

End-User Access to Big Data Using Ontologies

Part 2: Querying Data and Knowledge

Diego Calvanese

Free University of Bozen-Bolzano



Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

International Winter School on Big Data
Tarragona, Spain, January 26–30 , 2015

Part 2

Querying Data and Knowledge

Outline of Part 2

- 1 Querying databases and ontologies
 - Query answering in traditional databases
 - Query answering in ontologies
 - Query answering in ontology-based data access
- 2 Query answering in Description Logics
 - Certain answers
 - Complexity of query answering
- 3 References

Outline of Part 2

- 1 Querying databases and ontologies
 - Query answering in traditional databases
 - Query answering in ontologies
 - Query answering in ontology-based data access
- 2 Query answering in Description Logics
- 3 References

Query answering

In ontology-based data access we are interested in a reasoning service that is not typical in ontologies (or in a FOL theory, or in UML class diagrams, or in a knowledge base) but it is very common in databases: **query answering**.

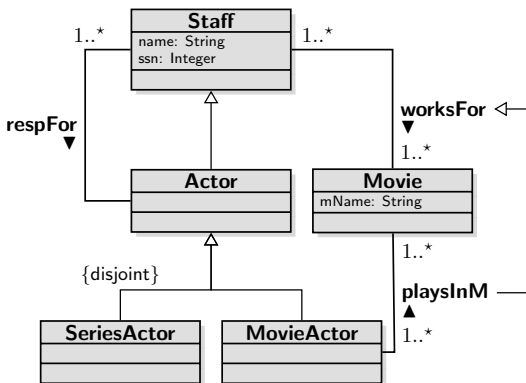
Query

Is an expression at the intensional level denoting a set of tuples of individuals satisfying a given condition.

Query Answering

Is the reasoning service that actually computes the answer to a query.

Example of a query over an ontology



$$q(st, sa, n) \leftarrow \exists t, a, m. \text{worksFor}(t, m) \wedge \text{playsInM}(a, m) \wedge \text{respFor}(t, a) \wedge \text{ssn}(t, st) \wedge \text{ssn}(a, sa) \wedge \text{name}(t, n) \wedge \text{name}(a, n)$$

Query answering under different assumptions

There are two fundamentally different assumptions when addressing query answering:

- **Complete information** on the data, as in traditional databases.
- **Incomplete information** on the data, as in ontologies (aka knowledge bases), but also information integration in databases.

Outline of Part 2

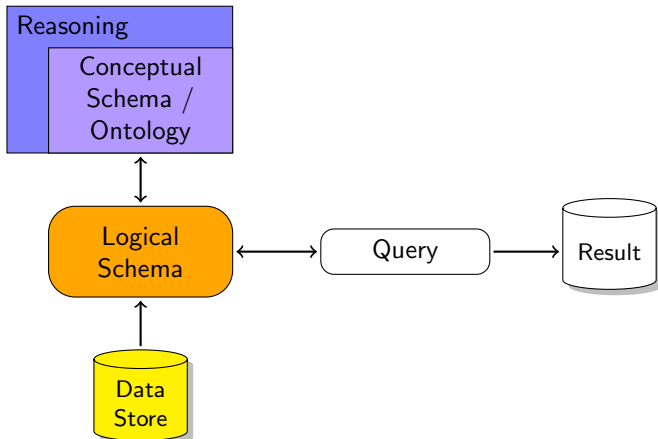
- 1 Querying databases and ontologies
 - Query answering in traditional databases
 - Query answering in ontologies
 - Query answering in ontology-based data access
- 2 Query answering in Description Logics
- 3 References

Query answering in traditional databases

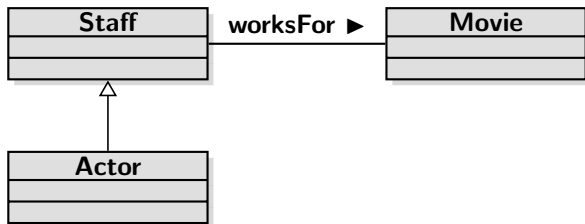
- Data are completely specified (CWA), and typically large.
- Schema/intensional information used in the design phase.
- At **runtime**, the data is assumed to satisfy the schema, and therefore the **schema is not used**.
- Queries allow for complex navigation paths in the data (cf. SQL).

↪ Query answering amounts to **query evaluation**, which is computationally easy.

Query answering in traditional databases (cont'd)



Query answering in traditional databases – Example



For each concept/relationship we have a (complete) table in the DB.

DB: Staff = { john, mary, nick }
 Actor = { john, nick }
 Movie = { mA, mB }
 worksFor = { (john,mA), (mary,mB) }

Query: $q(x) \leftarrow \exists m. \text{Actor}(x) \wedge \text{Movie}(m) \wedge \text{worksFor}(x, m)$

Answer: { john }

Outline of Part 2

- 1 Querying databases and ontologies
 - Query answering in traditional databases
 - Query answering in ontologies
 - Query answering in ontology-based data access
- 2 Query answering in Description Logics
- 3 References

Query answering in ontologies

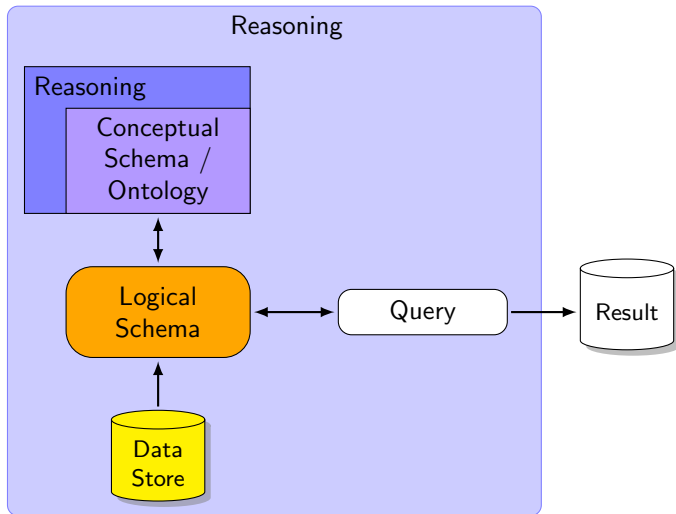
- An ontology (or conceptual schema, or knowledge base) imposes constraints on the data.
- Actual data may be incomplete or inconsistent w.r.t. such constraints.
- The system has to take into account the constraints during query answering, and overcome incompleteness or inconsistency.

↪ Query answering amounts to **logical inference**, which is computationally more costly.

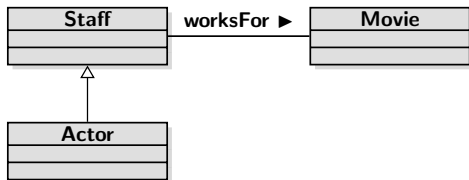
Note:

- The size of the data is not considered critical (comparable to the size of the intensional information).
- Queries are typically simple, i.e., atomic (a class name), and query answering amounts to instance checking.

Query answering in ontologies (cont'd)



Query answering in ontologies – Example



The tables in the database may be **incompletely specified**, or even missing for some classes/properties.

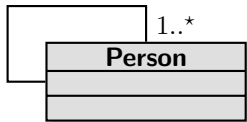
DB: Actor \supseteq { john, nick }
 Movie \supseteq { mA, mB }
 worksFor \supseteq { (john,mA), (mary,mB) }

Query: $q(x) \leftarrow \text{Staff}(x)$

Answer: { john, nick, mary }

Query answering in ontologies – Example 2

◀ hasFather



Each person has a father, who is a person.

DB: $\text{Person} \supseteq \{ \text{john}, \text{nick}, \text{toni} \}$
 $\text{hasFather} \supseteq \{ (\text{john}, \text{nick}), (\text{nick}, \text{toni}) \}$

Queries: $q_1(x, y) \leftarrow \text{hasFather}(x, y)$

$q_2(x) \leftarrow \exists y. \text{hasFather}(x, y)$

$q_3(x) \leftarrow \exists y_1, y_2, y_3. \text{hasFather}(x, y_1) \wedge \text{hasFather}(y_1, y_2) \wedge \text{hasFather}(y_2, y_3)$

$q_4(x, y_3) \leftarrow \exists y_1, y_2. \text{hasFather}(x, y_1) \wedge \text{hasFather}(y_1, y_2) \wedge \text{hasFather}(y_2, y_3)$

Answers: to q_1 : $\{ (\text{john}, \text{nick}), (\text{nick}, \text{toni}) \}$

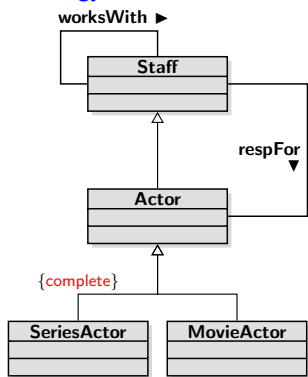
to q_2 : $\{ \text{john}, \text{nick}, \text{toni} \}$

to q_3 : $\{ \text{john}, \text{nick}, \text{toni} \}$

to q_4 : $\{ \}$

QA in ontologies – Andrea's Example ¹

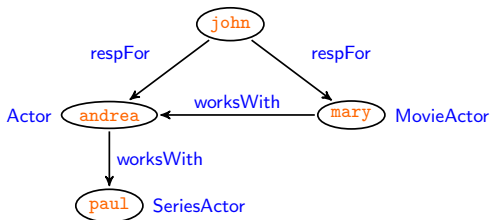
Ontology:



Actor is **partitioned into** SeriesActor and MovieActor.

DB (incomplete):

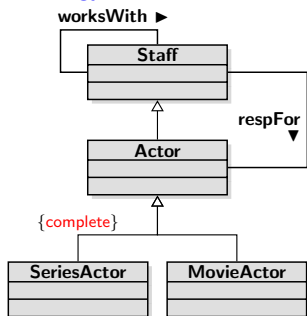
$\text{Staff} \supseteq \{ \text{andrea, paul, mary, john} \}$
 $\text{Actor} \supseteq \{ \text{andrea, paul, mary} \}$
 $\text{SeriesActor} \supseteq \{ \text{paul} \}$
 $\text{MovieActor} \supseteq \{ \text{mary} \}$
 $\text{respFor} \supseteq \{ (\text{john, andrea}), (\text{john, mary}) \}$
 $\text{worksWith} \supseteq \{ (\text{mary, andrea}), (\text{andrea, paul}) \}$



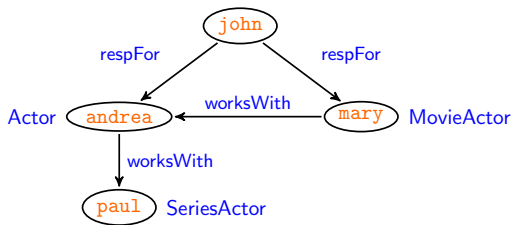
¹Due to Andrea Schaerf [Schaerf, 1993].

QA in ontologies – Andrea's Example (cont'd)

Ontology:



DB (incomplete):



$$q(x) \leftarrow \exists y, z. \text{respFor}(x, y) \wedge \text{MovieActor}(y) \wedge \text{worksWith}(y, z) \wedge \text{SeriesActor}(z)$$

Answer: { john }

To determine this answer, we need to **reason model by model**.

Outline of Part 2

- 1 Querying databases and ontologies
 - Query answering in traditional databases
 - Query answering in ontologies
 - Query answering in ontology-based data access
- 2 Query answering in Description Logics
- 3 References

Query answering in ontology-based data access

In OBDA, we have to face the difficulties of both settings:

- The actual **data** is stored in external information sources (i.e., databases), and thus its size is typically **very large**.
- The ontology introduces **incompleteness** of information, and we have to do logical inference, rather than query evaluation.
- We want to take into account at **runtime** the **constraints** expressed in the ontology.
- We want to answer **complex database-like queries**.
- We may have to deal with multiple information sources, and thus face also the problems that are typical of data integration.

Questions that need to be addressed

In the context of ontology-based data access:

- 1 Which is the “right” **query language**?
- 2 Which is the “right” **ontology language**?
- 3 How can we bridge the **semantic mismatch** between the ontology and the data sources?
- 4 How can **tools for ontology-based data access** take into account these issues?

Which language to use for querying ontologies?

Two borderline cases:

- 1 **Just classes and properties** of the ontology \rightsquigarrow instance checking
 - Ontology languages are tailored for capturing intensional relationships.
 - They are quite **poor as query languages**:
Cannot refer to same object via multiple navigation paths in the ontology, i.e., allow only for a limited form of JOIN, namely chaining.
- 2 **Full SQL** (or equivalently, domain independent first-order logic)
 - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

A good tradeoff is to use (unions of) **conjunctive queries**.

Unions of conjunctive queries

A good tradeoff is to use (unions of) **conjunctive queries** (UCQs):

- A (U)CQ is a first-order query using only conjunction, existential quantification (and disjunction) – No forms of negation.
- Correspond to SQL/relational algebra **(union) select-project-join (SPJ) queries** – the most frequently asked queries.
- For (U)CQs over an ontology, the predicates in atoms are concepts and roles of the ontology.

Notation for CQs

We write conjunctive queries as:

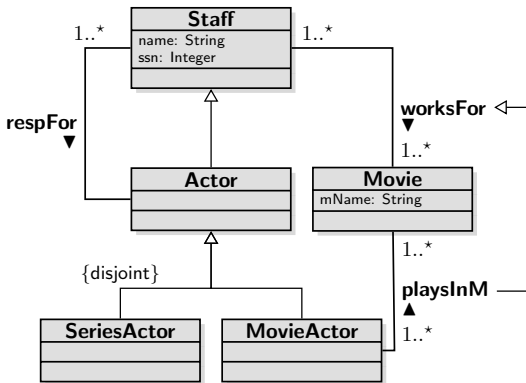
$$q(\vec{x}) \leftarrow \exists \vec{y}. E_1(\vec{z}_1) \wedge \cdots \wedge E_n(\vec{z}_n)$$

or as rules (Datalog notation):

$$q(\vec{x}) \leftarrow E_1(\vec{z}_1), \dots, E_n(\vec{z}_n)$$

Example of conjunctive query over an ontology

The queries we have seen so far are examples of conjunctive queries.



$$q(st, sa, n) \leftarrow \exists t, a, m. \text{worksFor}(t, m) \wedge \text{playsInM}(a, m) \wedge \text{respFor}(t, a) \wedge \text{ssn}(t, st) \wedge \text{ssn}(a, sa) \wedge \text{name}(t, n) \wedge \text{name}(a, n)$$

Outline of Part 2

- 1 Querying databases and ontologies
- 2 Query answering in Description Logics
 - Certain answers
 - Complexity of query answering
- 3 References

Outline of Part 2

- 1 Querying databases and ontologies
- 2 Query answering in Description Logics
 - Certain answers
 - Complexity of query answering
- 3 References

Certain answers to a query

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology, \mathcal{I} an interpretation for \mathcal{O} , and $q(\vec{x}) = \exists \vec{y}. conj(\vec{x}, \vec{y})$ a CQ.

The **answer** to $q(\vec{x})$ over \mathcal{I} , denoted $q^{\mathcal{I}}$

is the set of **tuples \vec{c} of constants of \mathcal{A}** such that the formula $\exists \vec{y}. conj(\vec{c}, \vec{y})$ evaluates to true in \mathcal{I} .

We are interested in finding those answers that hold in all models of an ontology.

The **certain answers** to $q(\vec{x})$ over $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, denoted $cert(q, \mathcal{O})$

are the **tuples \vec{c} of constants of \mathcal{A}** such that $\vec{c} \in q^{\mathcal{I}}$, for **every model \mathcal{I}** of \mathcal{O} .

Query answering in ontologies

Query answering over an ontology \mathcal{O}

Is the problem of **computing the certain answers** to a query over \mathcal{O} .

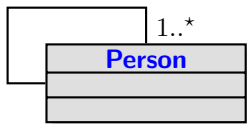
Computing certain answers is a form of **logical implication**:

$$\vec{c} \in \text{cert}(q, \mathcal{O}) \quad \text{iff} \quad \mathcal{O} \models q(\vec{c})$$

Note: A special case of query answering is **instance checking**: it amounts to answering the boolean query $q() \leftarrow A(c)$ (resp., $q() \leftarrow P(c_1, c_2)$) over \mathcal{O} (in this case \vec{c} is the empty tuple).

Query answering in ontologies – Example

◀ **hasFather**



TBox \mathcal{T} : $\exists \text{hasFather} \sqsubseteq \text{Person}$
 $\exists \text{hasFather}^- \sqsubseteq \text{Person}$
 $\text{Person} \sqsubseteq \exists \text{hasFather}$

ABox \mathcal{A} : $\text{Person}(\text{john}), \text{Person}(\text{nick}), \text{Person}(\text{toni})$
 $\text{hasFather}(\text{john}, \text{nick}), \text{hasFather}(\text{nick}, \text{toni})$

Queries:

$q_1(x, y) \leftarrow \text{hasFather}(x, y)$

$q_2(x) \leftarrow \exists y. \text{hasFather}(x, y)$

$q_3(x) \leftarrow \exists y_1, y_2, y_3. \text{hasFather}(x, y_1) \wedge \text{hasFather}(y_1, y_2) \wedge \text{hasFather}(y_2, y_3)$

$q_4(x, y_3) \leftarrow \exists y_1, y_2. \text{hasFather}(x, y_1) \wedge \text{hasFather}(y_1, y_2) \wedge \text{hasFather}(y_2, y_3)$

Certain answers: $\text{cert}(q_1, \langle \mathcal{T}, \mathcal{A} \rangle) = \{ (\text{john}, \text{nick}), (\text{nick}, \text{toni}) \}$
 $\text{cert}(q_2, \langle \mathcal{T}, \mathcal{A} \rangle) = \{ \text{john}, \text{nick}, \text{toni} \}$
 $\text{cert}(q_3, \langle \mathcal{T}, \mathcal{A} \rangle) = \{ \text{john}, \text{nick}, \text{toni} \}$
 $\text{cert}(q_4, \langle \mathcal{T}, \mathcal{A} \rangle) = \{ \}$

Outline of Part 2

- 1 Querying databases and ontologies
- 2 Query answering in Description Logics
 - Certain answers
 - Complexity of query answering
- 3 References

Complexity measures for queries over ontologies

When measuring the complexity of answering a query $q(\vec{x})$ over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, various parameters are of importance.

Depending on which parameters we consider, we get different complexity measures:

- **Data complexity**: only the size of the ABox (i.e., the data) matters. TBox and query are considered fixed.
- **Query complexity**: only the size of the query matters. TBox and ABox are considered fixed.
- **Schema complexity**: only the size of the TBox (i.e., the schema) matters. ABox and query are considered fixed.
- **Combined complexity**: no parameter is considered fixed.

In the OBDA setting, **the size of the data largely dominates** the size of the conceptual layer (and of the query).

\leadsto **Data complexity** is the relevant complexity measure.

Data complexity of query answering

When studying the complexity of query answering, we need to consider the associated decision problem:

Recognition problem for query answering

Given an ontology \mathcal{O} , a query q over \mathcal{O} , and a tuple \vec{c} of constants, **check whether** $\vec{c} \in \text{cert}(q, \mathcal{O})$.

We look mainly at the **data complexity** of query answering, i.e., complexity of the recognition problem computed **w.r.t. the size of the ABox only**.

Complexity of query answering in DLs

Studied extensively for (unions of) CQs and various ontology languages:

	Combined complexity	Data complexity
Plain databases	NP-complete	in AC^0 ⁽¹⁾
<i>ALCI</i> , <i>SH</i> , <i>SHIQ</i> , ...	2EXPTIME-complete ⁽³⁾	coNP-complete ⁽²⁾
OWL 2 (and less)	3EXPTIME-hard	coNP-hard

(1) This is what we need to scale with the data.

(2) coNP-hard already for a TBox with a single disjunction

[Donini *et al.*, 1994; Calvanese *et al.*, 2006; Calvanese *et al.*, 2013].

In coNP for very expressive DLs

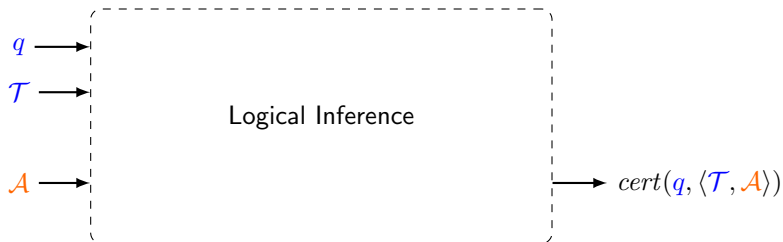
[Levy and Rousset, 1998; Ortiz *et al.*, 2006; Glimm *et al.*, 2007; Ortiz *et al.*, 2008].

(3) [Calvanese *et al.*, 1998; Calvanese *et al.*, 2008; Lutz, 2007]

Questions

- Can we find interesting (description) logics for which query answering can be done efficiently (i.e., in AC^0)?
- If yes, can we leverage relational database technology for query answering?

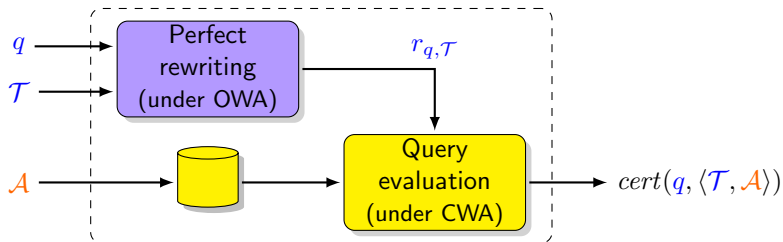
Inference in query answering



To be able to deal with data efficiently, we need to separate the contribution of \mathcal{A} from the contribution of q and \mathcal{T} .

\rightsquigarrow Query answering by **query rewriting**.

Query answering by query rewriting



Query answering can **always** be thought as done in two phases:

- 1 **Perfect rewriting**: produce from q and the TBox \mathcal{T} a new query $r_{q,\mathcal{T}}$ (called the perfect rewriting of q w.r.t. \mathcal{T}).
- 2 **Query evaluation**: evaluate $r_{q,\mathcal{T}}$ over the ABox \mathcal{A} seen as a complete database (and without considering the TBox \mathcal{T}).
 \leadsto Produces $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle)$.

Note: The “always” holds if we pose no restriction on the language in which to express the rewriting $r_{q,\mathcal{T}}$.

\mathcal{L}_Q -rewritability

Let:

- \mathcal{L}_Q be a target query language (i.e., a class of queries), e.g., FOL/SQL;
- \mathcal{L}_T be an ontology TBox language, e.g., \mathcal{ALC} , $DL\text{-Lite}$, OWL 2, ...

Def.: **\mathcal{L}_Q -rewritability** of conjunctive query answering

Conjunctive query answering is **\mathcal{L}_Q -rewritable** in \mathcal{L}_T , if for every TBox \mathcal{T} of \mathcal{L}_T and for every conjunctive query q , the perfect rewriting $r_{q,\mathcal{T}}$ of q w.r.t. \mathcal{T} can be expressed in \mathcal{L}_Q .


Note: Assume that the relevant measure is the size of the data \mathcal{A} . We have:

$$\begin{aligned} &\text{data complexity of computing } \text{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle) \\ &= \\ &\text{complexity of evaluating } r_{q,\mathcal{T}} \text{ over } \mathcal{A} \end{aligned}$$

Hence, **\mathcal{L}_Q -rewritability** is tightly related to the **data complexity of evaluating queries** expressed in the language \mathcal{L}_Q .

Language of the rewriting

The **expressiveness of the ontology language affects the rewriting language**, i.e., the language into which we are able to rewrite UCQs:

- When we can rewrite into **FOL/SQL**.
 \rightsquigarrow Query evaluation can be done in SQL, i.e., via an **RDBMS** (Note: FOL is in AC^0).
- When we can rewrite into **UCQs**.
 \rightsquigarrow Query evaluation can be “optimized” via an **RDBMS**.
- When we can rewrite into **non-recursive Datalog**.
 \rightsquigarrow Query evaluation can be done via an **RDBMS**, but using views.
- When we need an **NLOGSPACE-hard** language to express the rewriting.
 \rightsquigarrow Query evaluation requires (at least) **linear recursion**.
- When we need a **P TIME-hard** language to express the rewriting.
 \rightsquigarrow Query evaluation requires full recursion (e.g., **Datalog**).
- When we need a **coNP-hard** language to express the rewriting.
 \rightsquigarrow Query evaluation requires (at least) the power of **Disjunctive Datalog** 

References I

- [Calvanese *et al.*, 1998] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini.
On the decidability of query containment under constraints.
In Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS), pages 149–158, 1998.
- [Calvanese *et al.*, 2006] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
Data complexity of query answering in description logics.
In Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR), pages 260–270, 2006.
- [Calvanese *et al.*, 2008] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini.
Conjunctive query containment and answering under description logics constraints.
ACM Trans. on Computational Logic, 9(3):22.1–22.31, 2008.
- [Calvanese *et al.*, 2013] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
Data complexity of query answering in description logics.
Artificial Intelligence, 195:335–360, 2013.

References II

- [Donini *et al.*, 1994] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf.
Deduction in concept languages: From subsumption to instance checking.
J. of Logic and Computation, 4(4):423–452, 1994.
- [Glimm *et al.*, 2007] Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler.
Conjunctive query answering for the description logic *SHIQ*.
In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 399–404, 2007.
- [Levy and Rousset, 1998] Alon Y. Levy and Marie-Christine Rousset.
Combining Horn rules and description logics in CARIN.
Artificial Intelligence, 104(1–2):165–209, 1998.
- [Lutz, 2007] Carsten Lutz.
Inverse roles make conjunctive queries hard.
In *Proc. of the 20th Int. Workshop on Description Logic (DL)*, volume 250 of *CEUR Electronic Workshop Proceedings*, <http://ceur-ws.org/>, pages 100–111, 2007.

References III

- [Ortiz *et al.*, 2006] Maria Magdalena Ortiz, Diego Calvanese, and Thomas Eiter.
Characterizing data complexity for conjunctive query answering in expressive description logics.
In Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI), pages 275–280, 2006.
- [Ortiz *et al.*, 2008] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter.
Data complexity of query answering in expressive description logics via tableaux.
J. of Automated Reasoning, 41(1):61–98, 2008.
- [Schaerf, 1993] Andrea Schaerf.
On the complexity of the instance checking problem in concept languages with existential quantification.
J. of Intelligent Information Systems, 2:265–278, 1993.