Query Answering over Existential Rules

Sebastian Rudolph
TU Dresden

Tutorial @ ICCL summer school 2016, Rayong
Learning Outcomes and Prerequisites

A good understanding of:

- the fundamentals of query answering under existential rules
- the main concepts and techniques
- possible research directions

Basic knowledge of:

- first-order logic (syntax and semantics)
- databases (relational model)
Outline

- Classical Query Answering
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views
Outline

- Classical Query Answering
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views
Classical Query Answering

Data/Facts → Answers? → Query
Data / Facts

Relational Database

<table>
<thead>
<tr>
<th>parentOf</th>
<th>Male</th>
<th>Fem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
| ...      | ...  | ...

RDF (Semantic Web)

∃x( parentOf(A,B) ∧ parentOf(A,C) ∧ parentOf(C,x) ∧ F(A) ∧ M(B) ∧ M(x) )

Or in graphs / hypergraphs

Abstraction in First-Order Logic

Etc.
Some Notation

• Our basic vocabulary:
  o A countable set $C$ of constants - domain of a database
  o A countable set $N$ of (labeled) nulls - globally $\exists$-quantified variables
  o A countable set $V$ of (regular) variables - used in rule and queries

• A term is a constant, null or variable

• An atom has the form $P(t_1, \ldots, t_n)$ - $P$ is an $n$-ary predicate and $t_i$’s are terms
Queries

Typically expressed as formulae of some logic (the query language) with free variables.

A lot of options, tradeoff between expressivity and computational well-behavedness.

Popular Query Languages:

- **conjunctive queries (CQ)**
  ... and their unions (UCQ)
- **first-order logic (FOL)**
  Basis of SQL
- **Datalog**
  ...and its fragments
- **second-order logic (SOL)**
Conjunctive Queries

Example: «Find all x such that x is a female and has a child who is a female»

\[ \exists y \ (\text{Female}(x) \land \text{childOf}(x, y) \land \text{Female}(y)) \]

FOL formula

\[ Q(x) = \text{Female}(x), \text{childOf}(x, y), \text{Female}(y) \]

Common notation

\[ \text{ans}(x) \leftarrow \text{Female}(x), \text{childOf}(x, y), \text{Female}(y) \]

Datalog notation

\[ \text{SELECT} \ x \ \text{FROM} \ ... \ \text{WHERE} \ ... \]

SQL/SPARQL

Formally: A **CQ** Q has the form \( \exists x_{k+1},...,x_m \ A_1 \land ... \land A_p \) where \( A_1,...,A_p \) are atoms over the variables \( x_1,...,x_m \) and \( x_1 \ldots x_k \) are free variables (defining the answer part).

If \( k = 0 \), Q is a **Boolean CQ** (existentially closed conjunctive formula) then the answer can only be yes or no. CQ-Answering \( \equiv_{\text{LOGSPACE}} \) Boolean-CQ-Answering
Evaluating Boolean CQs over Data

Data:
\[ \exists x \ (\text{loves}(\text{bob}, \text{bob}) \land \text{hates}(\text{bob}, x) \land \text{hates}(\text{alice}, \text{bob})) \]

Query:
\[ \exists xyz \ (\text{loves}(x, y) \land \text{hates}(x, z) \land \text{hates}(y, z)) \]

homomorphism
Homomorphism

• Semantics of queries and existential rules definable via the key notion of homomorphism

• A substitution from a set of symbols $S$ to a set of symbols $T$ is a function $h : S \rightarrow T$ - $h$ is a set of mappings of the form $s \rightarrow t$, where $s \in S$ and $t \in T$

• A homomorphism from a set of atoms $A$ to a set of atoms $B$ is a substitution $h : C \cup N \cup V \rightarrow C \cup N \cup V$ such that:
  
  (i) $t \in C \Rightarrow h(t) = t$ - unique name assumption
  
  (ii) $P(t_1, \ldots, t_n) \in A \Rightarrow h(P(t_1, \ldots, t_n)) = P(h(t_1), \ldots, h(t_n)) \in B$

• Can be naturally extended to conjunctions of atoms
Exercise: Find the Homomorphisms

\[ \varphi_1 = P(X,Y) \land P(Y,Z) \land P(Z,X) \]

\[ \varphi_2 = P(X,X) \]

\[ \varphi_3 = P(X,Y) \land P(Y,X) \land P(Y,Y) \]

\[ \varphi_4 = P(X,Y) \land P(Y,X) \]

\[ \varphi_5 = P(X,Y) \land P(Y,Z) \land P(Z,W) \]
Exercise: Find the Homomorphisms

\[ \varphi_5 = P(X,Y) \land P(Y,Z) \land P(Z,W) \]

\[ \varphi_1 = P(X,Y) \land P(Y,Z) \land P(Z,X) \]

\[ \varphi_4 = P(X,Y) \land P(Y,X) \]

\[ \varphi_2 = P(X,X) \]

\[ \varphi_3 = P(X,Y) \land P(Y,X) \land P(Y,Y) \]
Outline

- Classical Query Answering
- Ontological Query Answering: Two Views
  - KR View: KB Rewriting into Nice Models
    - Finite models through Acyclicity
    - Bounded-treewidth Models through Guardedness
    - Joining Acyclicity and Guardedness
    - Algorithmic Aspects
  - DB View: Query rewriting
  - Mixing the Views
Why Ontological Query Answering?

- vocabulary of data and query may not coincide
  (→ information exchange)
- databases may be incomplete
- some information may only be obtained when factoring in background knowledge
Views on Ontological Query Answering

The Plain View:

\[ D \models_{\Sigma} Q \]
Views on Ontological Query Answering

The Knowledge Representation View:

\[ D \land \Sigma \models Q \]
Views on Ontological Query Answering

The Database View:

\[ D \models (\land \Sigma) \Rightarrow Q \]
Our Main Showcase:

Data → Ontology → Query
Our Main Showcase:

Data → Existential Rules ← Boolean Conjunctive Query
Existential Rules

∀X ∀Y ( B[X, Y] → ∃Z H[X, Z] )

X, Y, Z : tuples of variables
Any conjunction of atoms (on variables and constants)

∀x ∀y (siblingOf(x,y) → ∃ z (parentOf(z,x) ∧ parentOf(z,y)))

Simplified form: siblingOf(x,y) → parentOf(z,x) ∧ parentOf(z,y)

- Same as Tuple Generating Dependencies (TGDs)
- See also Datalog+/-
- Same as the logical translation of Conceptual Graph rules
- Generalize lightweight DLs used for OBQA
Semantics of Existential Rules

• A database $D$ is a model of the rule
  \[ \sigma = \forall X \forall Y (B[X,Y] \rightarrow \exists Z H[X,Z]) \]
  written as $D \models \sigma$, if the following holds:
  whenever there exists a homomorphism $h$ such that $h(B[X,Y]) \subseteq D$,
  then there exists $g \supseteq h_{|X}$ such that $g(H[X,Z]) \subseteq D$

  \[ \{ t \rightarrow h(t) \mid t \in X \} \quad \text{– the restriction of } h \text{ to } X \]

• Given a set $\Sigma$ of existential rules, $D$ is a model of $\Sigma$, written as $D \models \Sigma$,
  if the following holds: for each $\sigma \in \Sigma$, $D \models \sigma$

• $D \models \Sigma$ iff $D$ is a model of the first-order theory $\bigwedge_{\sigma \in \Sigma} \sigma$
Existential Rules vs. DLs

Existential rules and DLs rely on first-order semantics - comparable formalisms

**DL-Lite**: Popular family of DLs - at the basis of the OWL 2 QL profile of OWL

<table>
<thead>
<tr>
<th>DL-Lite Axioms</th>
<th>Existential Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \subseteq B )</td>
<td>( \forall x ,(A(x) \rightarrow B(x)) )</td>
</tr>
<tr>
<td>( A \subseteq \exists R )</td>
<td>( \forall x ,(A(x) \rightarrow \exists y ,R(x,y)) )</td>
</tr>
<tr>
<td>( \exists R \subseteq A )</td>
<td>( \forall x \forall y ,(R(x,y) \rightarrow A(x)) )</td>
</tr>
<tr>
<td>( \exists R \subseteq \exists P )</td>
<td>( \forall x \forall y ,(R(x,y) \rightarrow \exists z ,P(x,z)) )</td>
</tr>
<tr>
<td>( A \subseteq \exists R.B )</td>
<td>( \forall x ,(A(x) \rightarrow \exists y ,(R(x,y) \land B(y))) )</td>
</tr>
<tr>
<td>( R \subseteq P )</td>
<td>( \forall x \forall y ,(R(x,y) \rightarrow P(x,y)) )</td>
</tr>
<tr>
<td>( A \subseteq \neg B )</td>
<td>( \forall x ,(A(x) \land B(x) \rightarrow \bot) )</td>
</tr>
</tbody>
</table>
Existential Rules vs. DLs

Existential rules and DLs rely on first-order semantics - comparable formalisms

**EL**: Popular DL for biological applications - at the basis of OWL 2 EL profile

<table>
<thead>
<tr>
<th>EL Axioms</th>
<th>Existential Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \sqsubseteq B$</td>
<td>$\forall x \ (A(x) \rightarrow B(x))$</td>
</tr>
<tr>
<td>$A \sqcap B \sqsubseteq C$</td>
<td>$\forall x \ (A(x) \land B(x) \rightarrow C(x))$</td>
</tr>
<tr>
<td>$A \sqsubseteq \exists R.B$</td>
<td>$\forall x \ (A(x) \rightarrow \exists y \ (R(x,y) \land B(y)))$</td>
</tr>
<tr>
<td>$\exists R.B \sqsubseteq A$</td>
<td>$\forall x \forall y \ (R(x,y) \land B(y) \rightarrow A(x))$</td>
</tr>
</tbody>
</table>
Existential Rules vs. DLs

• Several Horn DLs (without disjunction) can be expressed via existential rules

• But, existential rules can express more

\[ \forall x \forall y (\text{siblingsOf}(x, y) \rightarrow \exists z (\text{parentOf}(z, x) \land \text{parentOf}(z, y))) \]

• Higher arity predicates allow for more flexibility
  o Direct translation of database relations
  o Adding contextual information is easy (provenance, trust, etc.)
Outline

- Classical Query Answering
- Ontological Query Answering: Two Views
  - KR View: KB Rewriting into Nice Models
    - Finite models through Acyclicity
    - Bounded-treewidth Models through Guardedness
    - Joining Acyclicity and Guardedness
    - Algorithmic Aspects
  - DB View: Query rewriting
  - Mixing the Views
rewrite knowledge base into a representation that is easier to query, by making the structure of its models more explicit

- for FOL: semantic tableau
  (also used for Description Logics)
- for Existential Rules: the chase
The Two Dimensions of Infinity

Consider the database $D$, and the set of existential rules $\Sigma$

$D \land \Sigma$ admits infinitely many models, and each one may be of infinite size
The Two Dimensions of Infinity

\[ D = \{P(c)\} \quad \Sigma = \{\forall x (P(x) \rightarrow \exists y (R(x,y) \land P(y)))\} \]

\[ \text{model of } D \land \Sigma \]

\[ P(c) \quad R(c,c) \]
\[ P(c) \quad R(c,z_1) \quad P(z_1) \quad R(z_1,z_1) \]
\[ P(c) \quad R(c,z_1) \quad P(z_1) \quad R(z_1,z_2) \quad P(z_2) \quad R(z_2,z_2) \]
\[ \ldots \]
\[ P(c) \quad R(c,z_1) \quad P(z_1) \quad R(z_1,z_2) \quad P(z_2) \quad R(z_2,z_3) \]
\[ \ldots \]
\[ z_1, z_2, z_3, \ldots \text{ are nulls of } N \]
Taming the First Dimension of Infinity

\[ D = \{ P(c) \} \quad \Sigma = \{ \forall x (P(x) \rightarrow \exists y (R(x,y) \wedge P(y))) \} \]

Key Idea: Focus on a representative, a model that is as general as possible.
Universal (also: Canonical) Models

An instance $U$ is a universal model of $D \land \Sigma$ if the following holds:

1. $U$ is a model of $D \land \Sigma$

2. $\forall J \in \text{models}(D \land \Sigma)$, there exists a homomorphism $h_J$ such that $h_J(U)$
Query Answering via Universal Models

Theorem: \( D \land \Sigma \models Q \) iff \( U \models Q \), where \( U \) is a universal model of \( D \land \Sigma \)

Proof: 

(\( \Rightarrow \)) Trivial since, for every \( J \in \text{models}(D \land \Sigma) \), \( J \models Q \)

(\( \Leftarrow \)) By exploiting the universality of \( U \)

\[
\forall J \in \text{models}(D \land \Sigma), \exists h_J \text{ such that } h_J(g(Q)) \subseteq J \\
\Rightarrow \forall J \in \text{models}(D \land \Sigma), J \models Q \\
\Rightarrow D \land \Sigma \models Q
\]
The Chase Procedure

• **Fundamental algorithmic tool** used in databases

• It has been applied to a **wide range of problems**:
  - Checking containment of queries under constraints
  - Computing data exchange solutions
  - Computing certain answers in data integration settings
  - …

… what’s the reason for the ubiquity of the chase in databases?

it constructs universal models
Value Invention
(Generation of Fresh Existentials)

\[ R = \forall x \forall y \ (\text{siblingOf}(x,y) \rightarrow \exists z \ (\text{parentOf}(z,x) \land \text{parentOf}(z,y))) \]

\[ D = \text{siblingOf}(A,B) \]

A rule \( \text{body} \rightarrow \text{head} \) is applicable to a database \( D \) if there is a homomorphism \( h \) from \( \text{body} \) to \( D \)

The resulting database is \( D' = D \cup h(\text{head}) \)
[with new names for existential variables of \( \text{head} \)]

\[ D' = \exists z_0 \ (\text{siblingOf}(A,B) \land \text{parentOf}(z_0,A) \land \text{parentOf}(z_0,B)) \]
The Chase

\[ t(x) \rightarrow s(x,y) \land t(y) \]

\[ t(x) \land t(y) \rightarrow r(x,y) \]

\[ D = \{ t(A) \} \]

\[ \sum \]

\[ \text{chase}(D, \Sigma) \]