salad\} not frequent
Frequent itemset mining

**Data mining** – discovering interesting patterns in large databases

**Database** – a (multi)set of **transactions**

**Transaction** – a set of **items** (aka. an **itemset**)

A simple kind of pattern to identify are **frequent itemsets**

\[
D = \{
\{\text{beer, diapers}\},
\{\text{beer, bread, butter}\},
\{\text{beer, bread, diapers}\},
\{\text{salad, tomato}\}
\}
\]

- Itemset is **frequent** if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{salad\} not frequent
- \{beer, diapers\} frequent
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{\text{beer, diapers}\}, \{\text{beer, bread, butter}\}, \{\text{beer, bread, diapers}\}, \{\text{salad, tomato}\} \} \]

- Itemset is frequent if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{salad\} not frequent
- \{beer, diapers\} frequent
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{ \text{beer, diapers} \}, \{ \text{beer, bread, butter} \}, \{ \text{beer, bread, diapers} \}, \{ \text{salad, tomato} \} \} \]

- Itemset is frequent if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{\text{salad}\} not frequent
- \{\text{beer, diapers}\} frequent
  \[ \Rightarrow \{\text{beer}\} \text{ is also frequent} \]
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{\text{beer, diapers}\}, \{\text{beer, bread, butter}\}, \{\text{beer, bread, diapers}\}, \{\text{salad, tomato}\} \} \]

- Itemset is frequent if it occurs in \[ \geq \theta = 50\% \] of transactions
- \{\text{salad}\} not frequent
- \{\text{beer, diapers}\} frequent
  \[ \Rightarrow \{\text{beer}\} \text{ is also frequent} \]
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{ \text{beer, diapers} \}, \{ \text{beer, bread, butter} \}, \{ \text{beer, bread, diapers} \}, \{ \text{salad, tomato} \} \} \]

- Itemset is frequent if it occurs in 
  \[ \geq \Theta = 50\% \] of transactions
- \{salad\} not frequent
- \{beer, diapers\} frequent
  \[ \Rightarrow \{\text{beer}\} \text{ is also frequent} \]
Frequent itemset mining

Data mining – discovering interesting patterns in large databases

Database – a (multi)set of transactions

Transaction – a set of items (aka. an itemset)

A simple kind of pattern to identify are frequent itemsets

\[ D = \{ \{\text{beer, diapers}\}, \{\text{beer, bread, butter}\}, \{\text{beer, bread, diapers}\}, \{\text{salad, tomato}\} \} \]

- Itemset is frequent if it occurs in \( \geq \Theta = 50\% \) of transactions
- \{salad\} not frequent
- \{beer, diapers\} frequent
- \( \Rightarrow \) \{beer\} is also frequent

→ We also assume we have a known taxonomy on the items
Human knowledge mining

- Some databases only exist in the minds of people
- Example: popular activities in Athens:
  - $t_1$: I went to the acropolis and to the museum.
    - $\Rightarrow \{\text{acropolis}, \text{museum}\}$
  - $t_2$: I visited Piraeus and had some ice cream.
    - $\Rightarrow \{\text{piraeus}, \text{icecream}\}$
  - $t_3$: On Monday I attended the keynote and had coffee.
    - $\Rightarrow \{\text{keynote}, \text{coffee}\}$
Human knowledge mining

- Some databases only exist in the minds of people
- Example: popular activities in Athens:
  - $t_1$: I went to the acropolis and to the museum.
    $$\Rightarrow \{\text{acropolis, museum}\}$$
  - $t_2$: I visited Piraeus and had some ice cream.
    $$\Rightarrow \{\text{piraeus, icecream}\}$$
  - $t_3$: On Monday I attended the keynote and had coffee.
    $$\Rightarrow \{\text{keynote, coffee}\}$$

- We want frequent itemsets: frequent activity combinations
  $$\Rightarrow$$ How to retrieve this data from people?
Harvesting the data

- We cannot collect such data in a centralized database:
  1. It's impractical to ask all users to surrender their data
     “Everyone please tell us all you did the last three months.”
  2. People do not remember the information
     “What were you doing on August 23th, 2013?”
Harvesting the data

- We cannot collect such data in a centralized database:
  1. It's impractical to ask all users to surrender their data
     “Everyone please tell us all you did the last three months.”
  2. People do not remember the information
     “What were you doing on August 23th, 2013?”

- People remember summaries that we could access
  “Do you often eat ice cream when attending a keynote?”

⇒ We can just ask people if an itemset is frequent
Crowdsourcing – solving hard problems through elementary queries to a crowd of users

- **Find out if an itemset is frequent** with the crowd:
  - **1. Draw** a sample of users from the crowd. *(black box)*
  - **2. Ask:** is this itemset frequent? *("Do you often have coffee?")*
  - **3. Corroborate** the answers to eliminate bad answers. *(black box)*
  - **4. Reward** the users. *(e.g., monetary incentive)*
Crowdsourcing

- **Crowdsourcing** – solving hard problems through elementary queries to a crowd of users

- Find out if an itemset is frequent with the crowd:
  1. **Draw** a sample of users from the crowd. *(black box)*
  2. **Ask**: is this itemset frequent? *(“Do you often have coffee?”)*
  3. **Corroborate** the answers to eliminate bad answers. *(black box)*
  4. **Reward** the users. *(e.g., monetary incentive)*

⇒ The crowd is an **oracle**: given an itemset, say if it is frequent
The problem

We can now describe the problem:

- We have:
  - A known item domain $\mathcal{I}$ (set of items)
  - A known taxonomy $\Psi$ on $\mathcal{I}$ (is-a relation, partial order)
  - A crowd oracle to decide if an itemset is frequent or not

- Choose questions interactively based on past answers

$\Rightarrow$ Find out the status of all itemsets
The problem

We can now describe the problem:

- We have:
  - A known item domain $\mathcal{I}$ (set of items)
  - A known taxonomy $\Psi$ on $\mathcal{I}$ (is-a relation, partial order)
  - A crowd oracle to decide if an itemset is frequent or not
- Choose questions interactively based on past answers
- Find out the status of all itemsets

What is a good algorithm to solve this problem?
The problem

We can now describe the **problem**:

- We have:
  - A known **item domain** $\mathcal{I}$ (set of items)
  - A known **taxonomy** $\Psi$ on $\mathcal{I}$ (is-a relation, partial order)
  - A crowd **oracle** to decide if an itemset is frequent or not

- Choose questions **interactively** based on past answers

$\Rightarrow$ Find out the status of **all** itemsets

**What is a good algorithm to solve this problem?**

**Crowd complexity:** The number of itemsets we ask about

(monetary cost, latency...)

**Computational complexity:** The complexity of computing the next question to ask
On the Complexity of Mining Itemsets from the Crowd Using Taxonomies

Antoine Amarilli¹,², Yael Amsterdamer¹, and Tova Milo¹

¹Tel Aviv University, Tel Aviv, Israel
²École normale supérieure, Paris, France

ABSTRACT

We study the problem of frequent itemset mining in domains where data is not recorded in a conventional database but only exists in human knowledge. We provide examples of such scenarios, and present a crowdsourcing model for them. The model uses the crowd as an oracle to find out whether an itemset is frequent or not, and relies on a known taxonomy of the item domain to guide the search for frequent itemsets. In the spirit of data mining with oracles, we analyze the complexity of this problem in terms of (i) crowd complexity, that measures the number of crowd questions required to identify individuals involved. As another example, consider a health researcher who wants to identify new drugs by analyzing the practices of folk medicine (also known as traditional medicine, i.e., medicinal practice that is neither documented in writing nor tested out under a scientific protocol): the researcher may want to determine, for instance, which treatments are often applied together for a given combination of symptoms. For this purpose too, the main source of knowledge are the folk healers and patients themselves.

In a previous work [2, 3] we have proposed to address
Ongoing extensions

- Two important aspects to handle:
  - The support of itemsets is a **numerical value**
    - Use them to estimate probabilities
  - Only the **most frequent itemsets** are really relevant
    - Focus on finding relevant queries for **top-k**
Ongoing extensions

- Two important aspects to handle:
  - The support of itemsets is a numerical value
  - Use them to estimate probabilities
  - Only the most frequent itemsets are really relevant
  - Focus on finding relevant queries for top-$k$

- Unexpected connections:
  - volume computation in convex polytopes
  - interpolation schemes for posets
Ongoing extensions

- Two important aspects to handle:
  - The support of itemsets is a numerical value
  - Use them to estimate probabilities
  - Only the most frequent itemsets are really relevant
  - Focus on finding relevant queries for top-k
- Unexpected connections:
  - Volume computation in convex polytopes
  - Interpolation schemes for posets
- Vision published at Uncrowd 2014 [Amarilli et al., 2014b]
- Ongoing work

Uncertainty in Crowd Data Sourcing under Structural Constraints

Antoine Amarilli¹, Yael Amsterdamer², and Tova Milo²

¹ Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI, Paris, France
² Tel Aviv University, Tel Aviv, Israel
<table>
<thead>
<tr>
<th></th>
<th>Table of contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Research Topic</td>
</tr>
<tr>
<td>2</td>
<td>Tractability for Treelike Probabilistic Data</td>
</tr>
<tr>
<td>3</td>
<td>Open-World Query Answering</td>
</tr>
<tr>
<td>4</td>
<td>Crowd Data Mining</td>
</tr>
<tr>
<td>5</td>
<td>Other Topics</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Joint work with M. Lamine Ba (Télécom ParisTech), Daniel Deutch (Tel Aviv University) and Pierre Senellart
Extend the positive (bag) relational algebra to ordered data
Manage uncertainty on the possible orderings
Study expressiveness and complexity
Uncertain ordered data

- Joint work with M. Lamine Ba (Télécom ParisTech), Daniel Deutch (Tel Aviv University) and Pierre Senellart
- Extend the positive (bag) relational algebra to ordered data
- Manage uncertainty on the possible orderings
- Study expressiveness and complexity
- Unsuccessfully submitted to PODS 2014
- Hoping to submit to PODS 2015 (deadline tomorrow :-P)

Provenance for Nondeterministic Order-Aware Queries
Antoine Amarilli
Télécom ParisTech; CNRS LTCI
M. Lamine Ba
Télécom ParisTech; CNRS LTCI
Daniel Deutch
Tel Aviv University

ABSTRACT
Data transformations that involve (partial) ordering, and consolidate data in presence of uncertainty, are common in the context of various applications. The complexity of such transformations, in addition to the possible presence of meta-data, call for provenance support. We introduce, for the first time, a framework that accounts for the conjunction of these needs. To this end, we enrich the positive relational algebra with order-aware operators, some of which are non-deterministic, accounting for uncertainty. We study the expressive power and the complexity of deciding possibility for the obtained language. We then equip the language with (semiing-based) provenance tracking and highlight the unique challenges in supporting provenance for the order-aware operations. We explain how to overcome these challenges, designing a new provenance structure and a provenance-aware semantics for our language. We show the usefulness of the construction, proving that it satisfies common desiderata for provenance tracking.

1. INTRODUCTION
Real-world applications often involve transformations that involve some (partial) ordering in the data; that need to consolidate data in presence of uncertainty. As a consequence, the orderings, or for scheduling of workflows, with constraints on tasks order and possible synchronization points. In all of these cases there is an inherent uncertainty in the transformations. As explained below, we take the operational approach of dealing with this uncertainty via non-determinism.

Consider for example a sensor network where each sensor issues observations on events happening within its range. We assume that information about events observed by a given sensor is saved in a relation and are ordered by timestamps. Observations of the different sensors need to be consolidated, to provide a complete picture of events and allow for their analysis. However, we may not trust the relative ordering of observations across sensors, as global clock synchronization is a tricky matter [30]; or maybe we can trust the relative ordering between sensors but only once some synchronization point has been reached (e.g. an event that is known to be common has been reported).

A Need for Provenance Tracking. Importantly, metadata may affect the transformation and consolidation of data. Continuing with our sensors example, each observation of each sensor may be associated with a different level of credibility (trust), depending e.g., on the sensor quality; some observations may be associated with different access control policies.

Querying Order-Incomplete Data
Antoine Amarilli
Télécom ParisTech; CNRS LTCI
Pierre Senellart
Télécom ParisTech; CNRS LTCI
M. Lamine Ba
Institut Mines-Télécom
Télécom ParisTech; CNRS LTCI
Daniel Deutch
Blavatnik School of Computer Science
Tel Aviv University

ABSTRACT
To combine ordered data originating from multiple sources, one needs a framework that can represent uncertainty about the possible orderings or, as we call, in-order-incomplete data. Examples of order-incomplete data are lists of properties (such as hotels and restaurants) ranked by an unknown function reflecting relevance or customer ratings, documents edited concurrently with uncertainty on the order of contributions, and the result of integrating event sequences such as sensor readouts or log entries. Our work extends the positive relational algebra to ordered and order-incomplete data, and introduces a set of axioms to guide the design of a bag semantics for the language, motivated by our use cases. We introduce two simple such semantics, one of which is shown to be the most general for our set of axioms. We next design a strong representation system for them, based on partial orders interpreted through a possible-world semantics. We study the expressiveness of our query language, connecting it to complexity measures on partial orders. We further introduce a top-k operator, and investigate the complexity of query evaluation, studied in the context of certain and possible answers. We last introduce a duplicate elimination operator to return to set semantics, and revisit our results.

1. INTRODUCTION
Real-world applications usually involve transformations over ordered data with incomplete knowledge about how in-
Possibility for probabilistic XML

- **Probabilistic XML**: represent uncertain XML documents
- Given such a document $D$ and deterministic document $W$:
  - is $W$ a possible world of $D$?
  - what is the probability of $D$?
- Show tractable and intractable problem settings
Possibility for probabilistic XML

- **Probabilistic XML**: represent uncertain XML documents
- Given such a document $D$ and deterministic document $W$:
  - is $W$ a possible world of $D$?
  - what is the probability of $D$?
- Show tractable and intractable problem settings
- Presented at AMW 2014 [Amarilli, 2014b]
- Extended version at BDA 2014

---

**The Possibility Problem for Probabilistic XML**
(Extended Version)

Antoine Amarilli
Télécom ParisTech; Institut Mines-Télécom; CNRS LTCI

**Abstract.** We consider the *possibility problem* of determining if a document is a possible world of a probabilistic document, in the setting of probabilistic XML. This basic question is a special case of query answering or tree automata evaluation, but it has specific practical uses, such as checking whether an user-provided probabilistic outcome is possible. Our results entirely classify the tractability boundary over all considered problem variants.
XML data pricing

- Joint work with Ruiming Tang and Stéphane Bressan (National University of Singapore) and Pierre Senellart
- Data pricing: set the price on intensional data accesses
- Here, incomplete fragments offered at a discount
- How to sample uniformly a subtree for the requested price
Joint work with Ruiming Tang and Stéphane Bressan (National University of Singapore) and Pierre Senellart

Data pricing: set the price on intensional data accesses

Here, incomplete fragments offered at a discount

How to sample uniformly a subtree for the requested price

Presented at DEXA 2014 [Tang et al., 2014]

Extended version to be submitted in TLKDS special issue

Planning to write a challenge paper for JDIQ

Ongoing work on efficiently samplable document classes

---

Abstract. While price and data quality should define the major trade-off for consumers in data markets, prices are usually prescribed by vendors and data quality is not negotiable. In this paper we study a model where data quality can be traded for a discount. We focus on the case of XML documents and consider completeness as the quality dimension. In particular, we present a framework to compute the best subset of an XML document for each query, where the complete subtree is always included in the result. The result is provided in pseudo-polynomial time, based on the fact that data quality is computed by querying the document. This approach results in a framework to get a sample for a discount.
Recent Topics of Research around the YAGO Knowledge Base

Antoine Amarilli1, Luis Galárraga1, Nicoleta Preda2, and Fabian M. Suchanek1

1 Télécom ParisTech, Paris, France
2 University of Versailles, France

Abstract. A knowledge base (KB) is a formal collection of knowledge about the world. In this paper, we explain how the YAGO KB is constructed. We also summarize our contributions to different aspects of KB management in general. One of these aspects is rule mining, i.e., the identification of patterns such as spouse(x, y) ∧ livesIn(x, z) ⇒ livesIn(y, z). Another aspect is the incompleteness of KBs. We propose to integrate data from Web Services into the KB in order to fill the gaps. Further, we show how the overlap between existing KBs can be used to align them, both in terms of instances and in terms of the schema. Finally, we show how KBs can be protected by watermarking.

1 Introduction

Recent advances in information extraction have led to the creation of large knowledge bases (KBs). These KBs provide information about a great variety of entities, such as people, countries, rivers, cities, universities, movies, animals, etc. Among the most prominent academic projects are Cyc [12], DBpedia [2], Freebase4, and our own YAGO [21]. Most of these projects are linked together in different ways, and this is clearly an entity name, but not the correct one. Worse, some Web pages contain several ids and several entity names at the same time, so we must correctly match the ids and names on the page. The excerpt of Figure 1 is taken from a page that lists dozens of Samsung products.

Finally, if we want to find entity ids and names at Web scale, we need an approach that is both fast and resilient. It must run on hundreds of millions of Web pages, and it must accept entirely arbitrary pages, with possibly erroneous content, broken structure, or noisy information. This makes it impossible to rely on wrapper induction, or indeed on any predefined or learned DOM tree structure. We have to be able to find the entity names in tables, in lists, as well as in plain unstructured text. These challenges come in addition to the usual difficulties such as non-standard HTML code, non-semantic markup (e.g., tables used for page layout), and creative tag combinations to arrange tabular information.

Contribution. In this paper, we show how to systematically collect unique ids from Web pages, and how to associate each id to the correct entity name. We first use vanilla NER methods to extract ids and candidate names from each Web page. Then, we rely on the inherent characteristics of unique identifiers to filter the name candidates so as to keep only the correct names for the entities. Our method is scalable, fast, and resilient enough to run on arbitrary Web pages. This allows us to extract millions of distinct entities from the Web, with an accuracy of 75% to 96% depending on the entity types. The result is a database of entity ids and names, with information about which pages mention which entities. The crucial advantage of our database is that every entry is guaranteed to be unique, so we can count distinct entities without being biased by duplicates. Thus, we can perform a detailed study of entities that exist on the Web: we can identify Web sites that are hubs for books or documents, we can build statistics about frequent first names of people, and we can determine which countries produce most products.
<table>
<thead>
<tr>
<th>1</th>
<th>Research Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Tractability for Treelike Probabilistic Data</td>
</tr>
<tr>
<td>3</td>
<td>Open-World Query Answering</td>
</tr>
<tr>
<td>4</td>
<td>Crowd Data Mining</td>
</tr>
<tr>
<td>5</td>
<td>Other Topics</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Conclusion

- Uncertainty, Intensionality, Structure
- **Main focus:** tractable probabilistic data and rules
- **Next steps:**
  - Study *feasibility* of practical implementations
  - Extend to probabilistic rules
  - Finish *writing up* other lines of work
Conclusion

- Uncertainty, Intensionality, Structure
- **Main focus:** tractable probabilistic data and rules
- **Next steps:**
  - Study feasibility of practical implementations
  - Extend to probabilistic rules
  - Finish writing up other lines of work

Thanks for your attention!


Harvesting entities from the web using unique identifiers.
Submitted to WWW 2015.

Get a sample for a discount: Sampling-based XML data pricing.
In Proc. DEXA, Munich, Germany.