Ontology-Based Query Answering
Overview and Relevant Work

Michaël Thomazo

Inria, Université Paris-Saclay
LIX, École Polytechnique, Université Paris-Saclay

WebClaimExplain – June 2nd, 2017
Is Semantics Needed?

Figure 2: Searching Data.gov for Natural Disaster Data Sets.
Benefits of Ontologies

- dealing with incompleteness of the data
- hiding even more the specifics of data storage
- using a vocabulary that is familiar to the user
The need for reasoning

Query answering needs explicit and implicit data!

- Materialization-based query answering
- Reformulation-based query answering
- Hybrids of the above: combined approaches
Materialization-based query answering

Materialization-based query answering involves the process of answering queries by precomputing and storing the results of those queries in a database system. This approach aims to speed up query processing by avoiding the need to compute the query results every time it is executed.

The process can be illustrated with the following diagram:

- **ontology**: The domain knowledge represented as a graph.
- **G**: The initial graph.
- **G∞**: The graph model used as a virtual table.
- **query q**: The query used to retrieve information from the graph model.
- **answer**: The result of the query.

The diagram shows how the query `q(G∞)` can be computed using a RDBMS. However, this method requires time to compute and space to store the graph model. It is not suitable for high update rates, as both the data and schema triples can change frequently, impacting the efficiency and performance of the system.

Courtesy of I. Manolescu
Materialization-based query answering

- $q(G^\infty)$ can be computed using an RDBMS
- $G^\infty$ needs time to be computed and space to be stored
- Not suitable for high update rate (data and/or schema triples)

Courtesy of I. Manolescu
Reformulation-based query answering

$\text{ontology}$

$G$

$\text{query } q$

$\text{query } q^{\text{ref}}$

$\text{answer}$

$\text{query } q^{\text{ref}}$ (of $G$) can be evaluated using an RDBMS.

Robust to updates

Reformulated queries are complex, thus costly to evaluate.

Courtesy of I. Manolescu
Reformulation-based query answering

$q^{\text{ref}}(G)$ can be evaluated using an RDBMS
- Robust to updates
- Reformulated queries are complex, thus costly to evaluate
Ontology Mediated Query Answering

- **Data**: Professor(Alice), Reviewer(Alice)
- **Query**: $\exists x \exists y \text{ Teacher}(x) \land \text{ reviews}(x, y)$
Ontology Mediated Query Answering

- **Data**: Professor(Alice), Reviewer(Alice)
- **Query**: $\exists x \exists y \text{ Teacher}(x) \land \text{ reviews}(x, y)$
- **Ontology (semantics)**:
  - $\forall x \text{ Reviewer}(x) \rightarrow \exists y \text{ reviews}(x, y)$
  - $\forall x \text{ Professor}(x) \rightarrow \text{ Teacher}(x)$
Ontology Mediated Query Answering

- **Data**: Professor(Alice), Reviewer(Alice)
- **Query**: $\exists x \ \exists y \ \text{Teacher}(x) \land \text{reviews}(x, y)$
- **Ontology (semantics)**:
  - $\forall x \ \text{Reviewer}(x) \rightarrow \exists y \ \text{reviews}(x, y)$
  - $\forall x \ \text{Professor}(x) \rightarrow \text{Teacher}(x)$

**Materialization (chase)**

Professor(Alice)
Reviewer(Alice)
Teacher(Alice)
$\exists y_1 \ \text{reviews}(\text{Alice}, y_1)$
Ontology Mediated Query Answering

- **Data**: Professor(Alice), Reviewer(Alice)
- **Query**: \( \exists x \exists y \, \text{Teacher}(x) \land \text{reviews}(x, y) \)
- **Ontology (semantics)**:
  - \( \forall x \, \text{Reviewer}(x) \rightarrow \exists y \, \text{reviews}(x, y) \)
  - \( \forall x \, \text{Professor}(x) \rightarrow \text{Teacher}(x) \)

**Materialization (chase)**
- Professor(Alice)
- Reviewer(Alice)
- Teacher(Alice)
- \( \exists y_1 \, \text{reviews}(Alice, y_1) \)

**Query Rewriting**
- \( \exists x \exists y \, \text{Teacher}(x) \land \text{reviews}(x, y) \)
- \( \exists x \, \text{Professor}(x) \land \text{Reviewer}(x) \)
- \( \exists x \, \text{Teacher}(x) \land \text{Reviewer}(x) \)
- \( \exists x \exists y \, \text{Professor}(x) \land \text{reviews}(x, y) \)
Formalization of the Problem

- Input: a set of ground atoms $I$, a set of existential rules (or a description logic) $\mathcal{R}$, a (Boolean) conjunctive query $q$
- Output: yes if and only if $I, \mathcal{R} \models q$
Formalization of the Problem

- Input: a set of ground atoms $I$, a set of existential rules (or a description logic) $R$, a (Boolean) conjunctive query $q$
- Output: yes if and only if $I, R \models q$

Existential Rule

An existential rule (or TGD) is a formula of the shape:

$$\forall x \forall y. [B(x, y) \rightarrow \exists z. H(y, z)],$$

- $B$ and $H$ are non-empty conjunctions of atoms on variables
- $x, y$ and $z$ are pairwise disjoint
Goal of the Talk

► incomplete...
► highly subjective...
► selection of topics, past, present and future
Problem 1: Does the Chase Terminate?

- it may not terminate:
  - \( I = \{ \text{Human}(Alice) \} \)
  - \( R = \{ \text{Human}(x) \rightarrow \text{hasParent}(x, y) \land \text{Human}(y) \} \)
Problem 1: Does the Chase Terminate?

- it may not terminate:
  - \( I = \{ \text{Human}(Alice) \} \)
  - \( R = \{ \text{Human}(x) \rightarrow \text{hasParent}(x, y) \land \text{Human}(y) \} \)
- checking if the chase w.r.t. \( R \) terminates on some/all instance is undecidable
- acyclicity based conditions have been proposed to ensure termination/non-termination
Problem 1: Does the Chase Terminate?

► it may not terminate:
  ► $I = \{\text{Human}(Alice)\}$
  ► $\mathcal{R} = \{\text{Human}(x) \rightarrow \text{hasParent}(x, y) \land \text{Human}(y)\}$

► checking if the chase w.r.t $\mathcal{R}$ terminates on some/all instance is undecidable

► acyclicity based conditions have been proposed to ensure termination/non-termination

References:

► Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies, Cuenca Grau et al., JAIR 2013

► Detecting Chase (Non)Termination for Existential Rules with Disjunctions, Carral et al., IJCAI 2017
Problem 2: Is there a Rewriting of $q$ in a Language $\mathcal{L}$? (1)

Given $q$ and $\mathcal{R}$, given a query language $\mathcal{L}$, does it exist $q' \in \mathcal{L}$ such that for all instance $I$,

$$I, \mathcal{R} \models q \iff I \models q'.$$
Problem 2: Is there a Rewriting of $q$ in a Language $\mathcal{L}$? (1)

Given $q$ and $\mathcal{R}$, given a query language $\mathcal{L}$, does it exist $q' \in \mathcal{L}$ such that for all instance $I$,

$$I, \mathcal{R} \models q \iff I \models q'.$$

Several target languages have been proposed:

- UCQs
- first-order logic
- non-recursive Datalog
- Datalog
- ...
Problem 2: Is there a Rewriting of $q$ in a Language $\mathcal{L}$? (2)

- this is not always the case: transitivity rules do not play well with first-order logic
- checking the existence of a rewriting is usually undecidable
- sufficient conditions have been proposed
- the size of generated rewritings has been studied

References:
- Sound, complete and minimal UCQ-rewriting for existential rules, König et al., SWJ 2015
- The price of query rewriting in ontology-based data access, Gottlob et al., AIJ 2014
Problem 2: Is there a Rewriting of $q$ in a Language $\mathcal{L}$? (2)

- this is not always the case: transitivity rules do not play well with first-order logic
- checking the existence of a rewriting is usually undecidable
- sufficient conditions have been proposed
- the size of generated rewritings has been studied

References:
- Sound, complete and minimal UCQ-rewriting for existential rules, König et al., SWJ 2015
- The price of query rewriting in ontology-based data access, Gottlob et al., AIJ 2014
Problem 3: Towards more Expressive Query Languages

- CQs are basic
- extension with aggregation
- extension with restricted form of recursivity (for instance, RPQs or CRPQs)

References:
- Complexity of Answering Counting Aggregate Queries over DL-Lite, Kostylev et al., DL 2013
- Answering Conjunctive Regular Path Queries over Guarded Existential Rules, Baget et al., IJCAI 2017
Problem 3: Towards more Expressive Query Languages

- CQs are basic
- extension with aggregation
- extension with restricted form of recursivity (for instance, RPQs or CRPQs)

References:
- Complexity of Answering Counting Aggregate Queries over DL-Lite, Kostylev et al., DL 2013
- Answering Conjunctive Regular Path Queries over Guarded Existential Rules, Baget et al., IJCAI 2017
Problem A: Optimization of Query Evaluation (1)

- the size of generated rewritings has been studied
- it does not tell much on the efficiency of query evaluation
- even small positive existential first-order rewritings are not easy to evaluate
- cost-based optimization of queries generated by rewriters is not a closed topic
Problem A: Optimization of Query Evaluation (2)

How come that current optimizers are not already efficient enough?
Problem A: Optimization of Query Evaluation (2)

How come that current optimizers are not already efficient enough?
A comment in Postgres optimizer code:

    /* we stop as soon as we hit a non-AND item */
Problem A: Optimization of Query Evaluation (2)

How come that current optimizers are not already efficient enough? A comment in Postgres optimizer code:

```c
/* we stop as soon as we hit a non-AND item */
```

Optimizing through unions is **crucial** for the kind of queries we are faced with.

References:

- Optimizing Reformulation-based Query Answering in RDF, Bursztyn et al., EDBT 2015
Problem B: Consistent Query Answering (1)

- in presence of inconsistencies, FOL semantics is not interesting
  - everything is entailed
- alternative to FOL need to be studied to keep some robustness
- variety of semantics based on the notion of repair
  - most common: keeping maximum consistent subset of the data
  - modifications of the data are also sometimes allowed

References:

- Inconsistency-Tolerant Semantics for Description Logics, Lembo et al., RR 2010
- Inconsistency-Tolerant Querying of Description Logic Knowledge Bases, Bienvenu et al. RW 2016
Problem C: Temporal OBQA

- time is important for applications
- several ways to integrate it
- interactions between time and reasoning explode quickly

References:

- Temporalizing Ontology-Based Data Access, Baader et al., CADE 2013
- Temporalized $\mathcal{EL}$ Ontologies for Accessing Temporal Data: Complexity of Atomic Queries, Gutiérrez-Basulto et al., IJCAI 2016
Recap

Chase Termination

Rewritability

Query Languages

Query Optimization

Consistent Query Answering

Temporal Data and Ontologies
Recap

Chase Termination
Rewritability
Query Languages

Query Optimization
Consistent Query Answering
Temporal Data and Ontologies