Ontology Based Query Answering

The story so far

Magdalena Ortiz

Institute of Information Systems, Vienna University of Technology

May 22, 2013

Outline

1. Introduction

2. Conjunctive Queries
2.1 Tractable combined complexity

3. Regular path queries (RPQs) and their extensions

4. Conclusions

Graph Databases

In many application areas data can be naturally modeled using unary and binary relations only

- essentially, a graph database
- e.g., data on the web, RDF datasets
- In description logics (DLs), we call such a database an ABox.

Querying these databases is a relevant problem in many areas.

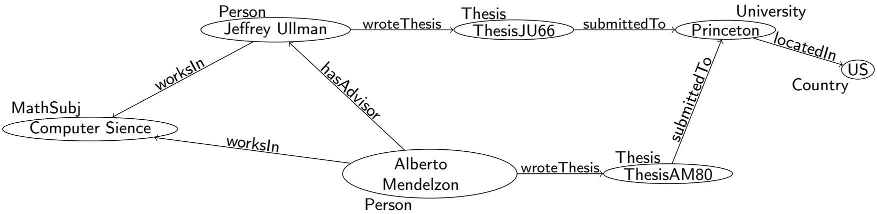
Graph Databases

In many application areas data can be naturally modeled using unary and binary relations only

- essentially, a graph database
- e.g., data on the web, RDF datasets
- In description logics (DLs), we call such a database an ABox.

  (and we call unary and binary relations concept and roles)

Querying these databases is a relevant problem in many areas.



Example graph database of the Mathematics Genealogy Project

Expressing Ontological Constraints

- Using a DL ontology (called a TBox), we can enrich the data with ontological constraints:
  - give a conceptual model of the application domain
  - provide an enriched vocabulary for querying

$$
\begin{array}{lrcl}
(1) & \exists \mathit{worksOn}.\{\mathit{CompSci}\} & \sqsubseteq & \mathit{CompScientist} \\
(2) & \mathit{PhD} & \sqsubseteq & \exists \mathit{hasAdvisor}.\mathit{PhD} \\
(3) & \mathit{PhD} & \sqsubseteq & \exists \mathit{wroteThesis}.\mathit{SubmittedThesis} \\
(4) & \exists \mathit{wroteThesis}.\mathit{SubmittedThesis} & \sqsubseteq & \mathit{PhD} \\
(5) & \mathit{Thesis} \sqcap \exists \mathit{submittedTo}.\mathit{University} & \sqsubseteq & \mathit{SubmittedThesis} \\
(6) & \mathit{SubmittedThesis} & \sqsubseteq & \mathit{Thesis} \sqcap \exists \mathit{submittedTo}.\mathit{University}
\end{array}
$$

Similarly to computer scientists, we can define logicians, physicists, b biologists, etc.

Ontology-based Query Answering (OBQA)

We can leverage the knowledge in the ontology when querying the data.

$$
\begin{aligned}
q_1(x,y) \leftarrow\, & \mathit{ComputerScientist}(x), \mathit{ComputerScientist}(y), \mathit{hasAdvisor}(x,y),\\
& \mathit{wroteThesis}(x,z), \mathit{wroteThesis}(y,z'),\\
& \mathit{submittedTo}(z,w), \mathit{submittedTo}(z',w)
\end{aligned}
$$

answers: $(\mathit{JeffreyUllman}, \mathit{AlbertoMendelzon})$

Note that the data does not include any instances of the concept *ComputerScientist*

Ontology-based Query Answering (OBQA)

We can leverage the knowledge in the ontology when querying the data.

$$
\begin{aligned}
q_2(x) \leftarrow & \mathit{ComputerScientist}(x), \mathit{hasAdvisor}(x, y), \\
& \mathit{wroteThesis}(y, z), \mathit{submittedTo}(z, w), \mathit{University}(z', w)
\end{aligned}
$$

answers: *AlbertoMendelzon*, *JeffreyUllman*

To get *JeffreyUllman* as an answer, we must map y, z, and w to anonymous objects whose existence is implied by the axioms.

Certain Answers

- In OBQA the data is seen as incomplete
- We are interested in the certain answers
- That is, the tuples of constants that are an answer to the query in every model of the data and the ontology

OBQA vs. OBDA

Ontology-based Data Access (OBDA) has become an important field

- Possibly many, heterogenous and distributed data sources
- An ontology provides a uniform conceptual model of the data
- Mappings are used to relate the data sources to the ontology
- Queries over the ontology must be pushed down to the sources

Our problem is simpler: plain ontological query evaluation

- we assume a unique data source over the language of the ontology
- direct access to the data, no mappings
- algorithms are applicable for OBDA

Scope of the talk

OBQA research has focused on answering two questions:

- how do we evaluate a query over an ontology?
- what is the computational cost of this evaluation?

Scope of the talk

OBQA research has focused on answering two questions:

- how do we evaluate a query over an ontology?
- what is the computational cost of this evaluation?

The answer depends on:

- the language of the query
- the language of the ontology

Here we survey some of the obtained results.

Most common combination: conjunctive queries and *DL-Lite*

Conjunctive Queries (CQs) and their unions

A *conjunctive query* is a formula of the form

$$q(\vec{x}) = \exists \vec{y}. A_1(\vec{y}_1) \wedge \ldots \wedge A_n(\vec{y}_n)$$

It can also be written as a rule

$$q(\vec{x}) \leftarrow A_1(\vec{y}_1), \ldots, A_n(\vec{y}_n)$$

- Equivalent to the plain Select-Project-Join fragment of SQL

A *union of conjunctive queries* (UCQ) is a disjunction of CQs.
It can be written as a set of rules.

DL-Lite

Ontological axioms in *DL-Lite*$_{core}$ have the following forms:

$$B \sqsubseteq C \qquad B \sqsubseteq \neg C$$

In *DL-Lite*$_{\mathcal{R}}$, we additionally allow

$$R \sqsubseteq S \qquad R \sqsubseteq \neg S$$

with

$$B, C ::= A \mid \exists R \quad \text{and} \quad R := r \mid r^-$$

where A is an atomic concept and r an atomic role name.

The Query Rewriting Approach

To answer a query in *DL-Lite*, we can incorporate all the ontological knowledge into the query:

Idea: (Calvanese et.al. 97)
Transform a query q and ontology $\mathcal{O}$ into a new query $q_{\mathcal{O}}$
such that, for every database $\mathcal{A}$
the answers to q over $\mathcal{A}$ and $\mathcal{O}$ coincide
with the answers to $q_{\mathcal{O}}$ over $\mathcal{A}$ alone

- we can directly evaluate $q_{\mathcal{O}}$ over the data using off-the-shelf RDBMSs
- If q is a CQ or UCQ, then $q_{\mathcal{O}}$ is a UCQ

$q_{\mathcal{O}}$ does not depend on $\mathcal{A}$: it shows that the data complexity of query answering in *DL-Lite* is not higher than for FO queries over standard DBs, that is, in AC_0

The PerfectRef algorithm

TBox $\mathcal{T}$: $$\begin{array}{lcl} B' & \sqsubseteq & B \\ \exists S & \sqsubseteq & A \end{array}$$

Query: $q \leftarrow A(x), R(x,y), B(y)$

The rewriting of q is the disjunction of: $$\begin{array}{l} A(x), R(x,y), B(y); \\ A(x), R(x,y), B'(y); \\ S(x,z), R(x,y), B(y); \\ S(x,z), R(x,y), B'(y); \end{array}$$

The rewriting algorithm is given as a set of rules that apply the axioms GCIs in(from right to left), unifying variables when possible:

$$\begin{array}{rccc} A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\ \exists P \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(x, \_), \ldots \\ \exists P^- \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, P(\_, x), \ldots \\ A \sqsubseteq \exists P & \ldots, P(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\ A \sqsubseteq \exists P^- & \ldots, P(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\ \exists P_1 \sqsubseteq \exists P_2 & \ldots, P_2(x, \_), \ldots & \rightsquigarrow & \ldots, P_1(x, \_), \ldots \\ P_1 \sqsubseteq P_2 & \ldots, P_2(x,y), \ldots & \rightsquigarrow & \ldots, P_1(x,y), \ldots \\ & \ldots & & \end{array}$$

where _ denotes a fresh unbound variable

Complexity of PerfectRef

- We saw it gives AC_0 data complexity
- In combined complexity, it gives an NP upper bound:
  - There are exponentially many rewritings
  - Each of them is of polynomial size
  - To verify a query answer, we can non-deterministically build the right rewriting
- In practice, it yields very large UCQs and does not scale easily to large queries
- Improvements have been proposed, e.g., optimizing the unification step, and rewriting into more succinct queries (positive existential FOL, non-recursive Datalog)

An alternative query rewriting

- It is well known that *DL-Lite* enjoys the canonical model property: there is one canonical model over which we can answer all queries
- In this model, the anonymous objects form tree-shaped structures

Idea:
Remove, in all possible ways, variables that can be mapped to anonymous objects, to obtain a query where all variables are mapped to known constants

Example

if $\mathcal{O} \models A \sqsubseteq \exists R$ and $\mathcal{O} \models \exists R^{-} \sqsubseteq B$

- $q \leftarrow\ R(x, y), B(y)$ is rewritten into $q' \leftarrow\ A(x)$
- $q \leftarrow\ R(x, y), R(z, y), C(z), B(y)$ is rewritten into $q' \leftarrow\ A(x), C(x)$

An alternative query rewriting (2)

- We obtain a UCQ by applying such a rewriting in all possible ways
- To answer it, it suffices to consider variables assignments to the constants (active domain)
- We still need to know all concepts that an object is an instance of (i.e., to close under the concept hierarchy)
  For DL-Lite$_{\mathcal{R}}$, the same apples to roles
- This can be done, e.g., building an SQL view for each atomic concept/role
- Same worst case complexity, but in practice, much smaller rewritings

$\mathcal{EL}$

$\mathcal{EL}$ is another prominent family of lightweight DLs, popular for life science ontologies

- In $\mathcal{EL}$, ontological axioms have the form

$$C \sqsubseteq D$$

  where

$$C, D ::= A \mid C \sqcap D \mid \exists r.C$$

  where A is an atomic concept and r an atomic role name.

- In $\mathcal{ELH}$, we additionally allow $r \sqsubseteq s$ for atomic roles r,s

Important: no inverse roles allowed!

$\mathcal{EL}$ and Query Rewriting

Can we use query rewriting to answer queries in $\mathcal{EL}$?

TBox $\mathcal{T}$: $\exists R.A \quad \sqsubseteq \quad A$

Query: $\quad q \quad \leftarrow \quad A(x)$

$\mathcal{EL}$ and Query Rewriting

Can we use query rewriting to answer queries in $\mathcal{EL}$?

TBox $\mathcal{T}$: $\exists R.A \quad \sqsubseteq \quad A$

Query: $q \quad \leftarrow \quad A(x)$

The rewriting of q is the disjunction of:

$$\begin{array}{l} A(x) \\ R(x, y_1), A(y_1) \\ R(x, y_1), R(y_1, y_2), A(y_2) \\ R(x, y_1), R(y_1, y_2), R(y_2, y_3), A(y_3) \\ \ldots \end{array}$$

$\mathcal{EL}$ and Query Rewriting

Can we use query rewriting to answer queries in $\mathcal{EL}$?

TBox $\mathcal{T}$: $\exists R.A \quad \sqsubseteq \quad A$

Query: $q \quad \leftarrow \quad A(x)$

The rewriting of q is the disjunction of:

$$
\begin{array}{l}
A(x) \\
R(x, y_1), A(y_1) \\
R(x, y_1), R(y_1, y_2), A(y_2) \\
R(x, y_1), R(y_1, y_2), R(y_2, y_3), A(y_3) \\
\ldots
\end{array}
$$

This can not be written as a finite SQL query!

It can be written as $R^*(x, y), A(y)$, but transitive closure is not FOL-expressible

$\mathcal{EL}$ and Query Rewriting

Can we use query rewriting to answer queries in $\mathcal{EL}$?

TBox $\mathcal{T}$: $\exists R.A \quad \sqsubseteq \quad A$

Query: $\quad q \quad \leftarrow \quad A(x)$

The rewriting of q is the disjunction of:

$$
\begin{array}{l}
A(x) \\
R(x, y_1), A(y_1) \\
R(x, y_1), R(y_1, y_2), A(y_2) \\
R(x, y_1), R(y_1, y_2), R(y_2, y_3), A(y_3) \\
\dots
\end{array}
$$

This can not be written as a finite SQL query!

It can be written as $R^*(x, y), A(y)$, but transitive closure is not FOL-expressible

> $\mathcal{EL}$ is P-hard in data complexity, hence we can not use the Query Rewriting approach as defined above

Query Rewriting beyond DL Lite

First-order rewritability fails for every DL beyond DL Lite

Query Rewriting beyond DL Lite

First-order rewritability fails for every DL beyond DL Lite

- Some rewritings into Datalog have been developed for the $\mathcal{EL}$ family
- For example, the alternative rewriting above works for $\mathcal{EL}$ and $\mathcal{ELH}$
  - The query is rewritten into a UCQ in the same way
  - But for closing under the concept/role hierarchy, we need recursion
  - Can be easily done with Datalog rules

  This works also for more expressive DLs (e.g., $\mathrm{Horn}\text{-}\mathcal{SHIQ}$)
- The *combined approach* of Lutz el.al. (2008) modifies both the data and the query, and then uses a standard RDBMS
- All these algorithms give NP upper bounds in combined complexity

The complexity of Query Answering in Lightweight DLs

	Combined complexity		Data complexity	
	IQs	**CQs**	IQs	**CQs**
Plain databases	in AC_0	NP-c	in AC_0	in AC_0
DL-Lite$_{(\mathcal{R})}$	NL-c	NP-c	in AC_0	in AC_0
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	NP-c	P-c	P-c

Here, IQs stands for *instance queries*, which are CQs with one atom. They have the same complexity as standard DL reasoning problems (satisfiability, subsumption, etc.)

The complexity of CQ answering in lightweight DLs is usually not higher than for plain DBs

CQ answering in expressive DLs

If we move beyond these lightweight DLs, the complexity increases significantly.

- $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ are the expressive DLs that underlie OWL-Lite and OWL-DL
- Standard reasoning in them is at least ExpTime-hard in combined complexity and coNP-hard in data
- For query answering, the complexity increases by an exponential
- Decidability of the problem remains open for $\mathcal{SHOIQ}$
- Their Horn (i.e., disjunction free) fragments are also ExpTime-hard, but
  - data complexity remains tractable
  - there is no blow-up in combined complexity for CQs

CQ answering in expressive DLs

	Combined complexity		Data complexity	
	IQs	**CQs**	IQs	**CQs**
Horn-$\mathcal{SHIQ}$, Horn-$\mathcal{SHOIQ}$	ExpTime-c	ExpTime-c	P-c	P-c
$\mathcal{SHIQ}$	ExpTime-c	2ExpTime-c	coNP-c	coNP-c
$\mathcal{SHOIQ}$	co-NExpTime-c	open	coNP-c	open

- For most expressive DLs, CQ answering is 2ExpTime-complete in combined complexity
- Many algorithms exist
- For example, one can use automata on infinite trees

Looking for tractable cases

- CQs over lightweight DLs are NP complete, just like for plain DBs
- Many classes of CQs are known to be tractable over plain DBs
- Most notable: acyclic queries
- What happens in OBQA? Can we transfer these results?

Looking for tractable cases

- CQs over lightweight DLs are NP complete, just like for plain DBs
- Many classes of CQs are known to be tractable over plain DBs
- Most notable: acyclic queries
- What happens in OBQA? Can we transfer these results?

Some negative results:

Tree-shaped queries over DL-Lite$_{\mathcal{R}}$ ontologies are NP hard (Kikot et al., 2011)

Looking for tractable cases

Positive results (Bienvenu,O.Šimkus,Xiao, IJCAI 13)

Claim

Let $\mathcal{O}$ be an ontology in DL-Lite, $\mathcal{EL}$ or $\mathcal{ELH}$, let $\mathcal{A}$ be an ABox and q be a query. We can construct in polynomial time an ABox $\mathcal{A}_{q,\mathcal{O}}$ such that the answers to q over $\mathcal{A}$ and $\mathcal{O}$ coincide with the answers to q over $\mathcal{A}_{q,\mathcal{O}}$.

All tractability results known from CQs transfer to these logics:

Corollary

Let $\mathcal{C}$ be any class of queries that can be answered in polynomial time for plain DBs. Then queries in $\mathcal{C}$ can be answered in polynomial time in DL-Lite, $\mathcal{EL}$ and $\mathcal{ELH}$.

For a class of *almost acyclic queries*, also a more practical rewriting algorithm

The combined complexity of restricted queries

For acyclic queries, queries of bounded tree-width, etc. we have:

Plain databases	in P
$\mathcal{EL}$	in P
$\mathcal{ELH}$	in P
DL-Lite	in P
DL-Lite$_\mathcal{R}$	NP-complete

For DL-Lite$_\mathcal{R}$ syntactic restrictions that make the problem tractable are known, and they seem to be realistic

Graph Databases and Regular Paths

- At the beginning of the talk, we were talking about graph databases
- But then, why are we talking about CQs?
- The fundamental languages for graph databases are regular path queries (RPQs) and their extensions: 2RPQs, CRPQs, C2RPQs. . .

Conjunctive 2-way Regular Path Queries

- In RPQs, binary atoms allow us to search for arbitrarily long paths between two individuals that comply to a regular expression.
- Called conjunctive, if we can have several atoms.
- Called 2-way if we can use both role names and their inverses.
- Sometimes, the *test* operator is used to search for explicit concept labels along the path

Conjunctive 2-way Regular Path Queries

- In RPQs, binary atoms allow us to search for arbitrarily long paths between two individuals that comply to a regular expression.
- Called conjunctive, if we can have several atoms.
- Called 2-way if we can use both role names and their inverses.
- Sometimes, the *test* operator is used to search for explicit concept labels along the path

This kind of path navigation is present in:

- XPath, the navigational core of the XQuery language for XML
- New versions of SPARQL, the query language for RDF:
  - extensions like nSPARQL and vSPARQL
  - path properties in SPARQL 1.1 (possibly the next standard)

Examples of Regular Path Queries

The following atom navigates a chain of advisors that are computer scientists or logicians, until a biologist is reached.

$$(\mathit{hasAdvisor} \circ \mathit{Logician}? \cup \mathit{hasAdvisor} \circ \mathit{CompScientist}?)^* \circ \mathit{hasAdvisor} \circ \mathit{Biologist}?(x, y)$$

Using a conjunction of atoms, we can find scientists x that have a student y and a co-author z sharing such an advisor u:

$$\begin{aligned} &\mathit{hasAdvisor}^-(x, y), \quad \mathit{hasCoauthor}(x, z), \\ &(\mathit{hasAdvisor} \circ \mathit{Logician}? \cup \mathit{hasAdvisor} \circ \mathit{CompScientist}?)^* \circ \\ &\qquad \mathit{hasAdvisor} \circ \mathit{Biologist}?(y, u), \\ &(\mathit{hasAdvisor} \circ \mathit{Logician}? \cup \mathit{hasAdvisor} \circ \mathit{CompScientist}?)^* \circ \\ &\qquad \mathit{hasAdvisor} \circ \mathit{Biologist}?(z, u) \end{aligned}$$

Regular Path Queries in DLs

- For expressive DLs, their complexity coincides with CQs
- Upper bounds obtained using automata on infinite trees
- If the DL is expressive enough, then the difference in expressive power is not so significant

For the lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families, they remained unexplored until recently.

Regular Path Queries in DLs

- For expressive DLs, their complexity coincides with CQs
- Upper bounds obtained using automata on infinite trees
- If the DL is expressive enough, then the difference in expressive power is not so significant

For the lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families, they remained unexplored until recently.

Complexity of RPQs

(Bienvenu,O.Šimkus, IJCAI 13)

	Combined complexity		Data complexity	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$	NL-c	NP-c	NL-c	NL-c
DL-Lite$_{(\mathcal{R})}$	P-c†	PSpace-c	NL-c	NL-c
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	PSpace-c	P-c	P-c

harder than plain CQs   harder also than RPQs over graph DBs

	IQs	CQs	IQs	CQs
Plain databases	in AC_0	NP-c	in AC_0	in AC_0
DL-Lite$_{(\mathcal{R})}$	NL-c	NP-c	in AC_0	in AC_0
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	NP-c	P-c	P-c

All hardness results hold for (C)RPQs (1-way), and for NFA and reg.exp. representation of atoms. Only † uses NFAs and needs 2RPQs for DL-Lite.

Lower Bounds

	Combined complexity		**Data complexity**	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$	NL-c	NP-c	NL-c	NL-c
DL-Lite$_{(\mathcal{R})}$	P-c	PSpace-c	NL-c	NL-c
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	PSpace-c	P-c	P-c

Lower Bounds

	Combined complexity		**Data complexity**	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$	NL-c	NP-c	NL-c	NL-c
DL-Lite$_{(\mathcal{R})}$	P-c	PSpace-c	NL-c	NL-c
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	PSpace-c	P-c	P-c

Lower Bounds

	Combined complexity		Data complexity	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$	**NL-c**		**NL-c** [1]	**NL-c**
DL-Lite$_{(\mathcal{R})}$			**NL-c**	**NL-c**
$\mathcal{EL}$, $\mathcal{ELH}$				

1. Known for plain RPQs over plain graph DBs: simple reduction from directed graph reachability using a query $r^*(x, y)$

Lower Bounds

	Combined complexity		Data complexity	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\text{RDFS}}$				
DL-Lite$_{(\mathcal{R})}$	**P-c** [2]			
$\mathcal{EL}$, $\mathcal{ELH}$				

1. Known for plain RPQs over plain graph DBs: simple reduction from directed graph reachability using a query $r^*(x, y)$
2. Reduction from entailment from a set of Horn clauses. Assumes NFA representation, and needs either role inclusions or 2RPQs

Lower Bounds

	Combined complexity		Data complexity	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$				
DL-Lite$_{(\mathcal{R})}$				
$\mathcal{EL}$, $\mathcal{ELH}$				

1. Known for plain RPQs over plain graph DBs: simple reduction from directed graph reachability using a query $r^*(x, y)$
2. Reduction from entailment from a set of Horn clauses. Assumes NFA representation, and needs either role inclusions or 2RPQs

Lower Bounds

	Combined complexity		Data complexity	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$				
DL-Lite$_{(\mathcal{R})}$		**PSpace-c** [3]		
$\mathcal{EL}$, $\mathcal{ELH}$		**PSpace-c**		

1. Known for plain RPQs over plain graph DBs: simple reduction from directed graph reachability using a query $r^*(x, y)$
2. Reduction from entailment from a set of Horn clauses. Assumes NFA representation, and needs either role inclusions or 2RPQs
3. Reduction from emptiness of the intersection of regular languages $L_1, \ldots, L_n$ over alphabet Σ

Lower Bounds

	Combined complexity		Data complexity	
	(2)RPQs	C(2)RPQs	(2)RPQs	C(2)RPQs
Plain databases DL-Lite$_{\mathsf{RDFS}}$	NL-c	NP-c	NL-c [1]	NL-c
DL-Lite$_{(\mathcal{R})}$	P-c [2]	PSpace-c [3]	NL-c	NL-c
$\mathcal{EL}$, $\mathcal{ELH}$	P-c	PSpace-c	P-c	P-c

1. Known for plain RPQs over plain graph DBs: simple reduction from directed graph reachability using a query $r^*(x, y)$
2. Reduction from entailment from a set of Horn clauses. Assumes NFA representation, and needs either role inclusions or 2RPQs
3. Reduction from emptiness of the intersection of regular languages $L_1, \ldots, L_n$ over alphabet Σ

PSpace-hardness

Reduction from emptiness of the intersection of regular languages $L_1, \ldots, L_n$ over alphabet Σ:

- ABox $\mathcal{A} = \{A(a)\}$

- Use TBox to generate all words in Σ^*
  - For DL-Lite

$$\mathcal{T} = \{A \sqsubseteq \exists r \mid r \in \Sigma\} \cup \{\exists r^- \sqsubseteq \exists s \mid r, s \in \Sigma\}$$

  - For $\mathcal{EL}$

$$\mathcal{T} = \{A \sqsubseteq \exists r.A \mid r \in \Sigma\}$$

  This generates a tree rooted at a, with one branch for each word in for every word $w \in \Sigma^*$

- the query $q \leftarrow \exists x\ L_1(a, x) \wedge \ldots \wedge L_n(a, x)$ is true iff $L_1 \cap \ldots \cap L_n$ is non-empty.

Membership Results

The membership results are shown via an adaptation of the alternative rewriting above

Important observations:

1. a binary atom might not be fully consumed during one rewriting step, so we need to keep track of what still needs to be satisfied
2. the path connecting variables x, y in an atom may be very complex, e.g., it can go deeper than x and y into the anonymous part and come back up several times

Membership Results

The membership results are shown via an adaptation of the alternative rewriting above

Important observations:

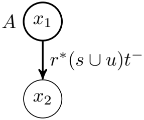
1. a binary atom might not be fully consumed during one rewriting step, so we need to keep track of what still needs to be satisfied
2. the path connecting variables x, y in an atom may be very complex, e.g., it can go deeper than x and y into the anonymous part and come back up several times

Our solution:

- guess a decomposition of a binary atom $\alpha(y, x)$ into pieces of the path which lie above x and those which lie below x (loops)
- in addition to ensuring that x's concept atoms hold, we must also check that the loops below x are implied
- for the remaining binary atoms, must "subtract" the edge from y to x

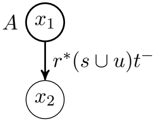
One rewriting step - complex binary atoms

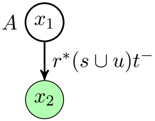
$$q \leftarrow r^*(s \cup u)t^-(x_1, x_2)$$



One rewriting step - complex binary atoms

In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$

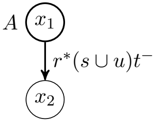


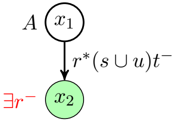


1. Select the non-distinguished variable x_2 as leaf.

One rewriting step - complex binary atoms

In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$





1. Select the non-distinguished variable x_2 as leaf.
2. Choose a "type" for x_2 - say $\exists r^-$.

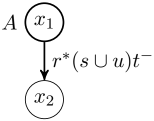
One rewriting step - complex binary atoms

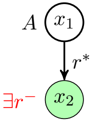
In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$

1. Select the non-distinguished variable x_2 as leaf.
2. Choose a "type" for x_2 - say $\exists r^-$.
3. Decomposition into two pieces: $r^*(x_1, x_2)$, $(s \cup u)t^-(x_2, x_2)$

One rewriting step - complex binary atoms

In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$





1. Select the non-distinguished variable x_2 as leaf.
2. Choose a "type" for x_2 - say $\exists r^-$.
3. Decomposition into two pieces: $r^*(x_1, x_2)$, $(s \cup u)t^-(x_2, x_2)$
4. Drop second atom as $\exists r^-$ ensures st^- loop

One rewriting step - complex binary atoms

In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$

1. Select the non-distinguished variable x_2 as leaf.
2. Choose a "type" for x_2 - say $\exists r^-$.
3. Decomposition into two pieces: $r^*(x_1, x_2)$, $(s \cup u)t^-(x_2, x_2)$
4. Drop second atom as $\exists r^-$ ensures st^- loop

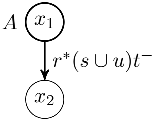
One rewriting step - complex binary atoms

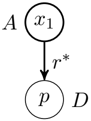
In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$

1. Select the non-distinguished variable x_2 as leaf.
2. Choose a "type" for x_2 - say $\exists r^-$.
3. Decomposition into two pieces: $r^*(x_1, x_2)$, $(s \cup u)t^-(x_2, x_2)$
4. Drop second atom as $\exists r^-$ ensures st^- loop
5. Choose "type" of parent node p (which may not be x_1) and role from p to x_2 - say D and r.
   - parent type must entail chosen role, which in turn must give child type

One rewriting step - complex binary atoms

In $\mathcal{T}$: $D \sqsubseteq \exists r, A \sqsubseteq \exists s, s \sqsubseteq t, \exists r^- \sqsubseteq A$





1. Select the non-distinguished variable x_2 as leaf.
2. Choose a "type" for x_2 - say $\exists r^-$.
3. Decomposition into two pieces: $r^*(x_1, x_2)$, $(s \cup u)t^-(x_2, x_2)$
4. Drop second atom as $\exists r^-$ ensures st^- loop
5. Choose "type" of parent node p (which may not be x_1) and role from p to x_2 - say D and r.
   - parent type must entail chosen role, which in turn must give child type
6. Add $D(p)$ and replace $r^*(x_1, x_2)$ by $r^*(x_1, p)$

Evaluation step

In the second step of our algorithm, we must check whether there is a match for Q which maps all variables to ABox individuals.

Subtlety: apart from closing under concept/role hierarchy, we must consider that paths may still need to pass through the anonymous part!

Our solution:

- when traversing the ABox, allow "shortcuts" which use loops passing through the anonymous part
- the loops which are available at an ABox individual depend solely on its type (i.e. which concepts it satisfies), so can be easily computed

Summing up

- Many things have been achieved in OBQA
- Still many challenges ahead:
  - scalability!
  - for expressive DLs, we still far from feasible algorithms
  - even for $\mathcal{EL}$, practicability is not so clear
- Languages like C2RPQs seem desirable, and not too costly
- More study of fragments with tractable combined complexity is needed
  - in other DLs
  - for the intractable cases, restrictions on the query and on the TBox possible
  - should be validated against real world ontologies

Thank you!