# Reformulating Ontological Queries using Materialised Rewritings

Alexandros Chortaras, Giorgos Stamou, Giorgos Stoilos
National Technical University of Athens

Bernardo Cuenca Grau
University of Oxford

# Ontology-based data access (OBDA)

Use of ontologies to provide a semantic-based access to the underlying data sources.

- The data are stored in a (usually relational) database with schema $\mathcal{D}$.
- An ontology $\mathcal{O}$ provides a high-level conceptual schema of the data.
- Query answering:
  - User poses a conjunctive query $\mathcal{U}$ formulated using the ontology terminology.
  - $\mathcal{U}$ must be transformed into a set of queries $\mathcal{Q}$ to the data schema $\mathcal{D}$.
  - The transformation should encode in $\mathcal{Q}$ the knowledge in $\mathcal{O}$.
- Techniques: Query rewriting.

  $\mathcal{O} = \{B \sqsubseteq A\}$, User query: $\mathcal{Q}_A \leftarrow A(x)$, Rewriting: $\{\mathcal{Q}_A \leftarrow A(x),\ \mathcal{Q}_A \leftarrow B(x)\}$

# Overview

- Rewritings and reformulations.
- Use case: Use of reformulations to implement restricted access to database information.
- Algorithm for constructing a complete rewriting from a set of precomputed rewritings.
- Evaluation for DL-Lite$_R$ and $\mathcal{ELHI}$ ontologies.

## Query Reformulations and Rewritings

- A *reformulation* of a query $\mathcal{U}$ using ontology $\mathcal{O}$ is a set of queries $\mathcal{R}$ such that:

$$\mathcal{O} \cup \mathcal{U} \models \mathcal{R}$$

  hence, for any dataset $D$:

$$\mathsf{cert}(\mathcal{U}, \mathcal{O} \cup D) \supseteq \mathsf{cert}(\mathcal{R}, D)$$

  *A set of queries that returns some of the answers to the original query*.

- A *rewriting* of a query $\mathcal{U}$ using ontology $\mathcal{O}$ is a reformulation $\mathcal{R}$ such that for any dataset $D$:

$$\mathsf{cert}(\mathcal{U}, \mathcal{O} \cup D) \subseteq \mathsf{cert}(\mathcal{R}, D)$$

  *A set of queries that returns all answers to the original query*.

- A reformulation is an “incomplete” rewriting.

## Query Reformulation and Rewriting

- Rewriting systems: DL-Lite$_R$: **Mastro**, **Presto**, **Quest**, **Nyaya**, **Icaros**, **Ontop**

  DL-Lite$_R$ and $\mathcal{ELHI}$: **Rapid**, **Requiem**

  Horn-$\mathcal{SHIQ}$: **Clipper**
- The rewriting is computed from scratch: existing rewritings are not exploited.
- The rewriting computation may be costly.
- Materialized rewritings could be used to:
  - Speed up computation of new query rewritings.
- Materialized reformulations could be used to:
  - Implement views for users with different access restrictions.

## Use case: Reformulation

- Ontology $\mathcal{O} = \{\text{OnPayroll} \sqsubseteq \text{Empl}, \text{Mngr} \sqsubseteq \text{Empl}, \text{DeptMngr} \sqsubseteq \text{Mngr}\}$
- Access level 1: access to all employs excluding department managers.
- Reform.: $\mathcal{Q}_1(v) \leftarrow \text{Empl}(v)$: $\{\mathcal{Q}_1(v) \leftarrow \text{Empl}(v),\ \mathcal{Q}_1(v) \leftarrow \text{Mngr}(v),\ \mathcal{Q}_1(v) \leftarrow \text{OnPayroll}(v)\}$

  $\mathcal{Q}_2(v) \leftarrow \text{DeptMngr}(x)$: $\{\ \}$
- User query $\mathcal{U}$: $\mathcal{Q}_A(x) \leftarrow \text{supervisedBy}(x, y), \text{Empl}(y)$
- Rewriting: $\{\mathcal{Q}_A(x) \leftarrow \text{supervisedBy}(x, y), \text{Empl}(y),$
  $\mathcal{Q}_A(x) \leftarrow \text{supervisedBy}(x, y), \text{Mngr}(y),$
  $\mathcal{Q}_A(x) \leftarrow \text{supervisedBy}(x, y), \text{OnPayroll}(y),$
  $\mathcal{Q}_A(x) \leftarrow \text{supervisedBy}(x, y), \text{DeptMngr}(y)\}$

## Finding relevant queries: Unfolding

- Assume a given set of precomputed rewritings. We need to identify the relevant queries to $\mathcal{U}$.

- $\mathcal{Q}_A(x) \leftarrow \mathrm{supervisedBy}(x,y), \mathrm{Empl}(y)$ with $\mathcal{Q}_1(v) \leftarrow \mathrm{Empl}(v)$

- A query $\mathcal{Q}$ *is $\sigma$-relevant to $\mathcal{U}$* if we can isomorphically map a subset of the body of $\mathcal{Q}$ to the body of $\mathcal{U}$: $\mathsf{bd}(\mathcal{Q}) \subseteq \mathsf{bd}(\mathcal{U}\sigma)$

- $\mathcal{Q}_A(x) \leftarrow \mathrm{supervisedBy}(x,y), \mathrm{Empl}(y)$ +

$$\mathcal{R} = \{\mathcal{Q}_1(v) \leftarrow \mathrm{Empl}(v),\ \mathcal{Q}_1(v) \leftarrow \mathrm{Mngr}(v),\ \mathcal{Q}_1(v) \leftarrow \mathrm{OnPayroll}(v)\}$$

- Unfolding: replace the matched atoms in $\mathcal{U}$ with the query bodies in $\mathcal{R}$.

$$\mathcal{U}' = [\mathsf{hd}(\mathcal{U}\sigma) \leftarrow \vee(\mathsf{bd}(\mathcal{U}\sigma) \setminus \mathsf{bd}(\mathcal{Q})) \cup \mathsf{bd}(\mathcal{Q}')]_\tau$$

- The unfolding is sound if $\mathcal{O} \cup \{\mathcal{U}\} \models \mathcal{U}'$.

## Guaranteeing soundness: Safeness

- Ontology: $\mathcal{O} = \{A \sqsubseteq \exists S,\ S \sqsubseteq R\}$
- User query $\mathcal{U}$: $\mathcal{Q}_A(x) \leftarrow R(x, y) \wedge C(y)$
- Computed rewriting for $\mathcal{Q}_1(v) \leftarrow R(v, u)$ : $\{\mathcal{Q}_1(v) \leftarrow R(v, u),$ $\mathcal{Q}_1(v) \leftarrow S(v, u),$ $\mathcal{Q}_1(v) \leftarrow A(v)\}$
- Unfolding: $\{\mathcal{Q}_A(x) \leftarrow R(x, y) \wedge C(y),\ \mathcal{Q}_A(x) \leftarrow S(x, y) \wedge C(y),\ \mathcal{Q}_A(x) \leftarrow A(x) \wedge C(y)\}$
- The last query is not entailed by $\mathcal{O} \cup \{\mathcal{U}\}$.
- The unfolding $\mathcal{U}'$ is *safe* if:

$$\mathsf{var}((\mathsf{bd}(\mathcal{U}\sigma) \setminus \mathsf{bd}(Q)) \cap \mathsf{var}(Q))_\tau \subseteq \mathsf{var}(\mathcal{Q}') \qquad \{v, u\} \subseteq \{v\}$$

$$(\mathsf{avar}(\mathcal{U}\sigma) \cap \mathsf{var}(Q))_\tau \subseteq \mathsf{var}(\mathcal{Q}') \qquad \{v\} \subseteq \{v\}$$

## Guaranteeing completeness

- Ontology: $\mathcal{O} = \{A \sqsubseteq \exists R, \exists R^{-} \sqsubseteq C\}$
- User query $\mathcal{U}$: $\mathcal{Q}_A(x) \leftarrow R(x,y) \wedge C(y)$
- Computed rewriting for $\mathcal{Q}_1(v) \leftarrow C(v)$ : $\{\mathcal{Q}_1(v) \leftarrow C(v),$
  $\mathcal{Q}_1(v) \leftarrow R(u,v)\}$
- Unfolding: $\{\mathcal{Q}_A(x) \leftarrow R(x,y) \wedge C(y),\ \mathcal{Q}_A(x) \leftarrow R(x,y) \wedge R(z,y) = \mathcal{Q}_A(x) \leftarrow R(x,y)\}$
- For the above reformulation $\mathcal{R}$ and data set $D = \{A(c)\}$ we have:

$$c \in \text{cert}(\mathcal{U}, \mathcal{O} \cup D) \text{ but } c \notin \text{cert}(\mathcal{R}, D)$$

- A *reasoning step* using some inference system $\Gamma$ is required.

$$\mathcal{Q}_1(x) \leftarrow R(x,y) + A \sqsubseteq \exists R \Rightarrow \mathcal{Q}_1(x) \leftarrow A(x)$$

## Algorithm

- Inputs: Ontology $\mathcal{O}$, user query $\mathcal{U}$, materialized rewriting sets $(\mathcal{Q}, \mathcal{R})$.
- Steps:
  - Initialize an empty set of queries $\mathcal{R}_{out}$.
  - Saturate the clausification of $\mathcal{O}$ using inference system $\Gamma$.
    Add the clauses to $\mathcal{R}_{out}$.
  - Unfold $\mathcal{U}$ using the relevant rewritings in $(\mathcal{Q}, \mathcal{R})$.
    Add the rewritings to $\mathcal{R}_{out}$.
  - Iteratively check if new queries can be obtained using inference system $\Gamma$.
    Add the non-reduntant new clauses to $\mathcal{R}_{out}$.
- Output: The set $\mathcal{R}_{out}$.

## Evaluation

- We constructed the prototype tool **QuDec**.
- It relies on **Requiem** as inference system $\Gamma$.
- Evaluation for DL-Lite$_R$ and $\mathcal{ELHI}$ ontologies.
- Tests using three precomputed rewriting sets:

  $\mathcal{Q}^{at}$: single atom queries

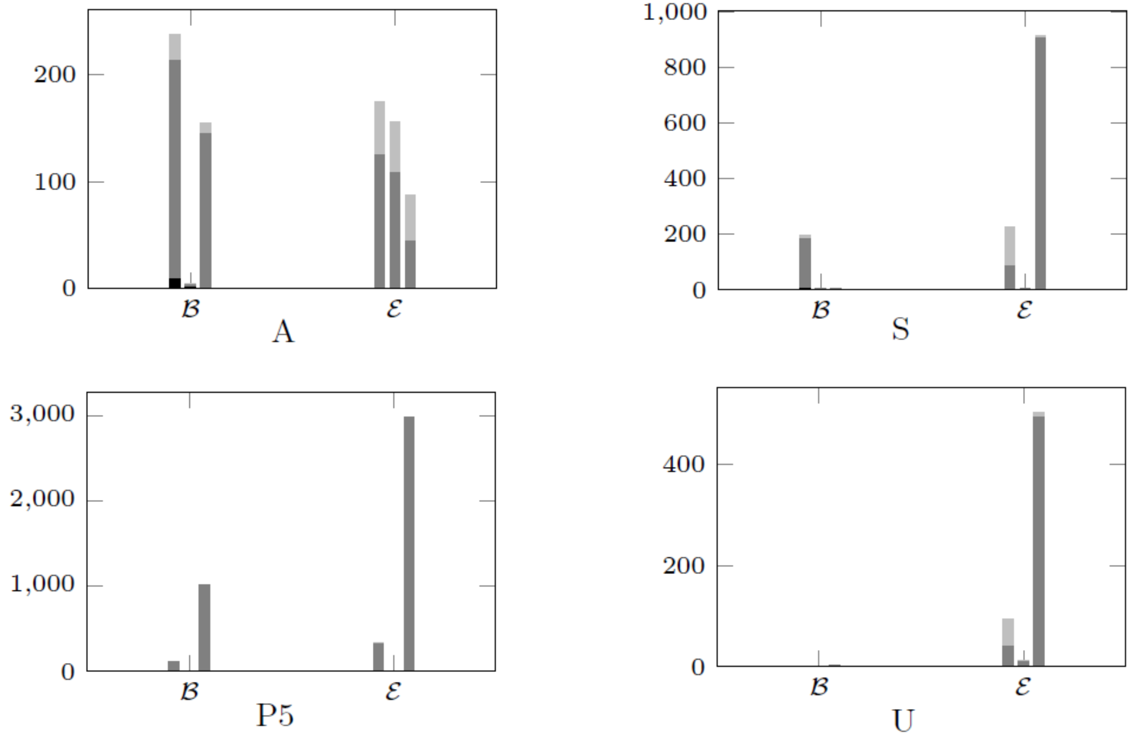
  $\mathcal{Q}^{tr}$: conjunctive queries computed by **SyGENIA**

  $\mathcal{Q}^{\emptyset}$: no precomputed rewritings
- Evaluation user queries:

  $\mathcal{B}$: conjunctive queries computed by **SyGENIA**

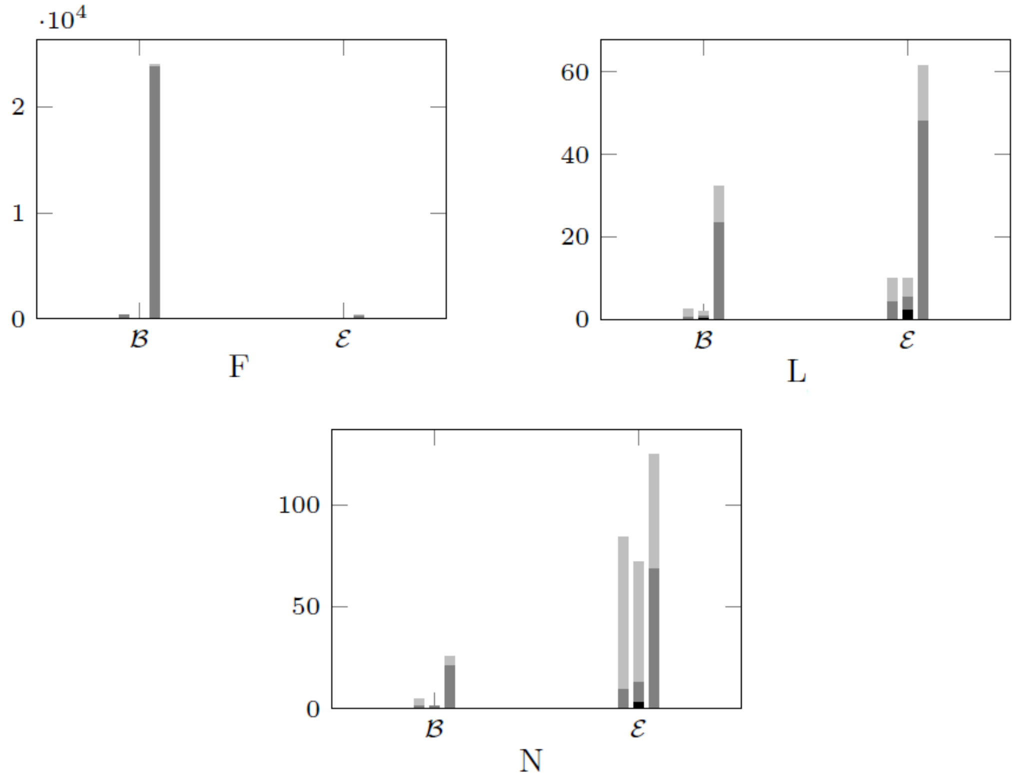
  $\mathcal{E}$: manually constructed queries

# Evaluation results (DL-Lite$_R$)




*A: Accessibility ontology, S: EU institutions ontology, U: University ontology, P5: Graph ontology*

# Evaluation results ($\mathcal{ELHI}$)




*F: Photo ontology, L: LUBM ontology, N: NASA SWEET ontology*

## Conclusions

- A sound reformulation algorithm to be used with OBDA applications where user access is restricted.
- A sound and complete rewriting algorithm that reuses precomputed rewritings to speed up computation.
- The evaluation shows that the use of materializes rewritings can lead to performance improvement.
- Investigation for further optimizations to the algorithm.