Uncertainty and Incompleteness

Antoine Amarilli

Télécom ParisTech; Institut Mines–Télécom; CNRS LTCI

May 19th, 2015

Motivation

- Traditional data management: data is correct and complete
- How realistic is this?

Motivation

- Traditional data management: data is correct and complete
- How realistic is this?
  - Noisy extractors
  - Untrustworthy contributors
  - Crappy crowd answers
  - Non-exhaustive sources

Motivation

- Traditional data management: data is correct and complete
- How realistic is this?
  - Noisy extractors
  - Untrustworthy contributors
  - Crappy crowd answers
  - Non-exhaustive sources

$\rightarrow$ How to adapt to uncertain and incomplete data?

Uncertain data (aka. closed world)

- The truth is a subset of our database
- Find out in which cases our query holds

Uncertain data (aka. closed world)

- The truth is a subset of our database
- Find out in which cases our query holds
  $\rightarrow$ Provenance!

Uncertain data (aka. closed world)

- The truth is a subset of our database
- Find out in which cases our query holds
  $\rightarrow$ Provenance!
- Quantitative models: probabilities on the database

Uncertain data (aka. closed world)

- The truth is a subset of our database
- Find out in which cases our query holds
  $\rightarrow$ Provenance!
- Quantitative models: probabilities on the database
  $\rightarrow$ #P-hardness lurks

Uncertain data (aka. closed world)

- The truth is a subset of our database
- Find out in which cases our query holds
  → Provenance!
- Quantitative models: probabilities on the database
  → #P-hardness lurks

→ Our idea: show tractability for restricted databases

Uncertain data (aka. closed world)

- The truth is a subset of our database
- Find out in which cases our query holds
  → Provenance!
- Quantitative models: probabilities on the database
  → #P-hardness lurks

→ Our idea: show tractability for restricted databases

Provenance Circuits for Trees and Treelike Instances

Antoine Amarilli[1], Pierre Bourhis[2], and Pierre Senellart[1,3]

[1] Institut Mines-Télécom; Télécom ParisTech; CNRS LTCI
first.last@telecom-paristech.fr
[2] CNRS CRIStAL; Université Lille 1; INRIA Lille
pierre.bourhis@univ-lille1.fr
[3] National University of Singapore; CNRS IPAL

Abstract. Query evaluation in monadic second-order logic is known to be tractable on trees and treelike instances, even though it is hard for arbitrary instances. This

Incomplete data (aka. open world)

- The truth is a superset of our database
- Harder to formalize...

Incomplete data (aka. open world)

- The truth is a superset of our database
- Harder to formalize…
  - → Impose logical constraints
  - → Ask whether the query is certain

Incomplete data (aka. open world)

- The truth is a superset of our database
- Harder to formalize...
  - → Impose logical constraints
  - → Ask whether the query is certain

→ Idea 1: combine existing approaches (description logics, existential rules)

Incomplete data (aka. open world)

- The truth is a superset of our database
- Harder to formalize...
  - → Impose logical constraints
  - → Ask whether the query is certain

→ Idea 1: combine existing approaches (description logics, existential rules)

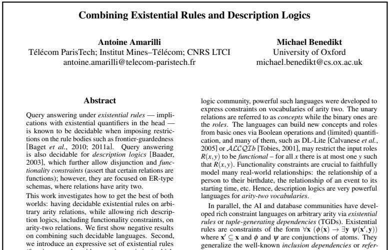
→ Idea 2: what about assuming finiteness?

Incomplete data (aka. open world)

- The truth is a superset of our database
- Harder to formalize...
  - → Impose logical constraints
  - → Ask whether the query is certain

→ Idea 1: combine existing approaches (description logics, existential rules)

→ Idea 2: what about assuming finiteness?



Combining Existential Rules and Description Logics

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

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

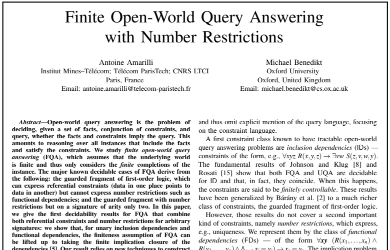
Abstract

Query answering under *existential rules* — implications with existential quantifiers in the head — is known to be decidable when imposing restrictions on the rule bodies such as frontier-guardedness [Baget *et al.*, 2010; 2011a]. Query answering is also decidable for *description logics* [Baader, 2003], which further allow disjunction and *functionality constraints* (assert that certain relations are functions); however, they are focused on ER-type schemas, where relations have arity two.

This work investigates how to get the best of both worlds: having decidable existential rules on arbitrary arity relations, while allowing rich description logics, including functionality constraints, on arity-two relations. We first show negative results on combining such decidable languages. Second, we introduce an expressive set of existential rules

logic community, powerful such languages were developed to express constraints on vocabularies of arity two. The unary relations are referred to as *concepts* while the binary ones are the *roles*. The languages can build new concepts and roles from basic ones via Boolean operations and (limited) quantification, and many of them, such as DL-Lite [Calvanese *et al.*, 2005] or $\mathcal{ALCQIb}$ [Tobies, 2001], may restrict the input roles $R(x,y)$ to be *functional* – for all x there is at most one y such that $R(x,y)$. Functionality constraints are crucial to faithfully model many real-world relationships: the relationship of a person to their birthdate, the relationship of an event to its starting time, etc. Hence, description logics are very powerful languages for *arity-two vocabularies*.

In parallel, the AI and database communities have developed rich constraint languages on arbitrary arity via *existential rules* or *tuple-generating dependencies* (TGDs). Existential rules are constraints of the form $\forall \mathbf{x}\ (\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}', \mathbf{y}))$ where $\mathbf{x}' \subseteq \mathbf{x}$ and ϕ and ψ are conjunctions of atoms. They generalize the well-known *inclusion dependencies* or *refer-*



Finite Open-World Query Answering with Number Restrictions

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

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

***Abstract*—Open-world query answering is the problem of deciding, given a set of facts, conjunction of constraints, and query, whether the facts and constraints imply the query. This amounts to reasoning over all instances that include the facts and satisfy the constraints. We study *finite open-world query answering* (FQA), which assumes that the underlying world is finite and thus only considers the *finite* completions of the instance. The major known decidable cases of 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 cannot express 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: we show that, for unary inclusion dependencies and functional dependencies, the finiteness assumption of FQA can be lifted up to taking the finite implication closure of the dependencies [5]. Our result relies on new techniques to construct**

and thus omit explicit mention of the query language, focusing on the constraint language.

A first constraint class known to have tractable open-world query answering problems are *inclusion dependencies* (IDs) — constraints of the form, e.g., $\forall xyz\ R(x,y,z) \rightarrow \exists vw\ S(z,v,w,y)$. The fundamental results of Johnson and Klug [8] and Rosati [15] show that both FQA and UQA are decidable for ID and that, in fact, they coincide. When this happens, the constraints are said to be *finitely controllable*. These results have been generalized by Bárány et al. [2] to a much richer class of constraints, the guarded fragment of first-order logic.

However, those results do not cover a second important kind of constraints, namely *number restrictions*, which express, e.g., uniqueness. We represent them by the class of *functional dependencies* (FDs) — of the form $\forall \boldsymbol{xy}\ (R(x_1, \ldots, x_n) \wedge$