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* $\geq 20\,m^2$ *?*

Intensionality

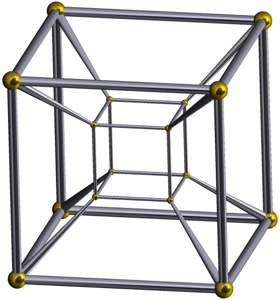
- We cannot collect all information:
  - → Storage space
  - → Bandwidth
  - → Access restrictions
- Need to access remote data sparingly
- Choose relevant accesses dynamically

→ *Web crawling*	→ *Crowdsourcing*	→ *Expensive processing*
→ *Web APIs*	→ *Deep Web*	→ *Rule consequences*

Structure

- Need to leverage existing structure
- Structure can be heterogeneous

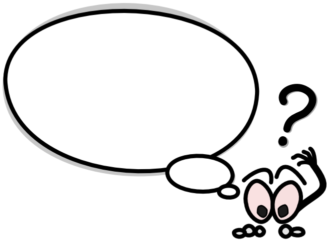
→ Avoid focusing only on one framework



→ *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

$\rightarrow$ Start with an initial knowledge about the world

$\rightarrow$ Locate promising articles

$\rightarrow$ Run expensive processing on the articles

$\rightarrow$ Uncertainty when accessing, disambiguating

$\rightarrow$ Use crowdsourcing to validate the facts

$\rightarrow$ Using logical rules to constrain them

Our vision of a general approach

- Unsucessfully 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 Pierre Senellart
Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI; Paris, France
firstname.lastname@telecom-paristech.fr

ABSTRACT

To answer user queries on Web data, it is necessary to crawl, extract, enrich, and process available information. The traditional *extensional* approach is to perform those steps one after the other, but it has many drawbacks. The choice of information that we retrieve and process must be guided by the query, because retrieving all the information is not feasible; the information cannot be maintained locally because it may become obsolete rapidly; it cannot be trusted blindly, as it may come from untrustworthy sources; it must be stored in a way which accounts for its heterogeneous structure (Web pages, relational facts, textual content, etc.). In this paper, we present UnSAID, our vision of a framework which addresses simultaneously the three main challenges faced by the extensional approach: *intensionality*, the need to access data selectively and take into account the cost of individual accesses; *uncertainty*, the need to reason on partial and inexact views of the world; and *structure*, the need to deal with data in various heterogeneous forms.

1. INTRODUCTION

Publicly available data, information, knowledge is abundant: the World Wide Web contains trillions of pages on an amazingly diverse collection of topics; hundreds of thousands of deep Web databases, accessible through Web forms, are also available; a social networking site such as Twitter sees hundreds of millions of new (public) messages posted *each day*; the open linked data now contains hundreds of knowledge bases covering tens of billions of semantic facts in the form of RDF triples; complex tools in areas such as information extraction, data mining, or natural language processing (NLP) are readily available to enrich existing data with even more information; rules mined from data, or machine learning models, can be used to make predictions; and when the data is not there and cannot be predicted, or when it is not easy to process automatically, it is always possible to resort to crowdsourcing platforms such as

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., in 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, annotators 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?" or "What is the optimal way to go from this place to that place at a given time of 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 archives* [26]: their goal is to build semantically annotated Web archives on specific topics or events (investment for growth in Europe, the 2014 Winter Olympics, etc.), guiding the process with clues from the social Web as to which documents are relevant. These archives

What Is the Best Thing to Do Next?

A Tutorial on Intensional Data Management

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

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

firstname.lastname@telecom-paristech.fr

ABSTRACT

We call data *intensional* when it is not directly available, but must be accessed through a costly interface. Intensional data naturally arises in a number of data management scenarios, such as crowdsourcing, Web crawling, or ontology-based data access. Such scenarios require us to model an uncertain view of the world, for which, given a query, we must answer the question "What is the best thing to do next?" Once data has been retrieved, 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.

1. INTRODUCTION

Intensional Data Management. Many data-centric applications involve data that is not directly available in extension, but can only be obtained after some access to the data is made, at some form of cost. In traditional database querying [13], the access may be disk I/O, and the I/O cost will depend on which indexes are available. In crowdsourcing platforms [4, 25], accessing data involves recruiting a worker to provide the data, and the cost is in terms of monetary compensation for workers and latency to obtain the data. In Web crawling [16], accesses are HTTP requests and cost involves bandwidth usage, network latency, and quota use for rate-limited interfaces. In ontology-based data access [10], accesses mean applying a reasoning rule of an ontology, and the cost is the computational cost of such an evaluation.

We abstract out the general problem of accessing data through costly interfaces as that of *intensional data management*. This ter-

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 *priors* 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" in order to answer the query, meaning, what is the best access that should be performed at this point, given its cost, potential gain, and the uncertain knowledge of the world. Once an access is chosen and performed, some data is retrieved, and the uncertain view of the world must be revised in light of the new knowledge obtained. The process is repeated until the user's query receives a satisfactory answer or some other termination condition is met.

Use Cases. 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., in 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, annotators 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

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

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:

Paper	Conference	Proba
1	PODS	0.2
1	ICDT	0.3
2	PODS	0.4
2	ICDT	0.5

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

Background: Instances and queries

- Given a relational instance with probabilities:

Paper	Conference	Proba
1	PODS	0.2
1	ICDT	0.3
2	PODS	0.4
2	ICDT	0.5

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

$\rightarrow$ Query evaluation: probability that q holds?

Background: Instances and queries

- Given a relational instance with probabilities:

Paper	Conference	Proba
1	PODS	0.2
1	ICDT	0.3
2	PODS	0.4
2	ICDT	0.5

- 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:

Paper	Conference	Proba
1	PODS	0.2
1	ICDT	0.3
2	PODS	0.4
2	ICDT	0.5

- Given a conjunctive query (CQ) (existentially quantified)
  $q : \exists p_1 p_2 c \; \mathrm{Accepted}(p_1, c) \land \mathrm{Accepted}(p_2, c) \land p_1 \neq p_2$

→ 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! :-(

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

instance I

$R(a, b)$ $R(b, c)$ $S(c)$

Bounded treewidth

An idea from instances without probabilities...

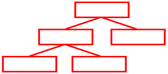
- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

instance I ⟶ tree encoding T_I

$R(a, b)$ $R(b, c)$ $S(c)$

tree decomposition

$O(|I|)$ for fixed width



Bounded treewidth

An idea from instances without probabilities...

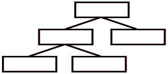
- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

instance $I \longrightarrow$ tree encoding T_I

$R(a, b)$ $R(b, c)$ $S(c)$

tree decomposition

$O(|I|)$ for fixed width



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

query q

Bounded treewidth

An idea from instances without probabilities...

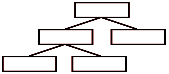
- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

instance I ⟶ tree encoding T_I

$R(a, b)$ $R(b, c)$ $S(c)$

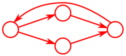
tree decomposition

O($|I|$) for fixed width



rewriting

O(1) data complexity



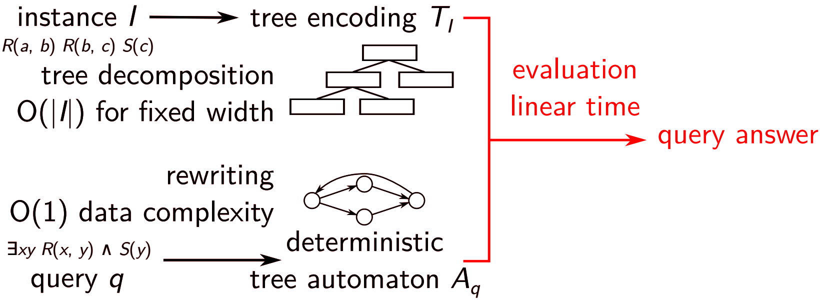
$\exists xy\ R(x, y) \wedge S(y)$

query q ⟶ deterministic tree automaton A_q

Bounded treewidth

An idea from instances without probabilities...

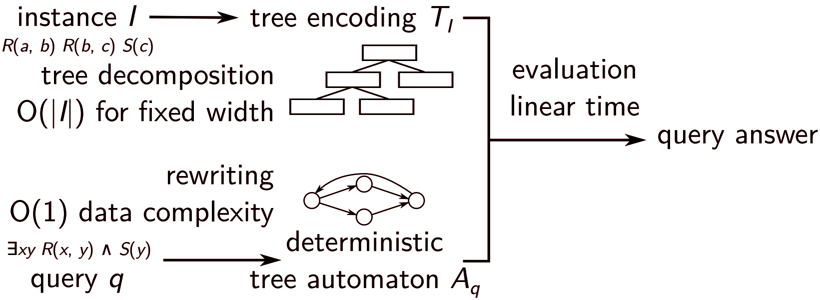
- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...



Bounded treewidth

An idea from instances without probabilities...

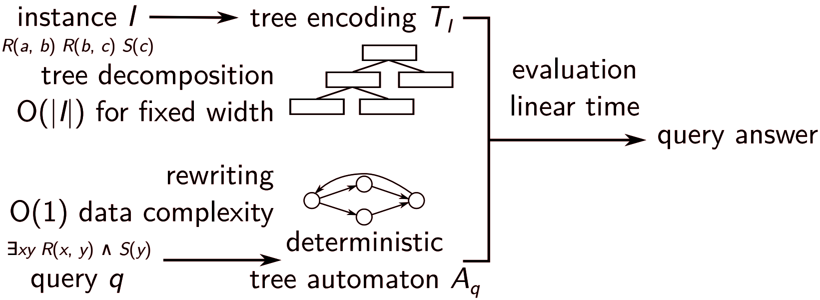
- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...



Bounded treewidth

An idea from instances without probabilities...

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



→ 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

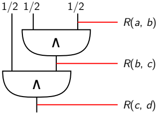
Our idea

- 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

$\rightarrow$ Compute query probability in linear time
(assuming fixed-cost arithmetics)

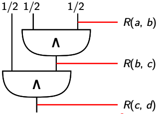
Main result in pictures

instance I



Main result in pictures

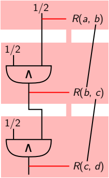
instance I



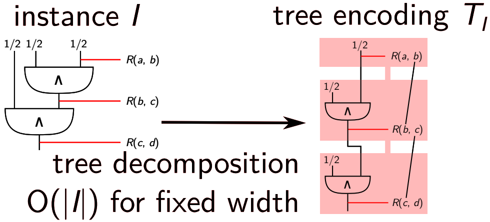
tree decomposition

O(|I|) for fixed width

tree encoding T_I



Main result in pictures

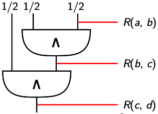


$\exists xy\ R(x, y) \land S(y)$

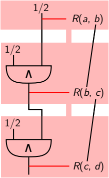
query q

Main result in pictures

instance I



tree encoding T_I



tree decomposition

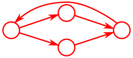
$O(|I|)$ for fixed width

rewriting

$O(1)$ data complexity

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

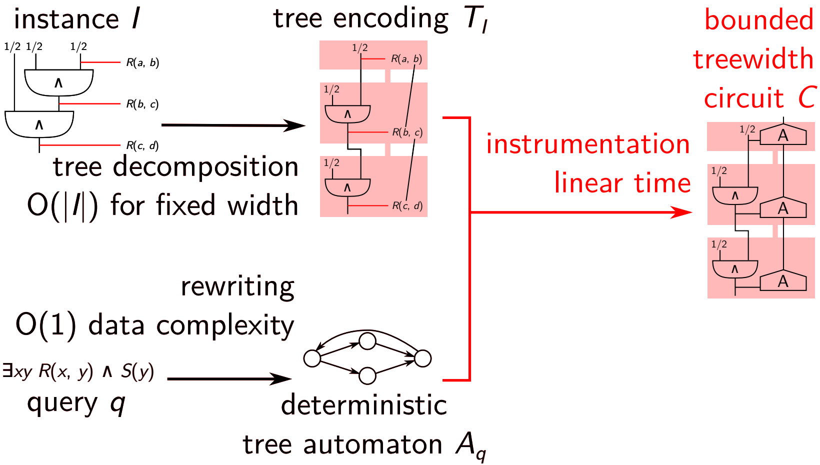
query q



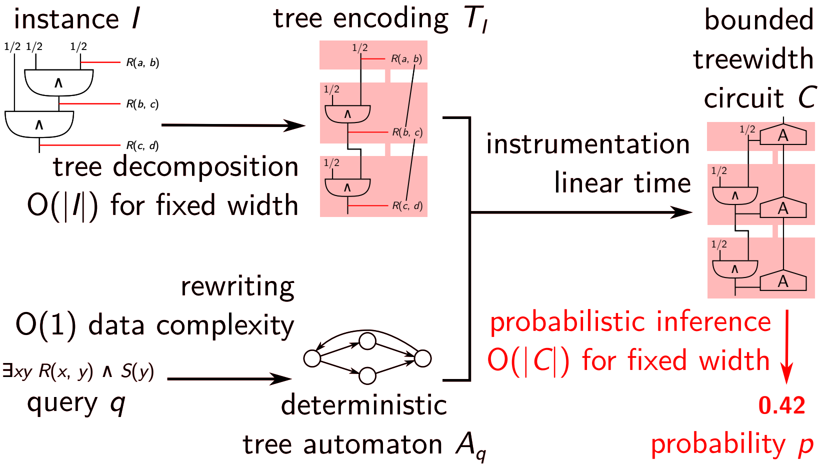
deterministic

tree automaton A_q

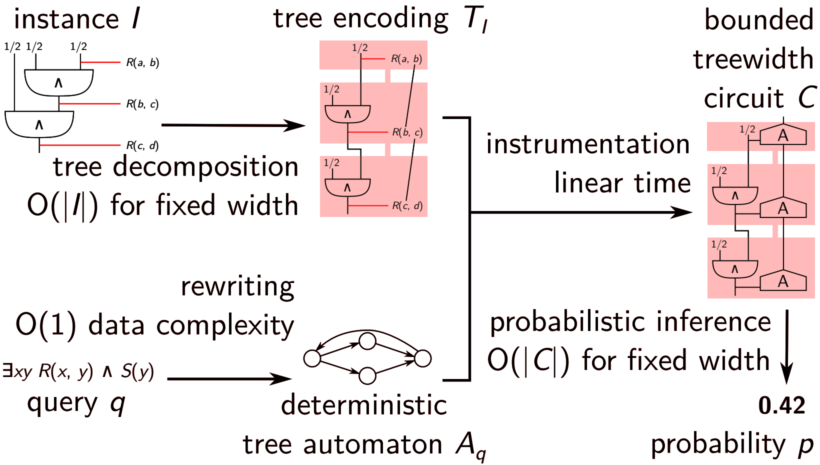
Main result in pictures



Main result in pictures



Main result in pictures



Specific consequences

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

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

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 ...
  - → 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 $O(I)$!

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 $O(I)$!

- 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 *treewidth*, 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-treewidth 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-treewidth instances, so that the probability of the query can then be evaluated in linear-time data complexity (assuming unit-

is bounded [17], intuitively restricting them to be close to trees. Such results also apply, e.g., to counting and reliability calculations [6], which suggests a natural question: can we adapt them to query evaluation on probabilistic instances and show tractability assuming bounded treewidth?

Two obstacles make this question harder to answer. First, there are many probabilistic frameworks (TID, BID, probabilistic c-tables, probabilistic XML...), so it is difficult to define a general notion of treewidth for all of them. Second, probabilistic models such as pc-tables have probabilistic *correlations* which can also cause hardness even for a trivial underlying instance: it is not clear how to bound simultaneously the instances and the correlations.

This work presents a solution to both of these problems. We introduce the probabilistic framework of *pcc-instances*, a straightforward extension of pc-tables with tuple annotations

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

MSc Internship

Querying Probabilitistic Data via Tree Decompositions

Pierre Senellart

Télécom ParisTech & National University of Singapore

Topic description

Probabilistic databases are compact representations of probability distributions over regular databases. A number of models have been proposed for probabilistic data, both relational [7] and XML [4]. Evaluating a Boolean query over such a probabilistic database means computing the probability that the query is true in the probability distribution represented by the database. While query evaluation is usually tractable on regular databases, evaluating queries in this sense on probabilistic databases is often intractable.

A number of research works have looked at characteristics of *queries* that can make them tractable. For instance, queries without self-joins are tractable over tuple-independent databases if and only if they are hierarchical [2], while tree-pattern queries on XML data with a single join are

Table of contents

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 Σ

→ Is q a logical consequence of I and Σ?

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 Σ

→ Is q a logical consequence of I and Σ?

- Constraints:
  - Unary inclusion dependencies (UID)
    Example: $\forall xy\, \text{Reviews}(x, y) \Rightarrow \exists z\, \text{Reviews}(y, z)$
  - Functional dependencies (FD)
    Example: $\forall xyz\, \text{Reviews}(x, z) \wedge \text{Reviews}(y, z) \Rightarrow x = y$

Finite vs unrestricted query answering

- Unrestricted QA:
  $I, \Sigma \models q$ if $J \models q$ for all $J \supseteq I$ s.t. $J \models \Sigma$
- Finite QA:
  $I, \Sigma \models q$ if $J \models q$ for all finite $J \supseteq I$ s.t. $J \models \Sigma$

Finite vs unrestricted query answering

- Unrestricted QA:
  $I, \Sigma \models q$ if $J \models q$ for all $J \supseteq I$ s.t. $J \models \Sigma$
- Finite QA:
  $I, \Sigma \models q$ if $J \models q$ for all finite $J \supseteq I$ 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:
  $I, \Sigma \models q$ if $J \models q$ for all $J \supseteq I$ s.t. $J \models \Sigma$
- Finite QA:
  $I, \Sigma \models q$ if $J \models q$ for all finite $J \supseteq I$ 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 Σ entail constraint τ:
  every instance satisfying Σ also satisfies τ
- 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

$\rightarrow$ UIDs/FDs are finitely controllable modulo finite closure

Our result

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

$\rightarrow$ 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

$\rightarrow$ 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 GC^2 and frontier-one acyclic dependencies

Conference submission

- 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"

Conference submission

- 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"
- Writing up the main result for LICS 2015
- Deadline in 1 month 1/2
- Possible further submission for the other result

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 its 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 [31]; 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 unraveling 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 *reification* 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 con-

Finite Open-World Query Answering with Number Restrictions

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

Michael Benedikt
Oxford University
Email: michael.benedikt@cs.ox.ac.uk

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 bold in prelim] [TODO: restate thms in apx]

I. INTRODUCTION

that, in fact, they 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 $\forall \vec{x}\vec{y}\, R(x_1 \ldots x_n) \wedge R(y_1 \ldots y_n) \wedge \bigwedge x_i = y_i \rightarrow x_j = y_j$. 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-incide.

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 IDs and FDs, 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

Table of contents

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

$$D = \{ \{\mathtt{beer}, \mathtt{diapers}\}, \{\mathtt{beer}, \mathtt{bread}, \mathtt{butter}\}, \{\mathtt{beer}, \mathtt{bread}, \mathtt{diapers}\}, \{\mathtt{salad}, \mathtt{tomato}\} \}$$

- 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

$$D = \{ \{\mathtt{beer}, \mathtt{diapers}\}, \{\mathtt{beer}, \mathtt{bread}, \mathtt{butter}\}, \{\mathtt{beer}, \mathtt{bread}, \mathtt{diapers}\}, \{\mathtt{salad}, \mathtt{tomato}\} \}$$

- Itemset is frequent if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\mathtt{salad}\}$ 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

$$D = \{ \{\texttt{beer}, \texttt{diapers}\}, \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \{\texttt{salad}, \texttt{tomato}\} \}$$

- Itemset is frequent if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\texttt{salad}\}$ 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

$$D = \{ \\ \{\texttt{beer}, \texttt{diapers}\}, \\ \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \\ \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \\ \{\texttt{salad}, \texttt{tomato}\} \\ \}$$

- Itemset is frequent if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\texttt{salad}\}$ 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

$$D = \{ \{\mathtt{beer}, \mathtt{diapers}\}, \{\mathtt{beer}, \mathtt{bread}, \mathtt{butter}\}, \{\mathtt{beer}, \mathtt{bread}, \mathtt{diapers}\}, \{\mathtt{salad}, \mathtt{tomato}\} \}$$

- Itemset is frequent if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\mathtt{salad}\}$ not frequent
- $\{\mathtt{beer}, \mathtt{diapers}\}$ 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

$$D = \{ \{\texttt{beer}, \texttt{diapers}\}, \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \{\texttt{salad}, \texttt{tomato}\} \}$$

- Itemset is frequent if it occurs in $\geq \Theta = 50\%$ of transactions
- $\{\texttt{salad}\}$ not frequent
- $\{\texttt{beer}, \texttt{diapers}\}$ 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

$$D = \{ \{\texttt{beer}, \texttt{diapers}\}, \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \{\texttt{salad}, \texttt{tomato}\} \}$$

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

$$D = \{ \{\texttt{beer}, \texttt{diapers}\}, \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \{\texttt{salad}, \texttt{tomato}\} \}$$

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

$$D = \{ \{\texttt{beer}, \texttt{diapers}\}, \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \{\texttt{salad}, \texttt{tomato}\} \}$$

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

$$D = \{ \{\texttt{beer}, \texttt{diapers}\}, \{\texttt{beer}, \texttt{bread}, \texttt{butter}\}, \{\texttt{beer}, \texttt{bread}, \texttt{diapers}\}, \{\texttt{salad}, \texttt{tomato}\} \}$$

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

$\rightarrow$ 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.
    $\Rightarrow$ $\{\texttt{acropolis}, \texttt{museum}\}$
  - t_2: I visited Piraeus and had some ice cream.
    $\Rightarrow$ $\{\texttt{piraeus}, \texttt{icecream}\}$
  - t_3: On Monday I attended the keynote and had coffee.
    $\Rightarrow$ $\{\texttt{keynote}, \texttt{coffee}\}$

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.
    $\Rightarrow$ $\{\texttt{acropolis}, \texttt{museum}\}$
  - t_2: I visited Piraeus and had some ice cream.
    $\Rightarrow$ $\{\texttt{piraeus}, \texttt{icecream}\}$
  - t_3: On Monday I attended the keynote and had coffee.
    $\Rightarrow$ $\{\texttt{keynote}, \texttt{coffee}\}$
- 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?"

$\Rightarrow$ We can just ask people if an itemset is frequent

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)*

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)*

$\Rightarrow$ 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 Ψ 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 Ψ 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?

The problem

We can now describe the problem:

- We have:
  - A known item domain $\mathcal{I}$ (set of items)
  - A known taxonomy Ψ 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

Conference publication

- Published at ICDT 2014 [Amarilli et al., 2014a]
- Presented at Tel Aviv University and in Lille
- Connections to work by people in Lille [Bonifati et al., 2014]

On the Complexity of Mining Itemsets from the Crowd Using Taxonomies*

Antoine Amarilli[1,2], Yael Amsterdamer[1], and Tova Milo[1]

[1]Tel Aviv University, Tel Aviv, Israel
[2]École 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 iden-

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

  $\rightarrow$ Use them to estimate probabilities
  - Only the most frequent itemsets are really relevant

  $\rightarrow$ 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 Amarilli[1], Yael Amsterdamer[2], and Tova Milo[2]

[1] Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI, Paris, France

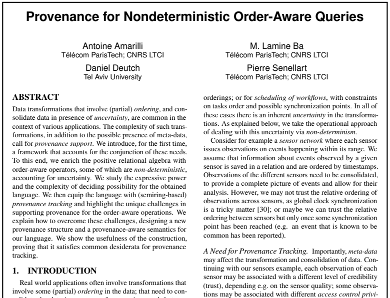
Table of contents

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

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

Pierre Senellart
Télécom ParisTech; CNRS LTCI

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 (semiring-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.

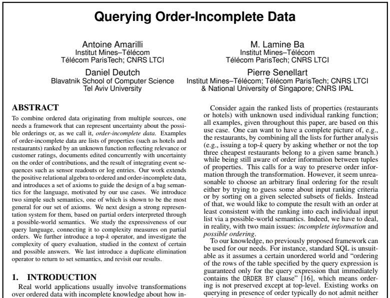
1. INTRODUCTION

Real world applications often involve transformations that involve some (partial) *ordering* in the data; that need to con-

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, *meta-data* 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 privi-*



Querying Order-Incomplete Data

Antoine Amarilli
Institut Mines–Télécom
Télécom ParisTech; CNRS LTCI

M. Lamine Ba
Institut Mines–Télécom
Télécom ParisTech; CNRS LTCI

Daniel Deutch
Blavatnik School of Computer Science
Tel Aviv University

Pierre Senellart
Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI
& National University of Singapore; CNRS IPAL

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 it, *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-

Consider again the ranked lists of properties (restaurants or hotels) with unknown used individual ranking function; all examples, given throughout this paper, are based on this use case. One can want to have a complete picture of, e.g., the restaurants, by combining all the lists for further analysis (e.g., issuing a top-k query by asking whether or not the top three cheapest restaurants belong to a given same branch.) while being still aware of order information between tuples of properties. This calls for a way to preserve order information through the transformation. However, it seem unreasonable to choose an arbitrary final ordering for the result either by trying to guess some about input ranking criteria or by sorting on a given selected subsets of fields. Instead of that, we would like to compute the result with an order at least consistent with the ranking into each individual input list via a possible-world semantics. Indeed, we have to deal, in reality, with two main issues: *incomplete information* and *possible ordering*.

To our knowledge, no previously proposed framework can be used for our needs. For instance, standard SQL is unsuitable as it assumes a certain unordered world and "ordering of the rows of the table specified by the query expression is guaranteed only for the query expression that immediately contains the `ORDER BY` clause" [16], which means ordering is not preserved except at top-level. Existing works on querying in presence of order typically do not admit neither

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-

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

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

→ 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

Get a Sample for a Discount
Sampling-Based XML Data Pricing

Ruiming Tang[1], Antoine Amarilli[2], Pierre Senellart[2], and Stéphane Bressan[1]

[1] National University of Singapore, Singapore
{tangruiming,steph}@nus.edu.sg
[2] Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI. Paris, France
{antoine.amarilli,pierre.senellart}@telecom-paristech.fr

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

A Framework for Sampling-Based XML Data Pricing

Ruiming Tang[1], Antoine Amarilli[2], Pierre Senellart[2], and Stéphane Bressan[1]

[1] National University of Singapore, Singapore
{tangruiming,steph}@nus.edu.sg
[2] Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI. Paris, France
{antoine.amarilli,pierre.senellart}@telecom-paristech.fr

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.

Knowledge bases

- Helped write an invited paper to APWEB 2014 [Amarilli et al., 2014d]
- Helped rewrite a submission to WWW 2015 [Talaika et al., 2014]

Recent Topics of Research around the YAGO Knowledge Base

Antoine Amarilli[1], Luis Galárraga[1], Nicoleta Preda[2], and Fabian M. Suchanek[1]

[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 $spouse(x, y) \wedge livesIn(x, z) \Rightarrow livesIn(y, z)$. 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], Freebase[3], and our own YAGO [21]. Most of these projects are linked together

Harvesting Entities from the Web Using Unique Identifiers

Aliaksandr Talaika[1], Joanna Biega[1], Antoine Amarilli[2], Fabian M. Suchanek[2]
[1] Max Planck Institute for Informatics, Germany
[2] Télécom ParisTech; Institut Mines-Télécom; CNRS LTCI

ABSTRACT

In this paper we study the prevalence of unique entity identifiers on the Web. These are, e.g., ISBNs (for books), GTINs (for commercial products), DOIs (for documents), email addresses, and others. We show how these identifiers can be harvested systematically from Web pages, and how they can be associated with human-readable names for the entities at large scale.

Starting with a simple extraction of identifiers and names from Web pages, we show how we can use the properties of unique identifiers to filter out noise and clean up the extraction result on the entire corpus. The end result is a database of millions of uniquely identified entities of different types, with an accuracy of 73–96% and a very high coverage compared to existing knowledge bases. We use this database to compute novel statistics on the presence of products, people, and other entities on the Web.

1. INTRODUCTION

Unique ids. The Web is an almost endless resource of named entities, such as commercial products, people, books, and organizations. In this paper, we focus on those entities that have unique *ids*. An id is any string or number that distinguishes the entity in a globally unique way from other entities. For example, commercial products have ids in the form of GTIN codes. These are the numeric codes printed below the bar code on the package or item. They also frequently appear on the Web. Figure 1 shows an excerpt from a Web page about a commercial product. The GTIN (8806085725072) appears at the bottom right.

Figure 1: A Web page snippet about a product

But not just commercial products have ids. A surprisingly large

the entity. In the example, the challenge is to find that the correct name for the id "8806085725072" is "Samsung Galaxy S4" – and not "Samsung", "VAT", or "GT-I9295ZAADBT".

It is far from trivial to associate the correct entity name to an id. First, Web pages contain usually dozens of entity names, so it is not clear which one corresponds to the id. In the example, "Samsung" 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 learnable 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 73% 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 entity 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 prod-

Table of contents

Conclusion

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

Thanks for your attention!

Amarilli, A. (2014a).
Open-world query answering under number restrictions.
Preprint:
http://a3nm.net/publications/amarilli2014open.pdf.

Amarilli, A. (2014b).
The possibility problem for probabilistic XML.
In *Proc. AMW*, Cartagena, Colombia.

Amarilli, A., Amsterdamer, Y., and Milo, T. (2014a).
On the complexity of mining itemsets from the crowd using taxonomies.
In *Proc. ICDT*, Athens, Greece.

References II

Amarilli, A., Amsterdamer, Y., and Milo, T. (2014b).
Uncertainty in crowd data sourcing under structural constraints.
In *Proc. UnCrowd*, Denpasar, Indonesia.

Amarilli, A., Bourhis, P., and Senellart, P. (2014c).
Probabilities and provenance via tree decompositions.
Preprint: `http://a3nm.net/publications/amarilli2015probabilities.pdf`. Submitted to PODS 2015.

Amarilli, A., Galárraga, L., Preda, N., and Suchanek, F. M. (2014d).
Recent topics of research around the YAGO knowledge base.
In *APWEB*.

References III

Amarilli, A. and Senellart, P. (2014a).
UnSAID: Uncertainty and structure in the access to intensional data.
Preprint: http://a3nm.net/publications/amarilli2014unsaid.pdf.
Vision article.

Amarilli, A. and Senellart, P. (2014b).
What is the best thing to do next?: A tutorial on intensional data management.
Preprint: http://a3nm.net/publications/amarilli2015what.pdf.
Tutorial proposal. Submitted to EDBT/ICDT 2015.

References IV

Barany, V., Gottlob, G., and Otto, M. (2010).
Querying the guarded fragment.
In *LICS*.

Bonifati, A., Ciucanu, R., and Staworko, S. (2014).
Interactive inference of join queries.
In *Proc. EDBT*, Athens, Greece.

Galárraga, L., Teflioudi, C., Hose, K., and Suchanek, F. M. (2013).
AMIE: association rule mining under incomplete evidence in ontological knowledge bases.
In *Proc. WWW*, Rio de Janeiro, Brazil.

References V

Ibáñez-García, Y., Lutz, C., and Schneider, T. (2014).
Finite model reasoning in horn description logics.
In *Proc. KR*, Vienna, Austria.

Maniu, S., Cheng, R., and Senellart, P. (2014).
ProbTree: A query-efficient representation of probabilistic graphs.
In *Proc. BUDA*, Snowbird, USA.

Mitchell, J. C. (1983).
The implication problem for functional and inclusion dependencies.
Information and Control, 56(3).

References VI

Pratt-Hartmann, I. (2009).
Data-complexity of the two-variable fragment with counting quantifiers.
Inf. Comput., 207(8).

Rosati, R. (2006).
On the decidability and finite controllability of query processing in databases with incomplete information.
In *Proc. PODS*, Chicago, USA.

References VII

Talaika, A., Biega, J., Amarilli, A., and Suchanek, F. M. (2014).
Harvesting entities from the web using unique identifiers.
Preprint: http://a3nm.net/publications/talaika2015harvesting.pdf.
Submitted to WWW 2015.

Tang, R., Amarilli, A., Senellart, P., and Bressan, S. (2014).
Get a sample for a discount: Sampling-based XML data pricing.
In *Proc. DEXA*, Munich, Germany.