Query Answering over Description Logic Ontologies

Diego Calvanese

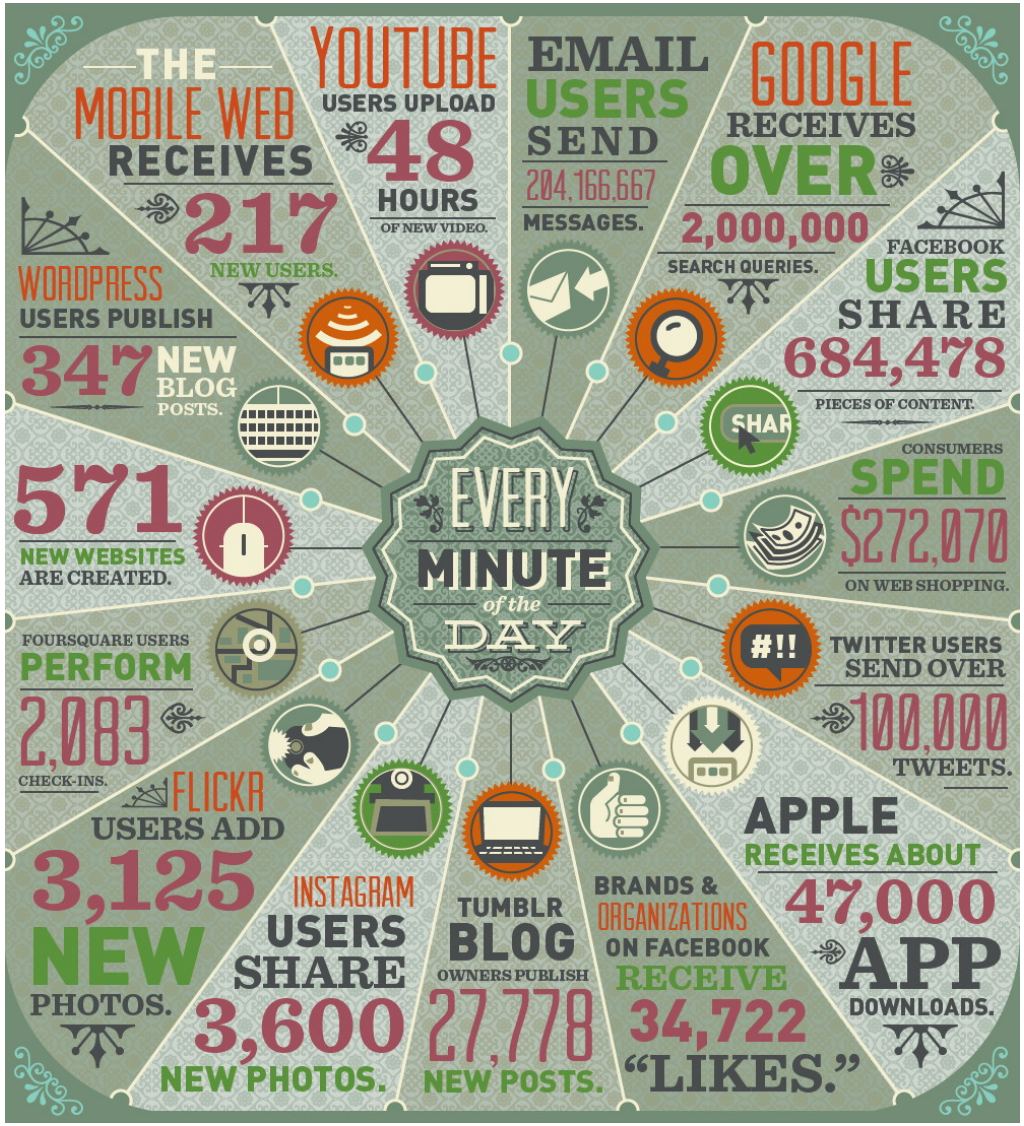
KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen-Bolzano

14th European Conference on Logics in Artificial Intelligence
JELIA 2014

Madeira, Portugal
24–26 September, 2014

Data everywhere



unibz

New challenges in information management

One of the key challenges in complex systems today is the management of information:

- The **amount** of information is steadily increasing.
- Information gets increasingly more **complex**.
- The underlying data may be of **low quality**, e.g., incomplete, or inconsistent.
- Information might be **heterogeneous** (and distributed), but nevertheless needs to be accessed in a uniform way.
- Information is consumed not only by humans, but also by machines.

This calls for sophisticated mechanisms to fulfill today's information management requirements.

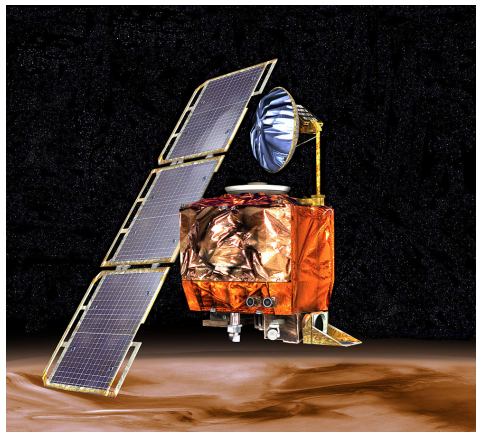
unibz

The role of knowledge representation

- Several efforts come from the **database area** (new data models, new query languages, information integration).
- Managing complex kinds of information has traditionally been the concern of **knowledge representation** in AI.
- However, techniques/tools of KR need to be **adapted** / **extended** / **tuned** to address the new data management challenges.

Emphasis is on the **semantics** of data!

- Fundamental for understanding, sharing, and reasoning.
- Example: Mars climate orbiter case in 1999: 327.6M $ lost because of a metric mixup: same data, different interpretations!



Mars Climate Orbiter 2 by NASA, JPL, Corby Waste.

unibz

Representing knowledge in Description Logics

- **Description Logics (DLs)** stem from early days (1970s) KR formalisms, and assumed their current form in the late 1980s & 1990s.
- Are logics designed to represent and reason on **structured knowledge**.
- Most DLs can be considered as computationally well-behaved **fragments of first-order logic**.
- Semantics given in terms of **first-order interpretations**.
- Tightly connected to **modal logics**: many DLs are syntactic variants of modal logics.
- Come in hundreds of variations, with different semantic and computational properties.
- Have strongly influenced the W3C standard Web Ontology Language **OWL**.

unibz

Description Logic ontology (or knowledge base)

In DLs, the domain of interest is represented by means of:

- concepts: unary predicates Ex.: *Actor*, *Director*, *Movie*, ...
- roles: binary predicates Ex.: *playsIn*, *directs*, ...

Complex concept and role expressions obtained using DL specific constructors.

A **DL ontology** is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

The **TBox** $\mathcal{T}$ represents intensional level information via a set of axioms:

- Concept inclusions: $C_1 \sqsubseteq C_2$, interpreted as $\forall x.C_1(x) \rightarrow C_2(x)$.
- Role inclusions: $R_1 \sqsubseteq R_2$, interpreted as $\forall x, y.R_1(x, y) \rightarrow R_2(x, y)$.
- Role properties: (**transitive** P), (**symmetric** P), ...

The **ABox** $\mathcal{A}$ represents information about individuals via a set of facts:

- Concept membership assertions: $A(d)$
- Role membership assertions: $P(d_1, d_2)$

Note: an ABox is conveniently represented as an edge and node labeled graph.

Traditional DL reasoning services

- Ontology satisfiability: Does $\mathcal{O}$ admit a model?
- Concept satisfiability w.r.t. an ontology: Is there a model $\mathcal{I}$ of $\mathcal{O}$ such that $C^{\mathcal{I}}$ is not empty?
- Concept subsumption w.r.t. an ontology: Does $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ hold, for every model $\mathcal{I}$ of $\mathcal{O}$?

Subsumption is at the basis of classification, i.e., determining the hierarchy of concepts of an ontology.

We are concerned here with a more complicated form of reasoning, namely **query answering**.

Note: A simple form of query answering is instance checking, where the query is simply a concept (instance query): Does $d^{\mathcal{I}} \in C^{\mathcal{I}}$ hold in every model $\mathcal{I}$ of $\mathcal{O}$?

unibz

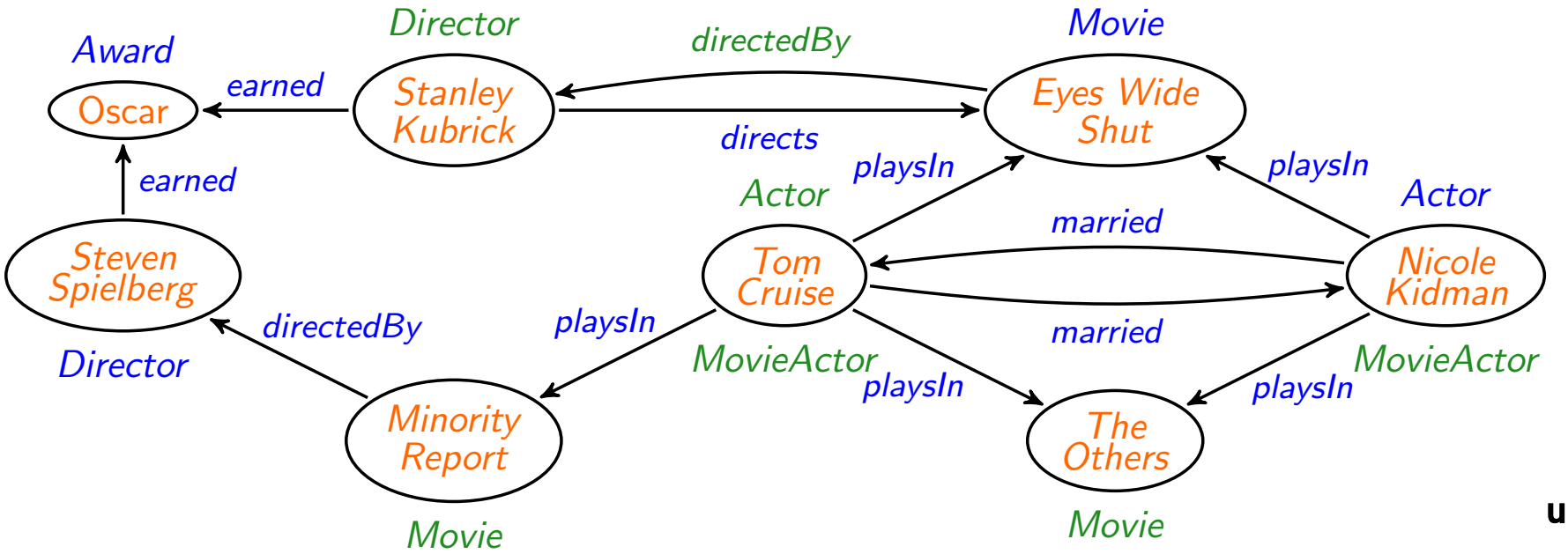
Example TBox and ABox — IMDB

TBox $\mathcal{T}_m$:

$$\exists playsIn.Movie \sqsubseteq MovieActor \qquad \exists directs.\top \sqsubseteq Director$$
$$\exists playsIn.Series \sqsubseteq SeriesActor \qquad Movie \sqsubseteq \exists directedBy.\top$$
$$Actor \equiv SeriesActor \sqcup MovieActor \qquad directs \equiv directedBy^{-}$$
$$MovieActor \sqsubseteq \forall playsIn.Movie \qquad \ldots$$

Note: we use $C_1 \equiv C_2$ as an abbreviation for $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$.

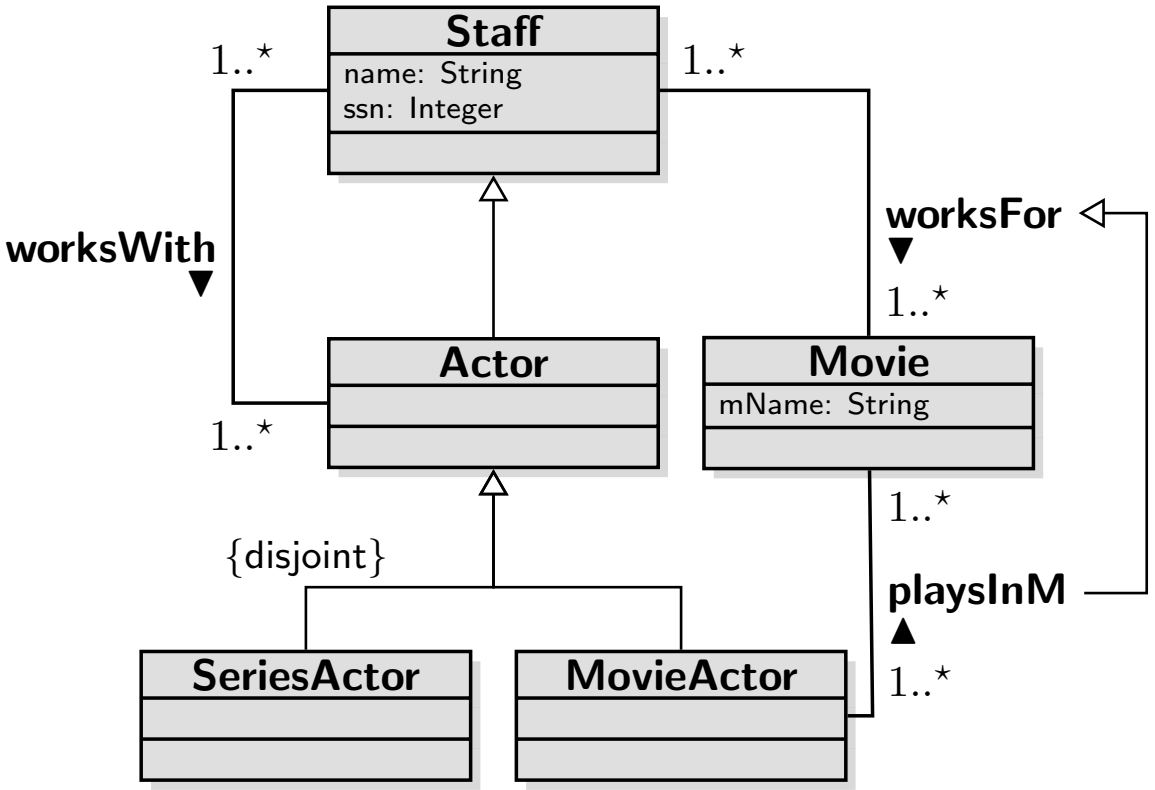
ABox $\mathcal{A}_m$:



unibz

Ontologies vs. conceptual models

We leverage on an extensive amount of work on the tight relationship between conceptual modeling formalisms and ontology languages [Lenzerini and Nobili, 1990; Bergamaschi and Sartori, 1992; Borgida, 1995; C. *et al.*, 1999; Borgida and Brachman, 2003; Berardi *et al.*, 2005; Queralt *et al.*, 2012].



$$
\begin{aligned}
\mathit{Actor} &\sqsubseteq \mathit{Staff}\\
\mathit{SeriesActor} &\sqsubseteq \mathit{Actor}\\
\mathit{MovieActor} &\sqsubseteq \mathit{Actor}\\
\mathit{SeriesActor} &\sqsubseteq \neg \mathit{MovieActor}\\
\\
\mathit{Staff} &\sqsubseteq \exists \mathit{ssn}\\
\exists \mathit{ssn}^- &\sqsubseteq \mathtt{xsd{:}int}\\
(\textbf{funct}\ \mathit{ssn})&\\
\\
\exists \mathit{playsInM} &\sqsubseteq \mathit{MovieActor}\\
\exists \mathit{playsInM}^- &\sqsubseteq \mathit{Movie}\\
\mathit{MovieActor} &\sqsubseteq \exists \mathit{playsInM}\\
\mathit{Movie} &\sqsubseteq \exists \mathit{playsInM}^-\\
\mathit{playsInM} &\sqsubseteq \mathit{worksFor}\\
&\ldots
\end{aligned}
$$

unibz

Outline

unibz

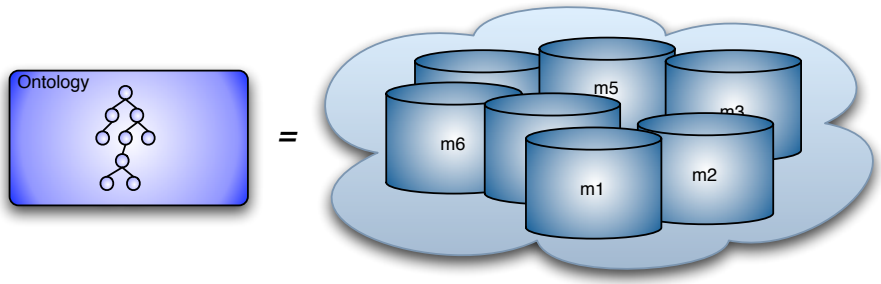
Outline

unibz

Ontology-based query answering

We are interested in answering queries over an ontology:

- Queries might be complex formulas, expressing complicated conditions on the data to be returned.
- The data is in the ABox, but is possibly **incomplete**.
- The knowledge in the TBox is used to complete missing knowledge.
- Incompleteness means that we have to deal with multiple models/databases:



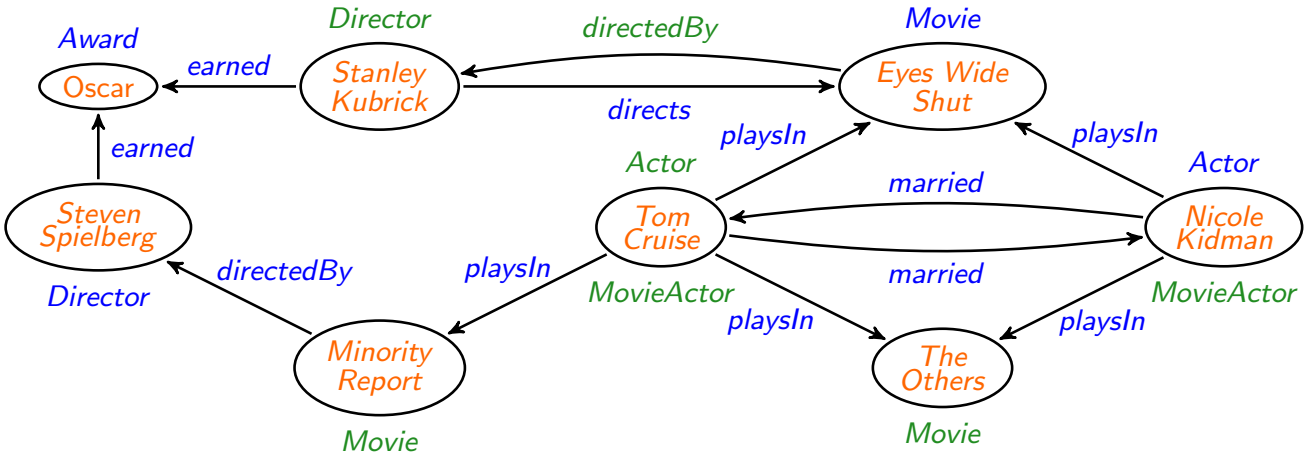
Certain answers

We are interested in the **certain answers**, i.e., the tuples of constants that are an answer to the query in every model of the ABox and TBox.

Ontology-based query answering – Example

Given $\mathcal{T}_m$ and $\mathcal{A}_m$, return all pairs of actors that are married, and that have played in movies directed by directors that have earned the same award.

$$q(x_1, x_2) \leftarrow \exists m_1, m_2, d_1, d_2, a.\, \mathit{Actor}(x_1) \wedge \mathit{playsIn}(x_1, m_1) \wedge \mathit{Movie}(m_1) \wedge \mathit{directedBy}(m_1, d_1) \wedge \mathit{Director}(d_1) \wedge \mathit{earned}(d_1, a) \wedge \mathit{Actor}(x_2) \wedge \mathit{playsIn}(x_2, m_2) \wedge \mathit{Movie}(m_2) \wedge \mathit{directedBy}(m_2, d_2) \wedge \mathit{Director}(d_2) \wedge \mathit{earned}(d_2, a) \wedge \mathit{married}(x_1, x_2) \wedge \mathit{Award}(a)$$



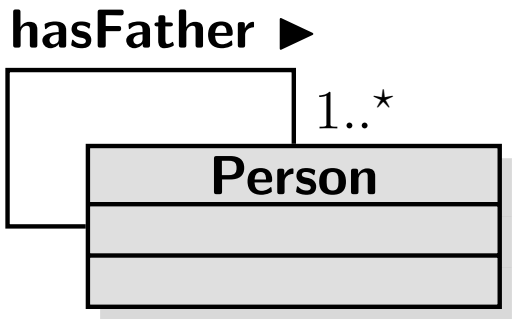
Leveraging the TBox axioms, we infer additional ABox facts.

In this way, we obtain as answer to $q(x_1, x_2)$ the pair *('Tom Cruise', 'Nicole Kidman')*

unibz

Incomplete information – Example

TBox: Each person has a father, who is a person.



ABox: *Person*: ann, bill, chuck
hasFather: (ann,bill), (bill,chuck)

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

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

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

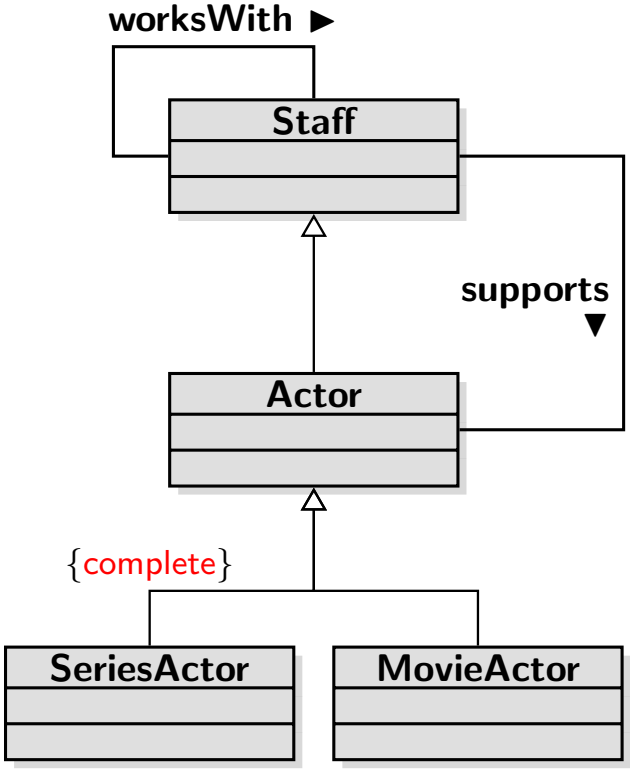
$q_4(x, y_3) \leftarrow \exists y_1, y_2.\, \mathit{hasFather}(x, y_1) \wedge \mathit{hasFather}(y_1, y_2) \wedge \mathit{hasFather}(y_2, y_3)$

Certain answers: to q_1: { (ann,bill), (bill,chuck) }
to q_2: { ann, bill, chuck }
to q_3: { ann, bill, chuck }
to q_4: { }

unibz

OBQA – Andrea's Example [1]

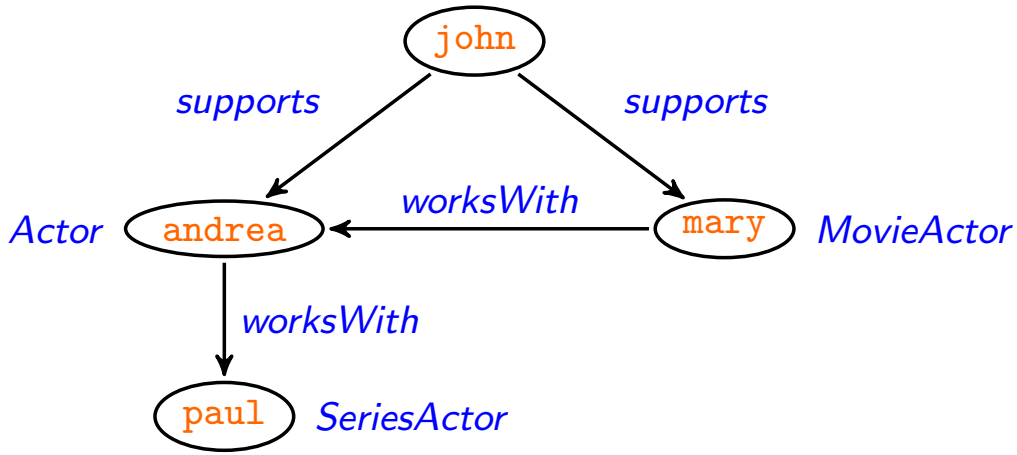
TBox:



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

ABox:

Staff: andrea, paul, mary, john
Actor: andrea, paul, mary
SeriesActor: paul
MovieActor: mary
supports: (john,andrea), (john,mary)
worksWith: (mary,andrea), (andrea,paul)

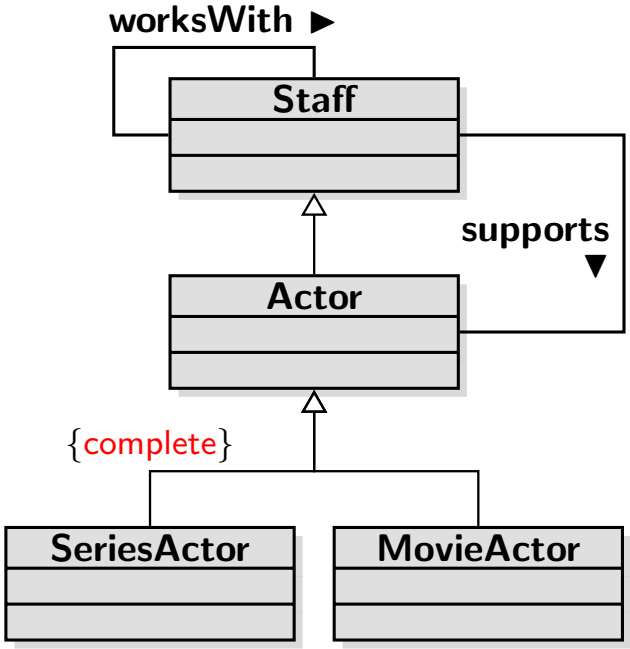


[1] Due to Andrea Schaerf [Schaerf, 1993].

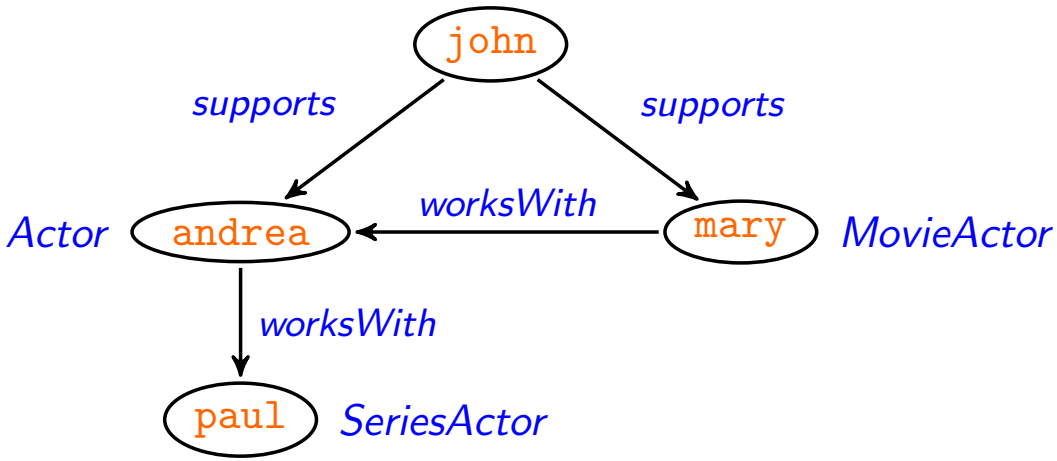
unibz

OBQA – Andrea's Example (cont'd)

TBox:



ABox:



$$q(x) \leftarrow \exists y, z.\, \mathit{supports}(x, y) \quad \wedge \mathit{MovieActor}(y) \wedge \mathit{worksWith}(y, z) \wedge \mathit{SeriesActor}(z)$$

Answer: { `john` }

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

unibz

In this talk

We want to:

1. Analyze techniques for evaluating a query over an ontology.
2. Characterize the computational complexity of the query evaluation problem.

The answer depends on:

- the query language
- the ontology language

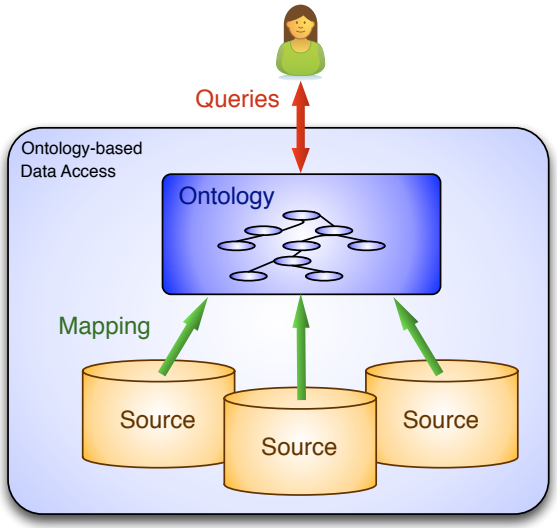
We will look at two representative cases:

- a lightweight Description Logic
- a very expressive Description Logic

unibz

Ontology-based data access

- In ontology-based data access, the aim is to query **external data sources**.
- An OBDA system is based on three main components:



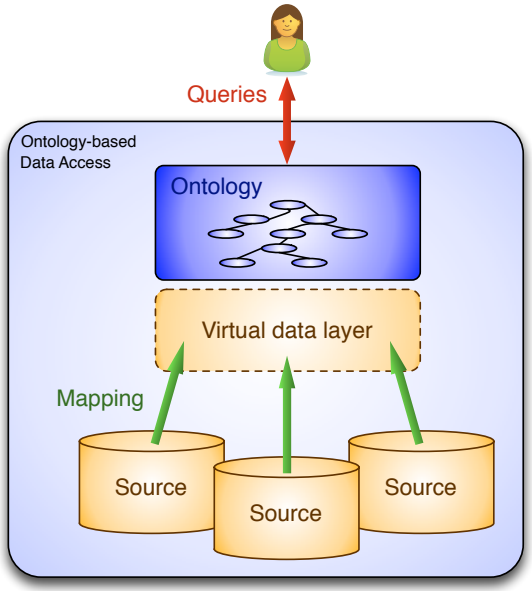
  - Ontology: provides a unified, conceptual view of the managed information.
  - Mappings: semantically link data at the sources with the ontology.
  - Data source(s): are external and independent (possibly heterogeneous).

- Taking into account proper data sources and **mappings** poses alone significant challenges.

unibz

Ontology based query answering vs. data access

- In an OBDA system, the **mapping** $\mathcal{M}$ encodes how the data $\mathcal{D}$ in the source(s) $\mathcal{S}$ should be used to populate the elements of the TBox $\mathcal{T}$.



The data $\mathcal{D}$ and the mapping $\mathcal{M}$ define a **virtual data layer** $\mathcal{V}$, which behaves like a (virtual) ABox.

- Queries are answered w.r.t. $\mathcal{T}$ and $\mathcal{V}$.
- One aim is to avoid materializing the data of $\mathcal{V}$.
- Instead, the intensional information in $\mathcal{T}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{T}$ into queries formulated over $\mathcal{S}$.

- Hence, OBDA relies on OBQA to process queries w.r.t. the TBox $\mathcal{T}$, but in addition is concerned with efficiently dealing with the mapping $\mathcal{M}$.

Ontology based query answering (OBQA) should not be confused with ontology based data access (OBDA).

In this talk we deal with ontology based query answering only.

Which query language to use?

Certain answers, i.e., answers that are logically implied

Query answering amounts to finding the **certain answers** $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ to a query $q(\vec{x})$, i.e., those answers that hold in all models of the ontology $\langle \mathcal{T}, \mathcal{A} \rangle$.

Two extreme cases for the query language to use:

1. Use the ontology language as query language.
   - Ontology languages are tailored for capturing intensional relationships.
   - They are quite **poor as query languages**.

2. Full SQL (or equivalently, first-order logic).
   - Problem: in the presence of incomplete information, query answering becomes **undecidable** (FOL validity).

unibz

Unions of conjunctive queries

A good tradeoff is to use conjunctive queries or their unions.

A conjunctive query (CQ) is a formula of the form

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

where each E_i is a concept or role of the TBox, and each $\vec{z}_i$ is in $\vec{x} \cup \vec{y}$.

We write CQs also as rules (Datalog notation):

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

Note: CQs correspond to the Select-Project-Join fragment of SQL.

A union of conjunctive queries (UCQ) is a disjunction of CQs.

We write a UCQ as a set of rules, all with the same head.

Outline

unibz

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

Note: **Data complexity** is the relevant complexity measure **when the size of the data dominates** the size of the conceptual layer (and of the query).

unibz

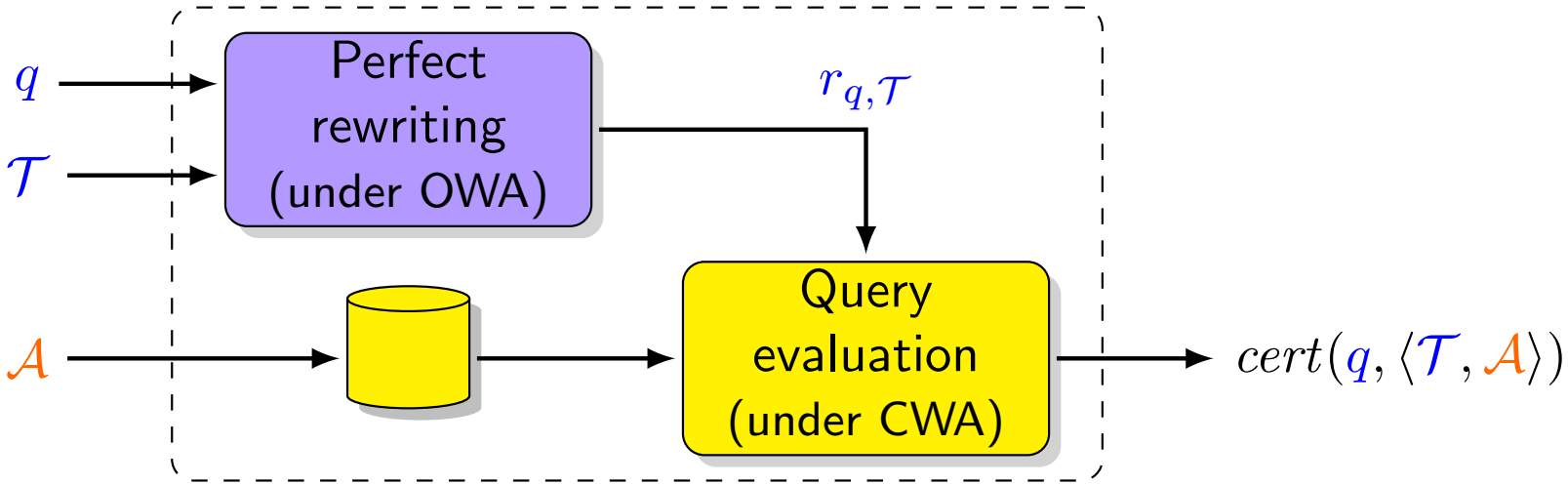
Inference in query answering

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**.

unibz

Query answering by 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 $\mathcal{T}$).
   $\rightsquigarrow$ 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 $r_{q,\mathcal{T}}$.

unibz

$\mathcal{L}_Q$-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-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:

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

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 needed to be 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 PTime-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

unibz

Complexity of conjunctive query answering in DLs

	Combined complexity	Data complexity
Plain databases	NP-c	in AC^0 (1)
DL-Lite family	NP-c (2)	in AC^0 (2)
$\mathcal{EL}$, $\mathcal{ELH}$	NP-c (3)	PTIME-c (3)
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{SHIQ}$, ...	2EXPTIME-c (4)	coNP-c (5)
OWL 2 (and less)	3EXPTIME-hard	coNP-hard

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

(2) [C. *et al.*, 2007a; C. *et al.*, 2013; Artale *et al.*, 2009].

(3) [Krisnadhi and Lutz, 2007; Rosati, 2007]. Becomes undecidable for $\mathcal{EL}^+$

(4) Hardness by [Lutz, 2008; Eiter *et al.*, 2009].
Tight upper bounds obtained for a variety of expressive DLs [C. *et al.*, 1998; Levy and Rousset, 1998; C. *et al.*, 2007b; C. *et al.*, 2008; Glimm *et al.*, 2008a; Glimm *et al.*, 2008b; Lutz, 2008; Eiter *et al.*, 2008; C. *et al.*, 2014].

(5) coNP-hard already for a TBox with a single disjunction
[Donini *et al.*, 1994; C. *et al.*, 2006; C. *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]

Outline

unibz

The *DL-Lite* family

- A family of DLs optimized according to the tradeoff between expressive power and **complexity** of query answering, with emphasis on **data**.
  - The same complexity as relational databases.
  - In fact, **query answering is FOL-rewritable** and hence can be delegated to a relational DB engine.
  - The DLs of the *DL-Lite* family are essentially the maximally expressive DLs enjoying these nice computational properties.
- Nevertheless they have the "right" expressive power: capture the essential features of conceptual modeling formalisms.

Note: The *DL-Lite* family is at the basis of the OWL 2 QL profile of the W3C standard Web Ontology Language OWL 2.

unibz

DL-Lite TBoxes (essential features)

Concept and role language:

- Roles R: either atomic: P
  or an inverse role: P^-
- Concepts C: either atomic: A
  or the projection of a role on one component: $\exists P$, $\exists P^-$

TBox assertions: encode terminological knowledge about the domain

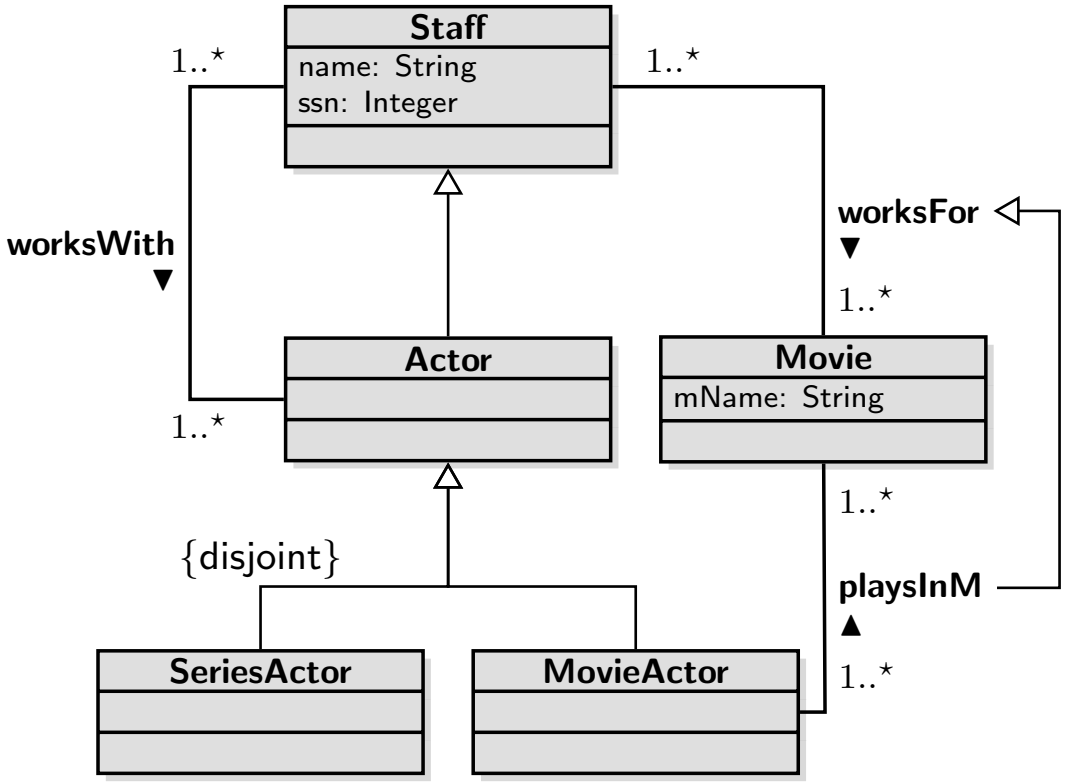
Role inclusion:	$R_1 \sqsubseteq R_2$	Concept inclusion:	$C_1 \sqsubseteq C_2$
Role disjointness:	$R_1 \sqsubseteq \neg R_2$	Concept disjointness:	$C_1 \sqsubseteq \neg C_2$
Role functionality:	$(\textbf{funct}\ R)$		

We also have to impose a restriction on the interaction between role inclusions and role functionality.

Note: *DL-Lite* distinguishes also between abstract objects and data values (ignored here).

unibz

Capturing UML class diagrams/ER schemas in *DL-Lite*



Note: *DL-Lite* cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

$$
\begin{aligned}
\mathit{Actor} &\sqsubseteq \mathit{Staff}\\
\mathit{SeriesActor} &\sqsubseteq \mathit{Actor}\\
\mathit{MovieActor} &\sqsubseteq \mathit{Actor}\\
\mathit{SeriesActor} &\sqsubseteq \neg \mathit{MovieActor}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Staff} &\sqsubseteq \exists \mathit{ssn}\\
\exists \mathit{ssn}^- &\sqsubseteq \mathit{xsd{:}int}\\
&(\textbf{funct}\ \mathit{ssn})
\end{aligned}
$$

$$
\begin{aligned}
\exists \mathit{worksFor} &\sqsubseteq \mathit{Staff}\\
\exists \mathit{worksFor}^- &\sqsubseteq \mathit{Movie}\\
\mathit{Staff} &\sqsubseteq \exists \mathit{worksFor}\\
\mathit{Movie} &\sqsubseteq \exists \mathit{worksFor}^-
\end{aligned}
$$

$$
\begin{aligned}
\exists \mathit{playsInM} &\sqsubseteq \mathit{MovieActor}\\
\exists \mathit{playsInM}^- &\sqsubseteq \mathit{Movie}\\
\mathit{MovieActor} &\sqsubseteq \exists \mathit{playsInM}\\
\mathit{Movie} &\sqsubseteq \exists \mathit{playsInM}^-\\
\mathit{playsInM} &\sqsubseteq \mathit{worksFor}\\
&\ldots
\end{aligned}
$$

unibz

Query answering in *DL-Lite*

Query answering via **query rewriting**

Given a (U)CQ q and an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

1. Compute the perfect rewriting of q w.r.t. $\mathcal{T}$, which is a FOL query.
2. Evaluate the perfect rewriting over $\mathcal{A}$. (We have ignored the mapping.)

We briefly look at *PerfectRef*, a simple algorithm for Step 1.

PerfectRef iterates over:

- rewriting steps that involve inclusion assertions, and
- unification steps.

Note: disjointness assertions and functionalities play a role in ontology satisfiability, but can be ignored during query rewriting (i.e., we have **separability**).

unibz

Query rewriting step: Basic idea

Intuition: an inclusion assertion corresponds to a logic programming rule.

Basic rewriting step:

When an atom in the query unifies with the **head** of the rule, generate a new query by substituting the atom with the **body** of the rule.

We say that the inclusion assertion **applies to** the atom.

Example

The inclusion assertion $\mathit{Actor} \sqsubseteq \mathit{Staff}$
corresponds to the logic programming rule $\mathit{Staff}(z) \leftarrow \mathit{Actor}(z)$.

Consider the query $q(x) \leftarrow \mathit{Staff}(x)$.

By applying the inclusion assertion to the atom $\mathit{Staff}(x)$, we generate:

$$q(x) \leftarrow \mathit{Actor}(x)$$

unibz

Query rewriting

To compute the perfect rewriting of a query q, start from q, iteratively get a CQ q' to be processed, and do one of the following:

- Apply to some atom of q' an inclusion assertion in $\mathcal{T}$ as follows:

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

('_' denotes a variable that appears only once)

- Choose two atoms of q' that unify, and apply the unifier to q'.

Each time, the result of the above step is added to the queries to be processed.

Note: Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method [C. *et al.*, 2007a].

The UCQ resulting from this process is the **perfect rewriting** $r_{q,\mathcal{T}}$.

unibz

Query answering in *DL-Lite* – Example

TBox:

$$\begin{aligned} \mathit{Actor} &\sqsubseteq \mathit{Staff} \\ \mathit{Staff} &\sqsubseteq \exists \mathit{worksFor} \\ \exists \mathit{worksFor}^- &\sqsubseteq \mathit{Movie} \end{aligned}$$

Corresponding rules:

$$\begin{aligned} \mathit{Actor}(x) &\rightarrow \mathit{Staff}(x) \\ \mathit{Staff}(x) &\rightarrow \exists y(\mathit{worksFor}(x, y)) \\ \mathit{worksFor}(y, x) &\rightarrow \mathit{Movie}(x) \end{aligned}$$

Query: $q(x) \leftarrow \mathit{worksFor}(x, y), \mathit{Movie}(y)$

Perfect rewriting:

$$\begin{aligned} q(x) &\leftarrow \mathit{worksFor}(x, y), \mathit{Movie}(y) \\ q(x) &\leftarrow \mathit{worksFor}(x, y), \mathit{worksFor}(\_, y) \\ q(x) &\leftarrow \mathit{worksFor}(x, \_) \\ q(x) &\leftarrow \mathit{Staff}(x) \\ q(x) &\leftarrow \mathit{Actor}(x) \end{aligned}$$

ABox: $\mathit{worksFor}(\texttt{'Tom Cruise'}, \texttt{'Minority Report'})$
$\mathit{worksFor}(\texttt{'Stanley Kubrick'}, \texttt{'Eyes Wide Shut'})$
$\mathit{Actor}(\texttt{'Tom Cruise'}), \quad \mathit{Actor}(\texttt{'Nicole Kidman'})$

Evaluating the perfect rewriting over the ABox (seen as a DB) produces as answer $\{\texttt{'Tom Cruise'}, \texttt{'Stanley Kubrick'}, \texttt{'Nicole Kidman'}\}$.

unibz

Canonical model of a *DL-Lite* ontology

The correctness proof of the rewriting based approach exploits the fact that (U)CQs are preserved under homomorphisms (which are mappings that preserve constants and membership to relations).

Canonical model of a *DL-Lite* ontology $\mathcal{O}$

Is a model of $\mathcal{O}$ that has homomorphisms to all models of $\mathcal{O}$.

The following result is due to the fact that *DL-Lite* is **convex**, i.e., cannot express disjunction:

Theorem (Canonical model property)

Every satisfiable *DL-Lite* ontology has a **canonical model**.

Properties of the canonical models of a *DL-Lite* ontology:

- All canonical models are homomorphically equivalent, hence, w.r.t. answering (U)CQs it suffices to consider one canonical model.
- The canonical model is in general infinite.

unibz

Query answering in *DL-Lite* – Canonical model

From the definition of canonical model, and since homomorphisms are closed under composition, we get that:

To compute the certain answer to a UCQ q over a *DL-Lite* ontology $\mathcal{O}$, one could in principle evaluate q over a canonical model $\mathcal{I}_\mathcal{O}$ of $\mathcal{O}$.

- This does not give us directly an algorithm for query answering over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, since $\mathcal{I}_\mathcal{O}$ may be infinite.
- However, one can show that evaluating q over $\mathcal{I}_\mathcal{O}$ amounts to evaluating the perfect rewriting $r_{q,\mathcal{T}}$ over $\mathcal{A}$.

unibz

Complexity of reasoning in *DL-Lite*

	Combined complexity		**Data complexity**	
	IQs	CQs	IQs	CQs
Plain databases	in AC^0	NP-c	in AC^0	in AC^0
DL-Lite	NL-c	NP-c	in AC^0	in AC^0

(IQs stands for instance queries, i.e., queries with a single atom.)

Ontology satisfiability and all classical DL reasoning tasks:

- These can be reduced to query answering (for queries of fixed size).
- Hence, the complexity is the same as for answering IQs.
- In fact, all reasoning tasks are FOL-rewritable, hence can be delegated to a relational database.

unibz

Tracing the expressivity boundary for FOL rewritability

	Lhs concept	Rhs concept	funct.	Relation incl.	Data complexity of query answering
0	*DL-Lite*		$\surd^*$	$\surd^*$	in AC^0
1	$A \mid \exists P.A$	A	–	–	NLogSpace-hard
2	A	$A \mid \forall P.A$	–	–	NLogSpace-hard
3	A	$A \mid \exists P.A$	$\surd$	–	NLogSpace-hard
4	$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTime-hard
5	$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTime-hard
6	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	$\surd$	–	PTime-hard
7	$A \mid \exists P.A \mid \exists P^-.A$	$A \mid \exists P$	–	–	PTime-hard
8	$A \mid \exists P \mid \exists P^-$	$A \mid \exists P \mid \exists P^-$	$\surd$	$\surd$	PTime-hard
9	$A \mid \neg A$	A	–	–	coNP-**hard**
10	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-**hard**
11	$A \mid \forall P.A$	A	–	–	coNP-**hard**

From [C. *et al.*, 2006; Artale *et al.*, 2009; C. *et al.*, 2013].

Note: Data complexity beyond AC^0 means that query answering is **not FOL-rewritable**, hence cannot be delegated to a relational DBMS.

unibz

Query rewriting beyond *DL-Lite*

- The rewriting based approach is in principle applicable to DLs that admit a canonical model.
- Consider the DLs $\mathcal{EL}$ and $\mathcal{ELH}$:
  - $\mathcal{EL}$: inclusions of concepts of the form: $C, C' \longrightarrow \exists P.C \mid C \sqcap C'$
  - $\mathcal{ELH}$: adds to $\mathcal{EL}$ also role inclusions.

  Important for biomedical ontologies (e.g., SNOWMED).
  At the basis of the OWL 2 EL profile of OWL 2.
- Rewritings can be expressed using (recursive) Datalog rules [Krisnadhi and Lutz, 2007; Rosati, 2007]:
  - The query is rewritten into a UCQ similarly to *DL-Lite*.
  - Recursion is introduced to close under the concept/role hierarchy.
- Approach extended also to Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHOIQ}$ [Ortiz *et al.*, 2011] (but more complicated).

unibz

Complexity for lightweight DLs beyond *DL-Lite*

We consider both instance queries (IQs) and (unions of) conjunctive queries (CQs).

	Combined complexity		**Data complexity**	
	IQs	CQs	IQs	CQs
Plain databases	in AC^0	NP-c	in AC^0	in AC^0
DL-Lite	NL-c	NP-c	in AC^0	in AC^0
$\mathcal{EL}$, $\mathcal{ELH}$	PTIME-c	NP-c	PTIME-c	PTIME-c
Horn-$\mathcal{SHIQ}$, Horn-$\mathcal{SHOIQ}$	EXPTIME-c	EXPTIME-c	PTIME-c	PTIME-c

unibz

Outline

unibz

Expressive and very expressive Description Logics

- DLs are considered **expressive** when they
  - are propositionally closed, and
  - allow for arbitrary inclusion axioms.
- Examples of expressive DLs: $\mathcal{ALC}$, $\mathcal{ALCI}$, $\mathcal{ALCQ}$, $\mathcal{ALCQI}$
- Well known: standard reasoning in expressive DLs is:
  - ExpTime-hard in combined complexity (and ExpTime-complete for most expressive DLs),
  - coNP-hard in data complexity.
- **Very expressive DLs** have in addition mechanisms for "non-local" navigation, i.e., one can refer to "distant objects", e.g., via transitive roles, reflexive transitive closure of roles, complex role inclusion axioms, or nominals.

unibz

Query answering in (very) expressive DLs

- Due to disjunction, expressive DLs are not convex, and the canonical model property does not hold.
- To check whether a tuple of individuals is in the answer to a query, one has to **argue model by model**.
- Query answering becomes:
  - coNP-hard in data complexity, and
  - (usually) exponentially more complex in combined complexity.
- It is difficult to process multiple answer tuples at once.
  Hence, instead of query answering, we consider ...

Query entailment

Given an ontology $\mathcal{O}$, a query $q(\vec{x})$, and a tuple $\vec{d}$ of individuals,
check whether $\mathcal{O} \models q(\vec{d})$ holds.

Since we allow for the use of individuals in queries, we can restrict the attention to **boolean queries**, i.e., queries without answer variables.

Query answering in (very) expressive DLs

Several techniques have been proposed to check query entailment in (very) expressive DLs.

Most techniques are based on finding a counterexample to the entailment.

- Reduction do (un)satisfiability [C. *et al.*, 1998; C. *et al.*, 2008; Glimm *et al.*, 2008b]
- Tableaux [Levy and Rousset, 1998; Ortiz *et al.*, 2008]
- Resolution [Hustadt *et al.*, 2004]
- Knots (mosaics) and types [Eiter *et al.*, 2009; Eiter *et al.*, 2012]
- Automata based techniques [C. *et al.*, 2007b; C. *et al.*, 2009; C. *et al.*, 2014]

unibz

The $\mathcal{ALCb}^{\mathsf{Self}}_{\mathsf{reg}}$ family of very expressive DLs

We consider the $\mathcal{ALCb}^{\mathsf{Self}}_{\mathsf{reg}}$ family, comprising $\mathcal{ALC}X\mathcal{b}^{\mathsf{Self}}_{\mathsf{reg}}$, where $X \subseteq \{\mathcal{O}, \mathcal{I}, \mathcal{Q}\}$:

- $\mathcal{O}$: nominals, i.e., concepts with a singleton extension.
- $\mathcal{I}$: inverse roles, to gain complete symmetry in how roles are used.
- $\mathcal{Q}$: qualified number restrictions, to constrain the number of individuals connected via a certain role.

The combination of these three constructs in DLs is notoriously difficult.

Also difficult: spell $\mathcal{ALCb}^{\mathsf{Self}}_{\mathsf{reg}}$. So, we abbreviate it to $\mathcal{Z}$.

We deal here only with $\mathcal{ZIQ}$, which corresponds to $\mathcal{ALCIQb}^{\mathsf{Self}}_{\mathsf{reg}}$.

unibz

The very expressive DL $\mathcal{ZIQ}$

We consider a DL with a very rich concept and role language:

$$\begin{array}{rcl} C, C' & \longrightarrow & A \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \\ & & \forall R.C \mid \exists R.C \mid \geqslant n\, S.C \mid \leqslant n\, S.C \mid \exists S.\mathsf{Self} \\ Q & \longrightarrow & P \mid P^- \\ S, S' & \longrightarrow & Q \mid S \cap S' \mid S \cup S' \mid S \setminus S' \\ R, R' & \longrightarrow & S \mid R \cup R' \mid R \circ R' \mid R^* \mid id(C) \end{array}$$

An ontology is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

- The TBox $\mathcal{T}$ is a set of inclusions between arbitrary concepts: $C_1 \sqsubseteq C_2$
  inclusions between simple roles: $S_1 \sqsubseteq S_2$
- The ABox is a set of membership assertions $A(d)$, and $P(d_1, d_2)$, involving atomic concepts and roles only.

Example: of TBox axioms

$$\begin{array}{rcl} \mathit{Actor} \sqcup \mathit{Footballer} & \sqsubseteq & \mathit{VIP} \\ \exists (\mathit{hasFriend} \circ \mathit{hasRelative})^*.\mathit{VIP} & \sqsubseteq & \mathit{VIP} \\ \mathit{hasFriend} & \sqsubseteq & \mathit{hasFriend}^- \end{array}$$

Positive 2-way regular path queries (P2RPQs)

- 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 a conjunction of several atoms, and called positive if the combination might use conjunction and disjunction.
- 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 the W3C standard query languages XPath and SPARQL.

Example: Is there a couple x and y of married people of different gender that are relatives, and at least one of them is an actor?

$$q = \exists x, y.\ \mathit{Female}(x) \wedge \mathit{Male}(y) \wedge \mathit{married}(x, y) \wedge (\mathit{hasChild} \cup \mathit{hasChild}^-)^*(x, y) \wedge (\mathit{Actor}(x) \vee \mathit{Actor}(y))$$

Checking query entailment via tree automata

Given a $\mathcal{ZIQ}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a boolean P2RPQ q, we want to check whether

$$\mathcal{O} \models q$$

We use an approach based on **tree automata** [C. *et al.*, 2007b]:

1. Construct an automaton $\mathbf{A}_{\mathcal{K}}$ that accepts the models of $\mathcal{K}$.
2. Construct an automaton $\mathbf{A}_{\neg q}$ that accepts the interpretations to which q does not match.
3. Intersect $\mathbf{A}_{\mathcal{K}}$ with $\mathbf{A}_{\neg q}$ and check the resulting automaton for emptiness.

unibz

Making the approach effective

To actually make the approach work, we need to take into account several issues:

1. To deal with the TBox, we first **internalize** it into a single concept.
2. Tree automata accept tree-shaped models $\rightsquigarrow$ The logic must have the **tree-model property**
3. We need automata that are able to "deal with" **inverses** and **number restrictions**.
4. We need to perform complex **automata-theoretic operations** (projection, intersection, complement).

unibz

Internalizing the TBox

1. We eliminate role inclusion assertions [Rudolph *et al.*, 2008]:

$$S_1 \sqsubseteq S_2 \quad \rightsquigarrow \quad \exists(S \setminus S').\top \sqsubseteq \bot$$

2. We eliminate top and bottom role and concept (standard).

3. We internalize the TBox $\mathcal{T}$ into a concept $C_{\mathcal{T}}$:

$$C_{\mathcal{T}} = \forall(\bigcup_{R \text{ in } \mathcal{T}} R)^*.\bigsqcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} (\neg C_1 \sqcup C_2)$$

Internalization for $\mathcal{ZIQ}$ TBoxes

Given a $\mathcal{ZIQ}$ TBox $\mathcal{T}$, we can construct in linear time a $\mathcal{ZIQ}$ concept $C_{\mathcal{T}}$ that preserves satisfiability.

unibz

Quasi-forest models

A **quasi-forest model** of a $\mathcal{ZIQ}$ concept C and ABox $\mathcal{A}$ is an interpretation $\mathcal{I}$ such that:

- The interpretation domain forms a forest, i.e., a set of trees.
- The individuals of $\mathcal{A}$ are the roots of the forest.
- Each atomic role connects either a root and some node (including other nodes), or a node and its predecessor, or a node to itself.
- The roots satisfy the assertions in $\mathcal{A}$, and some root satisfies C.

Quasi-forest model property of $\mathcal{ZIQ}$

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\mathcal{ZIQ}$ ontology. Then, for every P2RPQ q, if $\mathcal{O}$ has a model $\mathcal{I}$ with $\mathcal{I} \not\models q$, then $C_{\mathcal{T}}$ and $\mathcal{A}$ has a quasi-forest model $\mathcal{I}'$ with $\mathcal{I}' \not\models q$.

Note: the quasi-forest model property would **not** hold for $\mathcal{ZOIQ}$.

unibz

2-way alternating tree automata

We use 2-way alternating parity automata over infinite trees (2ATA):

- A 2ATA $\mathbf{A}$ runs over **infinite trees** labeled with symbols of an alphabet Σ.
  $\rightsquigarrow$ Σ contains the atomic concepts and roles of the TBox
- Runs can be infinite, and $\mathbf{A}$ specifies a **parity** acceptance condition on the infinite runs. $\rightsquigarrow$ Elegant mechanism for dealing with termination
- **Alternation** allows $\mathbf{A}$ to move in parallel to different combinations of nodes and states. $\rightsquigarrow$ Decompose complex expressions
- **2-way-ness** allows $\mathbf{A}$ to move up (-1), stay in the current node (0), and/or move to its successors when navigating the tree. $\rightsquigarrow$ Inverses
- $\mathbf{A}$ accepts trees with a dummy root, and where the first level nodes are the ABox individuals.

unibz

2ATA transitions – Example

Example of a transition of a 2ATA $\mathbf{A}$:

$$\delta(q_1, \sigma) = (\ (-1, q_1)\ \wedge\ (0, q_3)\ )\ \vee\ (\ (1, q_2)\ \wedge\ (4, q_1)\ )$$

If $\mathbf{A}$ is in state q_1 and reads at the current node x the label σ, it can either:

- move to the predecessor of x with state q_1 and stay on x with state q_3, or
- move to the 1st successors of x with state q_2 and also the 4th successor of x with state q_1.

unibz

Reducing ontology satisfiability to emptiness of 2ATAs

Given a $\mathcal{ZIQ}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$

1. We internalize the TBox $\mathcal{T}$ into a concept $C_{\mathcal{T}}$.
2. We construct a 2ATA $\mathbf{A}_{\mathcal{O}}$ accepting the quasi-forest models of $C_{\mathcal{T}}$ and $\mathcal{A}$.
   - Alternation is used to inductively decompose the concept expression, mimicking the semantics of the constructors.
   - To check number restrictions, we introduce specific "counting states" and count along successors.
   - The check that the ABox assertions are satisfied is done at the dummy root.

By the quasi-forest model property of $\mathcal{ZIQ}$, we get that $\mathbf{A}_{\mathcal{O}}$ is non-empty iff $\mathcal{O}$ is satisfiable.

Theorem ([C. *et al.*, 2007b])

Ontology satisfiability in $\mathcal{ZIQ}$ is ExpTime-complete.

unibz

Checking query entailment

To check whether $\mathcal{O} \models q$, we proceed as follows:

1. Construct a 2ATA $\mathbf{A}_{\mathcal{O}}$ accepting the quasi-forest models of $\mathcal{O}$.
2. Construct a 2ATA $\mathbf{A}_q$ accepting the quasi-forest interpretations to which q matches.
3. Complement $\mathbf{A}_q$, to obtain $\mathbf{A}_{\neg q}$.
4. Intersect $\mathbf{A}_{\mathcal{O}}$ with $\mathbf{A}_{\neg q}$, to obtain an automaton $\mathbf{A}_{\mathcal{O} \not\models q}$ accepting models of $\mathcal{O}$ to which q does not match.
5. Check $\mathbf{A}_{\mathcal{O} \not\models q}$ for emptiness.

Note: Steps 2 and 3 require to first consider the query variables as if they were individuals, and then **project** them away. This leads to an exponential blowup, which is, however, unavoidable.

unibz

Complexity of query entailment in $\mathcal{ZIQ}$

Theorem

Given a $\mathcal{ZIQ}$ ontology $\mathcal{O}$ and a P2PRQ q, deciding whether $\mathcal{O} \models q$ is in 2ExpTime in the total size of q and $\mathcal{O}$.

- The result holds when numbers in number restrictions are encoded in unary.
- Modulo coding of numbers, the bound is optimal [Lutz, 2008; Eiter *et al.*, 2009].
- There seems to be no easy way to adapt the automata-based technique to obtain optimal bounds in data-complexity.
- The approach can be extended also to expressive DLs with nominals (using a more powerful automata model), as long as the DL satisfies the tree model property.

Conjunctive query entailment is still open for ontologies expressed in OWL 2.

Outline

unibz

Conclusions

- We have considered the challenging problem of answering complex queries over Description Logic ontologies.
- A variety of techniques have been proposed, depending on the expressive power of the considered DL. We have considered:
  - a rewriting-based approach for lightweight DLs
  - an automata-based approach for very expressive DLs.
- We have not discussed implementation of the techniques:
  - For the lightweight DLs of the *DL-Lite* family, substantial implementation efforts are ongoing.
  - Implementation efforts are carried out in the context of ontology based data access.
  - For expressive DLs, implementors have essentially given up: SPARQL 1.1 has weakened the semantics of existential variables, so that CQs cannot be expressed.

unibz

Further research directions and open issues

- Several results are still open, especially regarding expressive DLs:
  - decidability of query entailment in OWL 2 (i.e., $\mathcal{SROIQ}$)
  - decidability of satisfiability in $\mathcal{ZOIQ}$
  - data complexity for very expressive DLs
- Work is ongoing on more expressive query languages, e.g., nested RPQs, query languages with fixpoints
- A promising novel research direction is considering the non-uniform approach: i.e., study the complexity for a specific TBox, rather than for a TBox language.
- OBDA: management of mappings and query processing w.r.t. mappings..
- User-friendly ontology querying modalities (graphical query languages, natural language querying).

unibz

Thanks

Thank you for your attention!

. . . and thanks to many people who contributed to this work:

- Alessandro Artale (FUB)
- Giuseppe De Giacomo (Uniroma1)
- Thomas Eiter (TU Wien)
- Roman Kontchakov (Birkbeck)
- Domenico Lembo (Uniroma1)
- Maurizio Lenzerini (Uniroma1)
- Magdalena Ortiz (TU Wien)
- Riccardo Rosati (Uniroma1)
- Michael Zakharyaschev (Birkbeck)

EU IP Project

Optique

(Nov. 2012 – Oct. 2016)

unibz

References I

[Artale *et al.*, 2009] Alessandro Artale, Diego C., Roman Kontchakov, and Michael Zakharyaschev.

The *DL-Lite* family and relations.

J. of Artificial Intelligence Research, 36:1–69, 2009.

[Berardi *et al.*, 2005] Daniela Berardi, Diego C., and Giuseppe De Giacomo.

Reasoning on UML class diagrams.

Artificial Intelligence, 168(1–2):70–118, 2005.

[Bergamaschi and Sartori, 1992] Sonia Bergamaschi and Claudio Sartori.

On taxonomic reasoning in conceptual design.

ACM Trans. on Database Systems, 17(3):385–422, 1992.

[Borgida and Brachman, 2003] Alexander Borgida and Ronald J. Brachman.

Conceptual modeling with description logics.

In Franz Baader, Diego C., Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 10, pages 349–372. Cambridge University Press, 2003.

unibz

References II

[Borgida, 1995] Alexander Borgida.

Description logics in data management.

IEEE Trans. on Knowledge and Data Engineering, 7(5):671–682, 1995.

[C. *et al.*, 1998] Diego C., 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.

[C. *et al.*, 1999] Diego C., Maurizio Lenzerini, and Daniele Nardi.

Unifying class-based representation formalisms.

J. of Artificial Intelligence Research, 11:199–240, 1999.

[C. *et al.*, 2006] Diego C., 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.

unibz

References III

[C. *et al.*, 2007a] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family.

J. of Automated Reasoning, 39(3):385–429, 2007.

[C. *et al.*, 2007b] Diego C., Thomas Eiter, and Magdalena Ortiz.

Answering regular path queries in expressive description logics: An automata-theoretic approach.

In *Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI)*, pages 391–396, 2007.

[C. *et al.*, 2008] Diego C., 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.

[C. *et al.*, 2009] Diego C., Thomas Eiter, and Magdalena Ortiz.

Regular path queries in expressive description logics with nominals.

In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 714–720, 2009.

unibz

References IV

[C. *et al.*, 2013] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

Data complexity of query answering in description logics.

Artificial Intelligence, 195:335–360, 2013.

[C. *et al.*, 2014] Diego C., Thomas Eiter, and Magdalena Ortiz.

Answering regular path queries in expressive description logics via alternating tree-automata.

Information and Computation, 237:12–55, 2014.

[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.

[Eiter *et al.*, 2008] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus.

Query answering in the description logic Horn-$\mathcal{SHIQ}$.

In *Proc. of the 11th Eur. Conference on Logics in Artificial Intelligence (JELIA)*, pages 166–179, 2008.

unibz

References V

[Eiter *et al.*, 2009] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus.
Query answering in description logics with transitive roles.
In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 759–764, 2009.

[Eiter *et al.*, 2012] Thomas Eiter, Magdalena Ortiz, and Mantas Simkus.
Conjunctive query answering in the description logic $\mathcal{SH}$ using knots.
J. of Computer and System Sciences, 78(1):47–85, 2012.

[Glimm *et al.*, 2007] Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler.
Conjunctive query answering for the description logic $\mathcal{SHIQ}$.
In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 399–404, 2007.

[Glimm *et al.*, 2008a] Birte Glimm, Ian Horrocks, and Ulrike Sattler.
Unions of conjunctive queries in $\mathcal{SHOQ}$.
In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pages 252–262, 2008.

[Glimm *et al.*, 2008b] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler.
Conjunctive query answering for the description logic $\mathcal{SHIQ}$.
J. of Artificial Intelligence Research, 31:151–198, 2008.

unibz

References VI

[Hustadt *et al.*, 2004] Ullrich Hustadt, Boris Motik, and Ulrike Sattler.

A decomposition rule for decision procedures by resolution-based calculi.

In *Proc. of the 11th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 21–35, 2004.

[Krisnadhi and Lutz, 2007] Adila Krisnadhi and Carsten Lutz.

Data complexity in the $\mathcal{EL}$ family of description logics.

In *Proc. of the 14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 333–347, 2007.

[Lenzerini and Nobili, 1990] Maurizio Lenzerini and Paolo Nobili.

On the satisfiability of dependency constraints in entity-relationship schemata.

Information Systems, 15(4):453–461, 1990.

[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.

unibz

References VII

[Lutz, 2008] Carsten Lutz.

The complexity of conjunctive query answering in expressive description logics.

In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 179–193. Springer, 2008.

[Ortiz *et al.*, 2006] Maria Magdalena Ortiz, Diego C., 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 C., and Thomas Eiter.

Data complexity of query answering in expressive description logics via tableaux.

J. of Automated Reasoning, 41(1):61–98, 2008.

[Ortiz *et al.*, 2011] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus.

Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$.

In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1039–1044. IJCAI/AAAI, 2011.

unibz

References VIII

[Queralt *et al.*, 2012] Anna Queralt, Alessandro Artale, Diego C., and Ernest Teniente.

OCL-Lite: Finite reasoning on UML/OCL conceptual schemas.

Data and Knowledge Engineering, 73:1–22, 2012.

[Rosati, 2007] Riccardo Rosati.

On conjunctive query answering in $\mathcal{EL}$.

In *Proc. of the 20th Int. Workshop on Description Logic (DL)*, volume 250 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, pages 451–458, 2007.

[Rudolph *et al.*, 2008] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler.

Cheap boolean role constructors for description logics.

In *Proc. of the 11th Eur. Conference on Logics in Artificial Intelligence (JELIA)*, volume 5293 of *Lecture Notes in Computer Science*, pages 362–374. Springer, 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.

unibz