Ontology-based Data Management: Present and Future

Maurizio Lenzerini

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

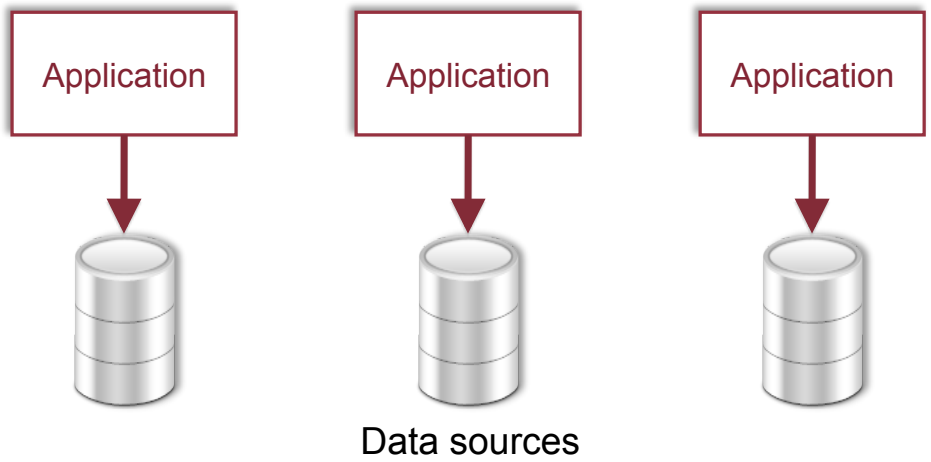
SAPIENZA
UNIVERSITÀ DI ROMA

13th International Conference on Principles of Knowledge Representation and Reasoning

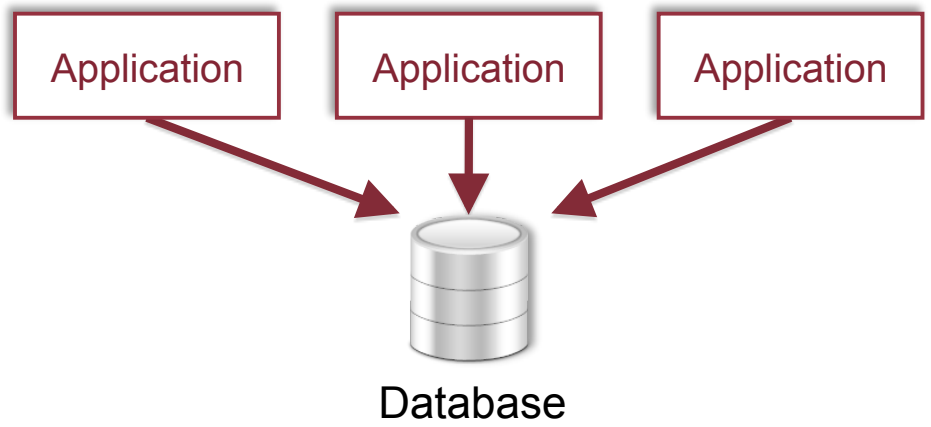
Roma, Italy, June 10 – 14, 2012

Information system architecture enabled by DBMS

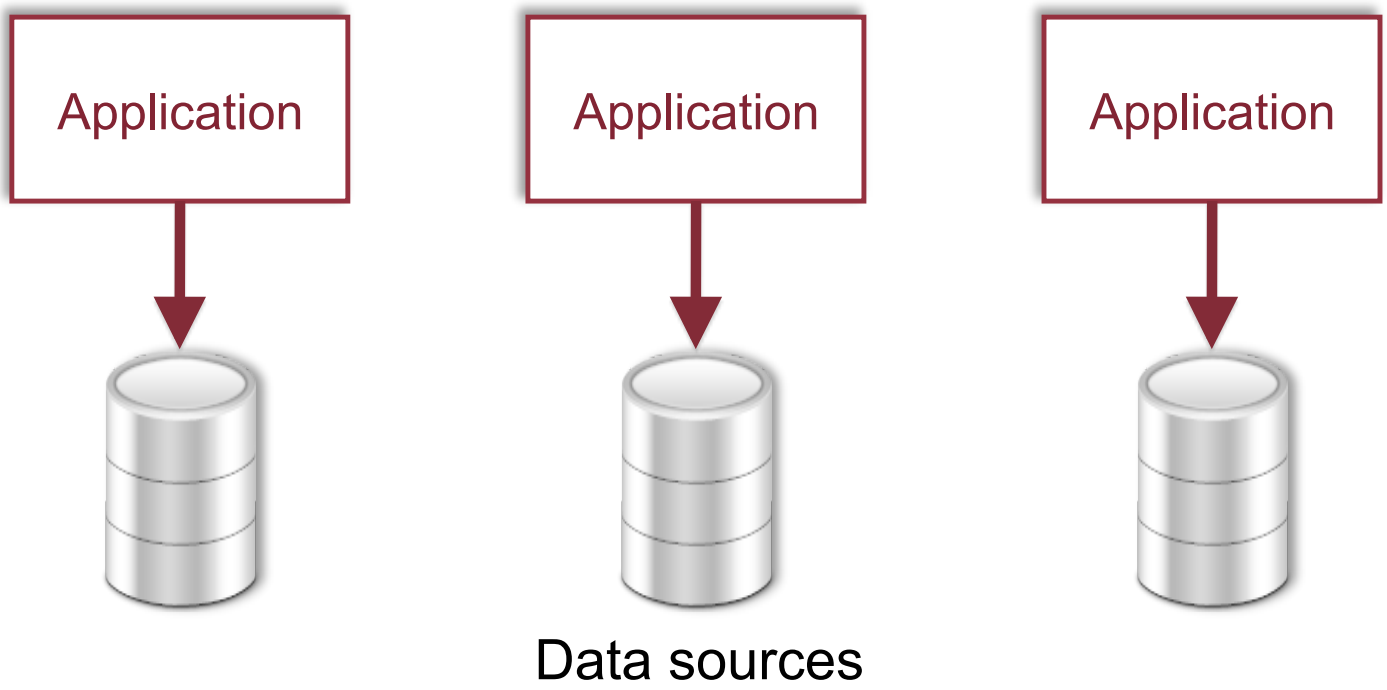
Pre-DBMS architecture (need of a unified data storage):



"Ideal information system architecture" with DBMS ('70s):



Today in many organizations ...



- Distributed, redundant, application-dependent, and mutually incoherent data
- Desperate need of a coherent, conceptual, unified view of data

... even with just one data source

Fragment of a relational table in a Bank Information system:

CUC	TS_START	TS_END	ID_GRUP	FLAG_CP	FLAG_CF	FATTURATO	FLAG_FATT
124589	30-lug-2004	1-gen-9999	92736	S	N	195000,00	N
140904	15-mag-2001	15-giu-2005	35060	N	N	230600,00	N
124589	5-mag-2001	30-lug-2004	92736	N	S	195000,00	S
-452901	13-mag-2001	27-lug-2004	92770	S	N	392000,00	N
129008	10-mag-2001	1-gen-9999	62010	N	S	247000,00	S
-472900	10-mag-2001	1-gen-9999	62010	S	N	0 00	N
130976	7-mag-2001	9-lug-2003	75680				

... even with just one data source

Negative value denotes a holding

CUC	TS_START	TS_END	ID_GRUP	FLAG_CP	FLAG_CF	FATTURATO	FLAG_FATT	
124589	30-lug-2004	1-gen-9999	92736	S	N	195000,00	N	
140904	15-mag-2001	15-giu-2005	35060	N	N	230600,00	N	
124589	5-mag-2001	30-lug-2004	92736	N	S	195000,00	S	
-452901	13-mag-2001	27-lug-2004	92770	S	N	392000,00	N	
129008	10-mag-2001	1-gen-9999	62010	N	S	247000,00	S	
-472900	10-mag-2001	1-gen-9999	62010	S	N	0 00	N	
130976	7-mag-2001	9-lug-2003	75680					

... even with just one data source

S means that the customer is the leader of the group it belongs to

S means that the customer is the head of the group it belongs to

CUC	TS_START	TS_END	ID_GRUP	FLAG_CP	FLAG_CF	FATTURATO	FLAG_FATT
124589	30-lug-2004	1-gen-9999	92736	S	N	195000,00	N
140904	15-mag-2001	15-giu-2005	35060	N	N	230600,00	N
124589	5-mag-2001	30-lug-2004	92736	N	S	195000,00	S
-452901	13-mag-2001	27-lug-2004	92770	S	N	392000,00	N
129008	10-mag-2001	1-gen-9999	62010	N	S	247000,00	S
-472900	10-mag-2001	1-gen-9999	62010	S	N	0 00	N
130976	7-mag-2001	9-lug-2003	75680				

... even with just one data source

N means that the FATTURATO field is not valid

CUC	TS_START	TS_END	ID_GRUP	FLAG_CP	FLAG_CF	FATTURATO	FLAG_FATT
124589	30-lug-2004	1-gen-9999	92736	S	N	195000,00	N
140904	15-mag-2001	15-giu-2005	35060	N	N	230600,00	N
124589	5-mag-2001	30-lug-2004	92736	N	S	195000,00	S
-452901	13-mag-2001	27-lug-2004	92770	S	N	392000,00	N
129008	10-mag-2001	1-gen-9999	62010	N	S	247000,00	S
-472900	10-mag-2001	1-gen-9999	62010	S	N	0 00	N
130976	7-mag-2001	9-lug-2003	75680				

Information integration

From [Bernstein & Haas, CACM Sept. 2008]:

- Large enterprises spend a great deal of time and money on information integration (e.g., 40% of information-technology shops' budget).
- Market for information integration software estimated to grow from \$1.87 billion in 2011 to \$2.79 billion in 2015 (+15% per year) [Gartner, 2012]
- Data integration is a large and growing part of software development, computer science, and specific applications settings, such as scientific computing, semantic web, "big data" processing etc..

Basing the information system on a clean, rich and abstract conceptual representation of the data has always been both a goal and a challenge [Mylopoulos et al 1984]

Ontology-based data management: our program

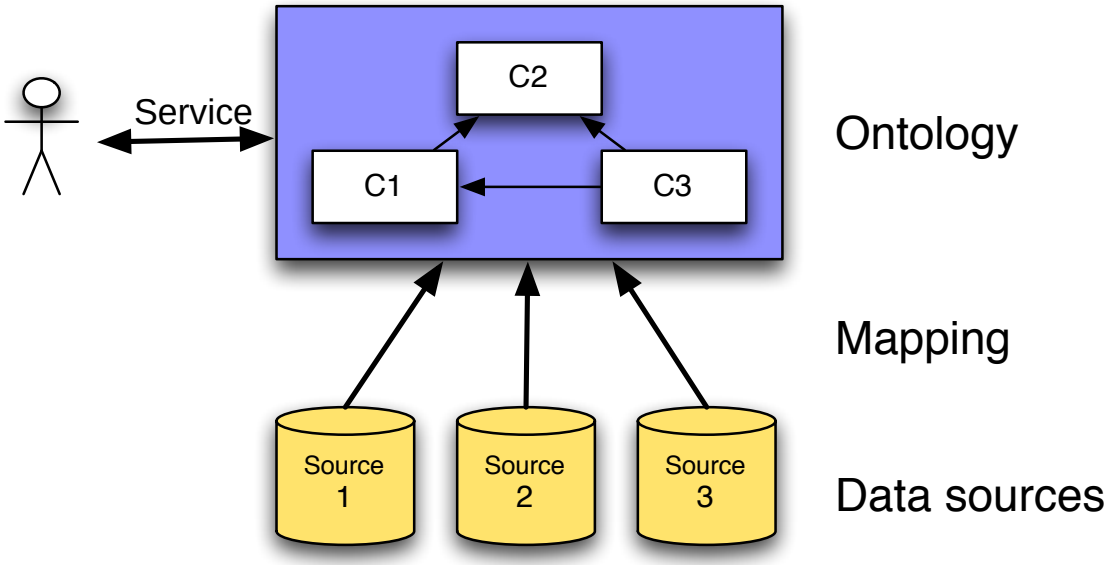
Use Knowledge Representation and Reasoning principles and techniques for a new way of managing data.

- Leave the data where they are
- Build a conceptual specification of the domain of interest, in terms of knowledge structures
- Map such knowledge structures to concrete data sources
- Express all services over the abstract representation
- Automatically translate knowledge services to data services

Experiment techniques in real-world settings

- Logistic (2007)
- Bank (2009)
- Public Administration (2010)
- Telecom (2011)

Ontology-based data management: architecture



Based on three main components:

- **Ontology**, a declarative specification of the domain of interest, used as a unified, conceptual view for clients.
- **Data sources**, representing external, independent, heterogeneous, storage (or, more generally, computational) structures.
- **Mappings**, used to semantically link data at the sources to the ontology.

Outline

Outline

Formal framework of ontology-based data management

An ontology-based data management system is a triple $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{O}$ is the ontology, expressed as TBox in a DL
- $\mathcal{S}$ is a federated relational database representing the sources
- $\mathcal{M}$ is a set of GLAV mapping assertions, each one of the form

$$\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$$

where

- $\Phi(\vec{x})$ is a FOL query over $\mathcal{S}$, returning values for $\vec{x}$
- $\Psi(\vec{x})$ is a FOL query over $\mathcal{O}$, whose free variables are from $\vec{x}$.

Semantics

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for the ontology $\mathcal{O}$.

Def.: Semantics

$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a **model** of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ if:

- $\mathcal{I}$ is a model of $\mathcal{O}$;
- $\mathcal{I}$ satisfies $\mathcal{M}$ wrt $\mathcal{S}$, i.e., satisfies every assertion in $\mathcal{M}$ wrt $\mathcal{S}$.

Def.: Mapping satisfaction (sound mappings)

We say that $\mathcal{I}$ satisfies $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$ wrt a database $\mathcal{S}$, if the sentence

$$\forall \vec{x} \; (\Psi(\vec{x}) \rightarrow \Psi(\vec{x}))$$

is true in $\mathcal{I} \cup \mathcal{S}$.

Which languages?

- Which **language** for expressing the ontology?
  - We use Description Logics, but which one?
- Which **language** for expressing the mappings?
- Which **language** for expressing services (i.e., queries) over the ontology?

Challenge: optimal compromise between expressive power and data complexity.

Abstracting from the mapping

In this talk, we abstract from the mapping, because we want to focus on ontologies.

We assume that $\mathcal{M}$ is a GAV mapping, and we denote by $\mathcal{M}(\mathcal{S})$ the database obtained by "transferring" the data from the sources to the alphabet of the ontology.

$\mathcal{M}(\mathcal{S})$ can be seen as a set of facts built on the alphabet of $\mathcal{O}$ (i.e., a set of ground atomic formulas in logic, or simply, an ABox, in DL terminology).

In practice, to obtain a query over $\mathcal{S}$ from a query over $\mathcal{M}(\mathcal{S})$, we can rewrite the query based on $\mathcal{M}$.

Ontology-based data management (OBDM): topics

- *Ontology-based data access and integration (OBDA)*
  - query answering
  - inconsistency tolerant query answering
- *Ontology-based data quality (OBDQ)*
- *Ontology-based data governance (OBDG)*
- *Ontology-based data restructuring (OBDR)*
- *Ontology-based business intelligence (OBBI)*
- *Ontology-based data update (OBDU)*
- *Ontology-based service and process management (OBDS)*
- *Ontology-based data exchange and coordination (OBDC)*

General requirements:

- large data collections
- efficiency with respect to size of data (data complexity)

Outline

Ontology-based data access: queries

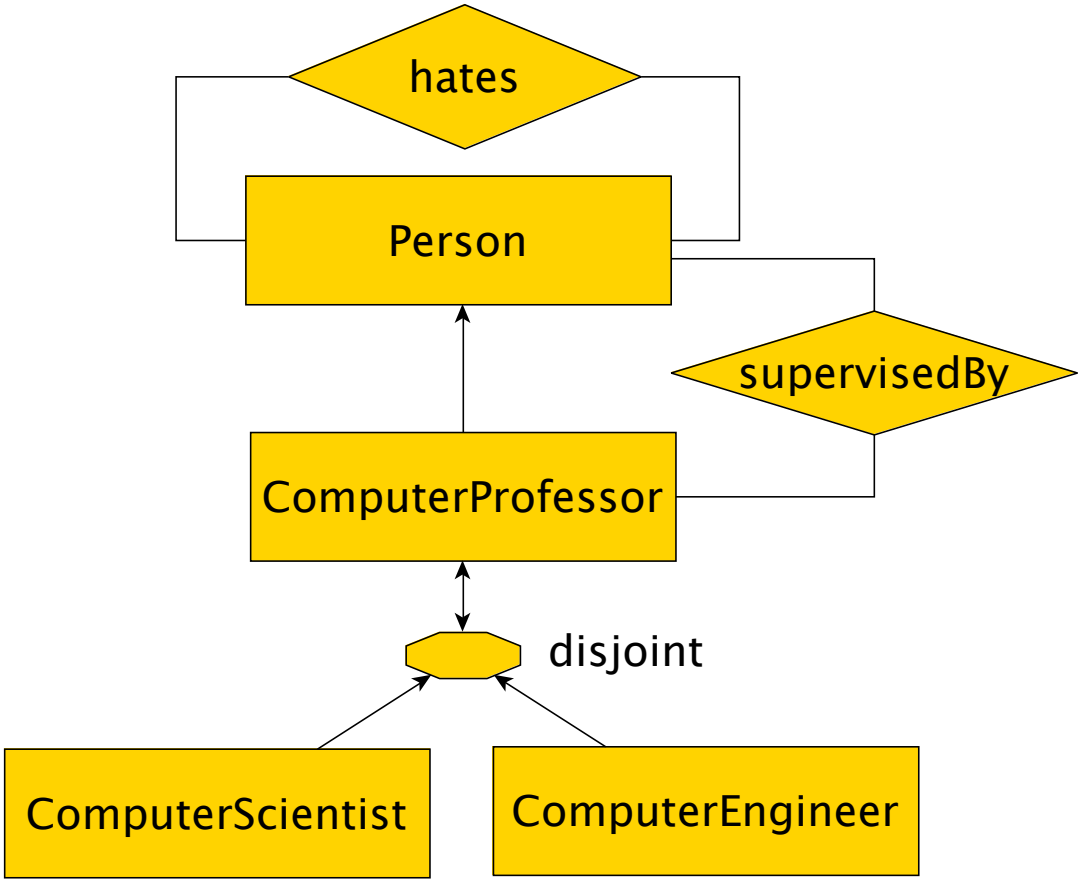
In principle, we are interested in First-order logic (FOL), which is the standard query language for databases. Mostly, we consider **conjunctive queries (CQ)**, i.e., queries of the form (Datalog notation)

$$q(\vec{x}) \leftarrow R_1(\vec{x}, \vec{y}), \ldots, R_k(\vec{x}, \vec{y})$$

where the lhs is the query head, the rhs is the body, and each $R_i(\vec{x}, \vec{y})$ is an atom using (some of) the free variables $\vec{x}$, the existentially quantified variables $\vec{y}$, and possibly constants.

- CQs contain no disjunction, no negation, no universal quantification.
- Correspond to SQL/relational algebra **select-project-join (SPJ) queries** – the most frequently asked queries.
- They can also be written as **SPARQL** queries.
- A Union of CQs (UCQ) is a set of CQs with the same head predicate.

Example of query



$$q(x) \leftarrow \text{supervisedBy}(x, y), \text{ComputerScientist}(y), \text{hates}(y, z), \text{ComputerEngineering}(z)$$

Semantics of queries: certain answers

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for the ontology $\mathcal{O}$.

Def.: Semantics

$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a **model** of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ if:

- $\mathcal{I}$ is a model of $\mathcal{O}$;
- $\mathcal{I}$ satisfies $\mathcal{M}$ wrt $\mathcal{S}$, i.e., satisfies every assertion in $\mathcal{M}$ wrt $\mathcal{S}$.

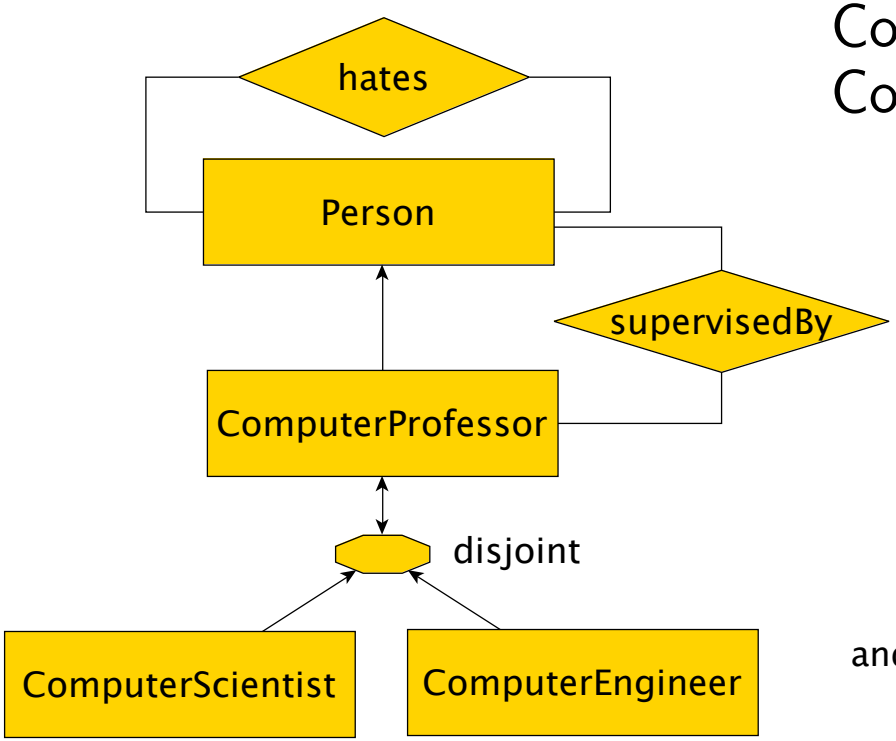
Def.: The certain answers to a query $q(\vec{x})$ over $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$

$$cert(q, \mathcal{K}) = \{ \vec{c}^{\mathcal{I}} \mid \vec{c}^{\mathcal{I}} \in q^{\mathcal{I}} \text{ for every model } \mathcal{I} \text{ of } \mathcal{K} \}$$

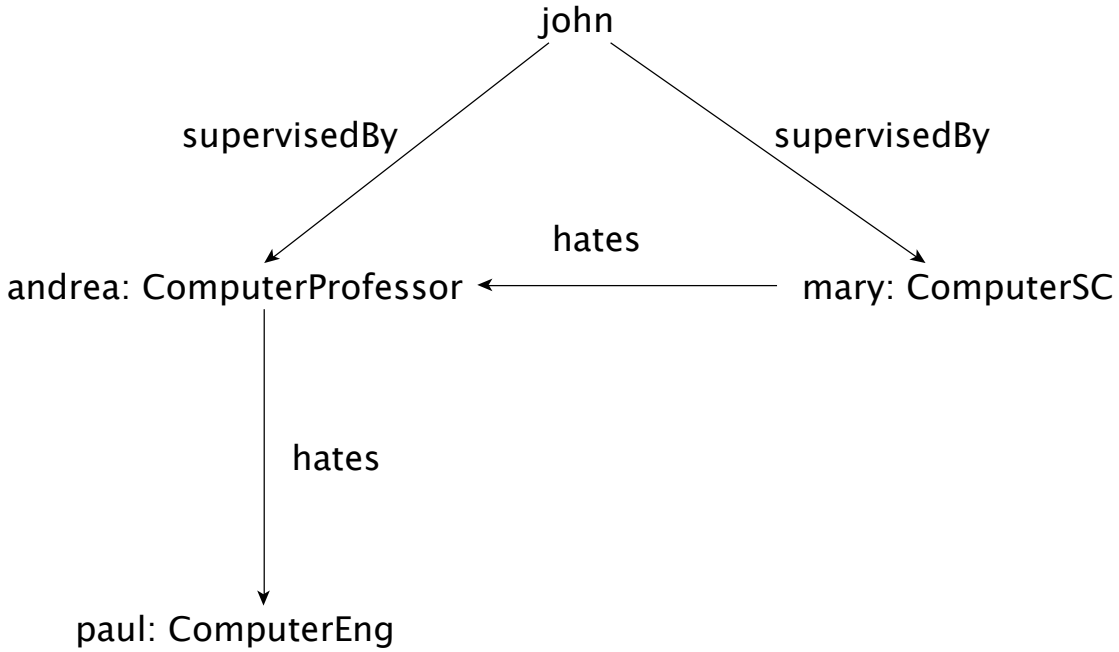
Query language for user queries

- Answering FOL queries is **undecidable**, even if the ontology is empty, and the set of mappings is empty.
- Unions of conjunctive queries (UCQs) do not suffer from this problem.
- We can go beyond unions of conjunctive queries without falling into undecidability, but we get intractability in data complexity very soon.

QA in OBDA – Example(*)

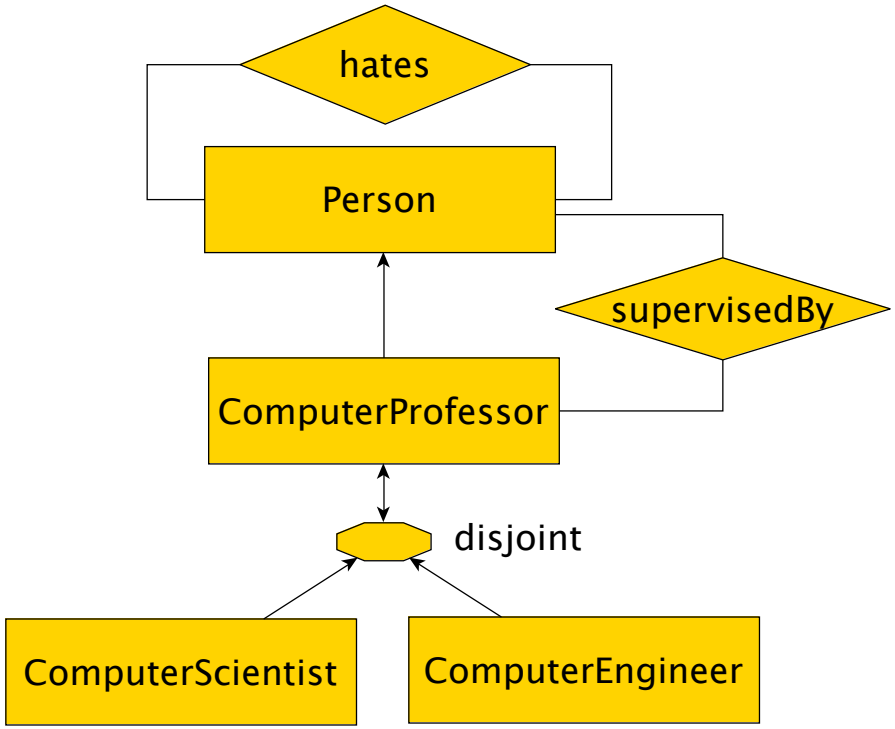


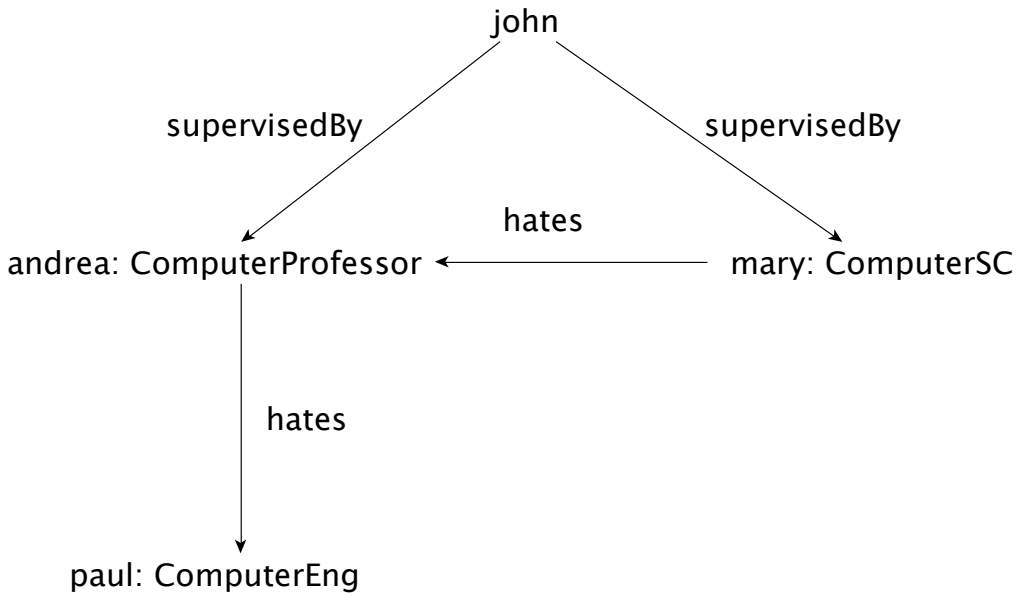
ComputerProfessor is **partitioned into** ComputerScientist and ComputerEngineer.



(*) [Andrea Schaerf 1993]

QA in OBDA – Example (cont'd)

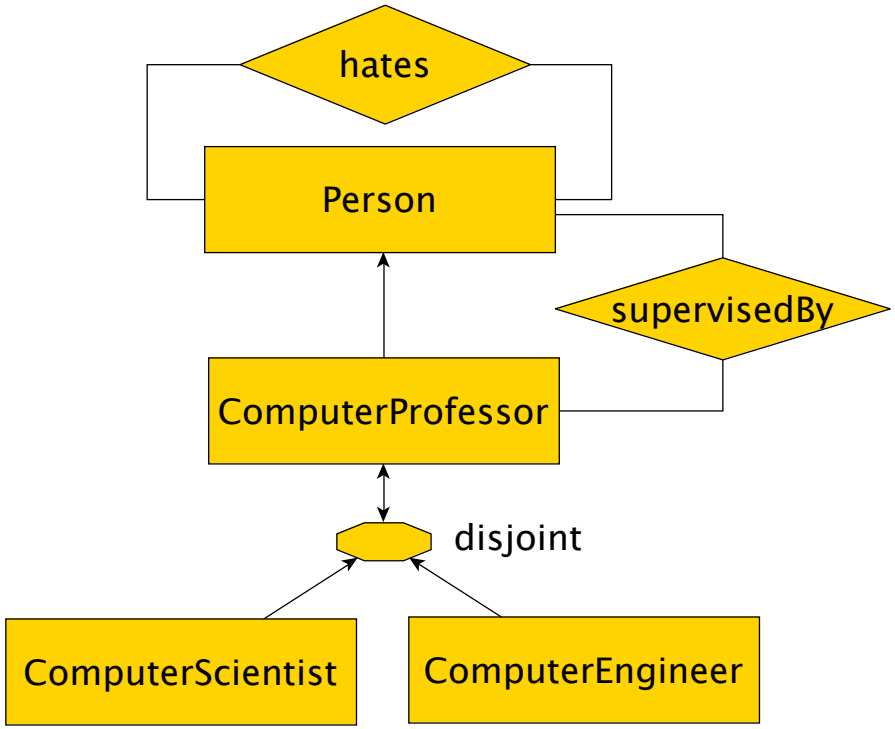


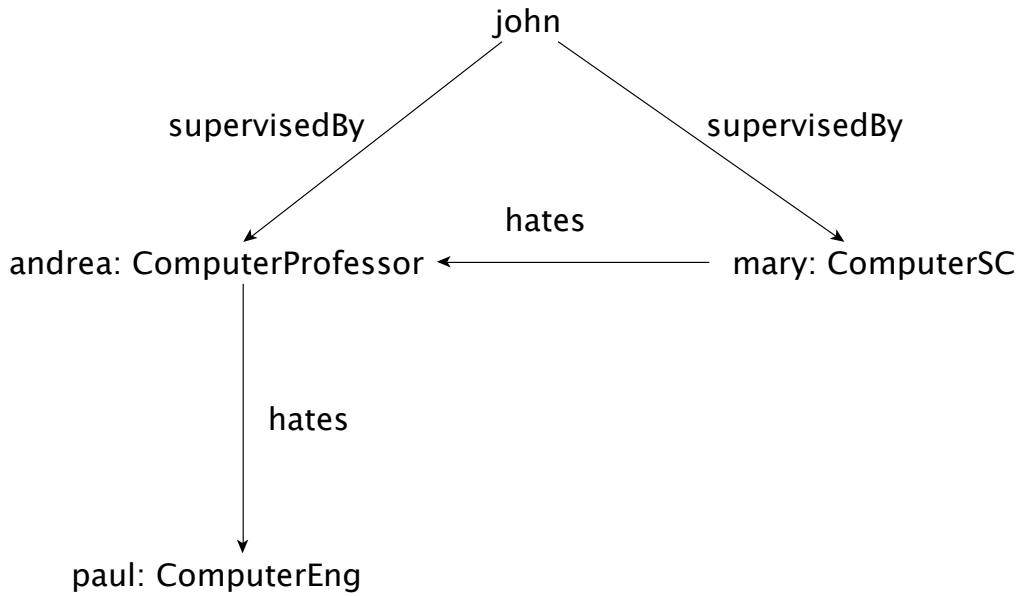


$q(x) \leftarrow \text{supervisedBy}(x, y), \text{ComputerScientist}(y),$
$\text{hates}(y, z), \text{ComputerEngineer}(z)$

Answer: ???

QA in OBDA – Example (cont'd)





$q(x) \leftarrow \text{supervisedBy}(x, y), \text{ComputerScientist}(y),$
$\text{hates}(y, z), \text{ComputerEngineer}(z)$

Answer: { `john` }

To determine this answer, we need to resort to **reasoning by cases** on the instances.

Complexity of conjunctive query answering in DLs

	Combined complexity	Data complexity
Plain databases	NP-complete	in LogSpace (1)
OWL 2 (and less)	?	coNP-hard (2)

(1) Going beyond probably means not scaling with the data.
(2) Already for a TBox with a single disjunction (see example above).

Questions

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

Complexity of conjunctive query answering in DLs

	Combined complexity	Data complexity
Plain databases	NP-complete	in LogSpace (1)
OWL 2 (and less)	?	coNP-hard (2)

(1) Going beyond probably means not scaling with the data.

(2) Already for a TBox with a single disjunction (see example above).

Questions

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

Complexity of conjunctive query answering in DLs

	Combined complexity	Data complexity
Plain databases	NP-complete	in LogSpace (1)
OWL 2 (and less)	?	coNP-hard (2)

(1) Going beyond probably means not scaling with the data.
(2) Already for a TBox with a single disjunction (see example above).

Questions

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

Complexity of conjunctive query answering in DLs

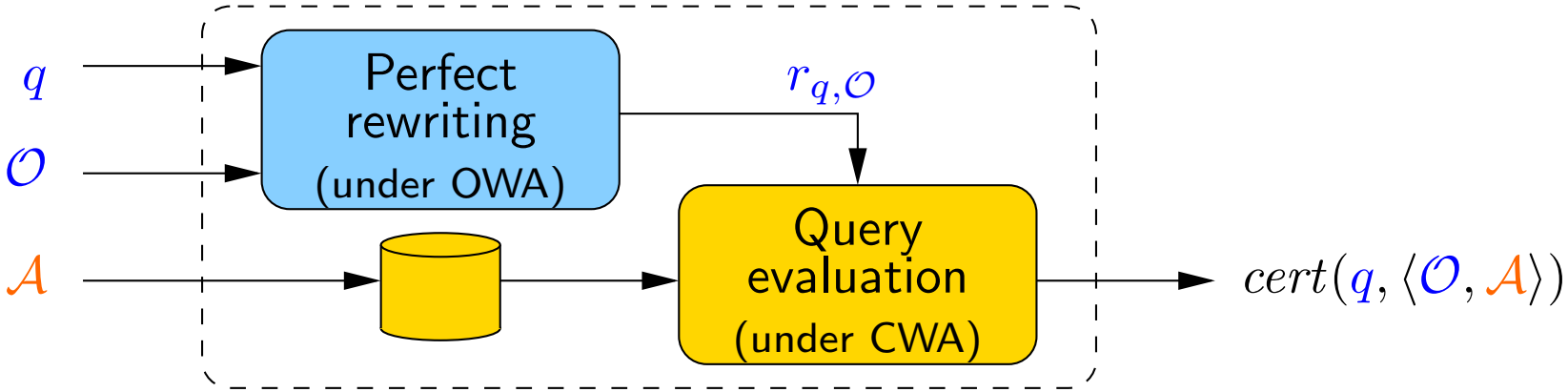
	Combined complexity	Data complexity
Plain databases	NP-complete	in LogSpace (1)
OWL 2 (and less)	?	coNP-hard (2)

(1) Going beyond probably means not scaling with the data.
(2) Already for a TBox with a single disjunction (see example above).

Questions

- Can we find interesting DLs for which the query answering problem can be solved efficiently (in LogSpace wrt data complexity)?
- If yes, can we leverage relational database technology for query answering in OBDA?

Query rewriting



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

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

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

$\mathcal{Q}$-rewritability

Let $\mathcal{Q}$ be a class of queries (or query language) and $\mathcal{L}$ an ontology language.

Def.: $\mathcal{Q}$-rewritability

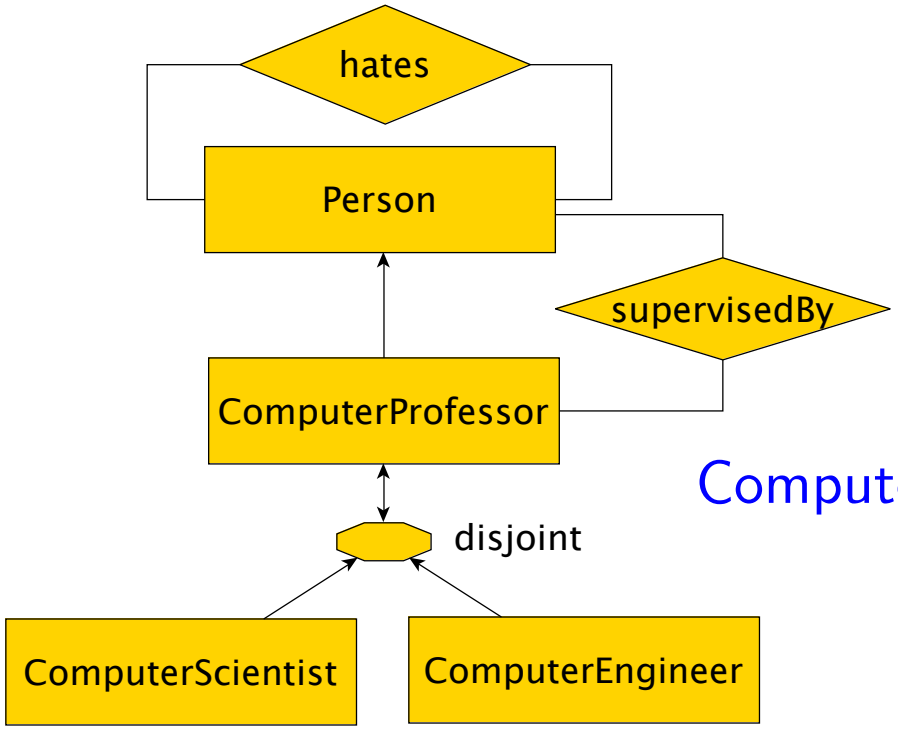
For an ontology language $\mathcal{L}$, query answering is **$\mathcal{Q}$-rewritable** if for every TBox $\mathcal{O}$ of $\mathcal{L}$ and for every query q, the perfect rewriting $r_{q,\mathcal{O}}$ of q w.r.t. $\mathcal{O}$ can be expressed in the query language $\mathcal{Q}$.

Language of the rewriting

The expressiveness of the ontology language affects the **query language into which we are able to rewrite CQs**:

- When we can rewrite into **UCQ**.
  $\rightsquigarrow$ Query evaluation can be "optimized" via RDBMS
- When we can rewrite into **FOL/SQL**.
  $\rightsquigarrow$ Query evaluation can be done in SQL, i.e., via RDBMS
- When we can rewrite into **non recursive Datalog**.
  $\rightsquigarrow$ Query evaluation can be still done via RDBMS, but with subqueries/views
- When we can rewrite into an NLogSpace-hard language.
  $\rightsquigarrow$ Query evaluation requires (at least) **linear recursion**.
- When we can rewrite into a PTime-hard language.
  $\rightsquigarrow$ Query evaluation requires full recursion (e.g., **Datalog**).
- When we can rewrite into a coNP-hard language.
  $\rightsquigarrow$ Query evaluation requires (at least) **Disjunctive Datalog**.

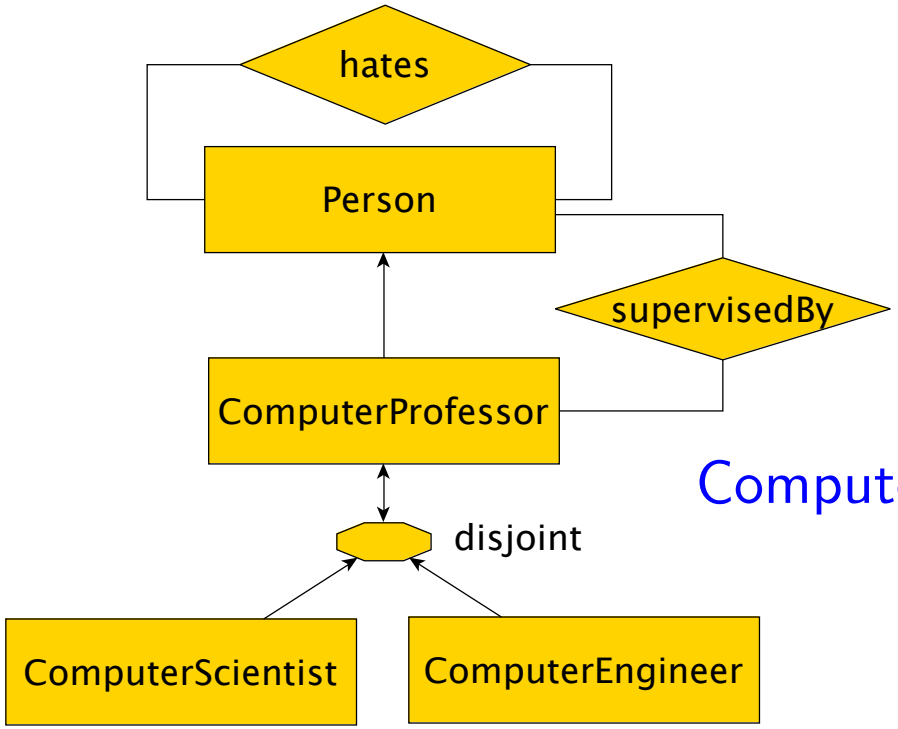
QA in OBDA – Example



$\mathsf{ComputerScientist} \sqsubseteq \forall \mathsf{hates}.\mathsf{ComputerScientist}$

$$q(x) \leftarrow \mathsf{ComputerScientist}(x)$$

QA in OBDA – Example



$\mathsf{ComputerScientist} \sqsubseteq \forall \mathsf{hates}.\mathsf{ComputerScientist}$

$q(x) \leftarrow \mathsf{ComputerScientist}(x)$

Perfect rewriting: $q(x) \leftarrow \mathsf{ComputerScientist}(x)$

$q(x) \leftarrow \mathsf{ComputerScientist}(y), \mathsf{hates}^{+}(y, x)$

Complexity of query answering in DLs

Questions

- Can we find interesting DLs for which query answering is FOL-rewritable?
- Even more specifically, can we find interesting DLs for which query answering is UQC-rewritable?

If yes, we can indeed leverage relational database technology for query answering in OBDA (RDBMs are generally very good at optimizing UCQs).

Our proposal: DL-Lite$_{A,id}$

DL-Lite$_{A,id}$ is the most expressive logic in the *DL-Lite* family

Expressions in *DL-Lite*$_{A,id}$:

$$B \longrightarrow A \mid \exists Q \mid \delta(U) \qquad E \longrightarrow \rho(U) \qquad C \longrightarrow B \mid \neg B$$
$$Q \longrightarrow P \mid P^- \qquad V \longrightarrow U \mid \neg U \qquad R \longrightarrow Q \mid \neg Q$$
$$T \longrightarrow \top_D \mid T_1 \mid \cdots \mid T_n$$

Assertions in *DL-Lite*$_{A,id}$:

$B \sqsubseteq C$	(*concept inclusion*)	$E \sqsubseteq T$	(*value-domain inclusion*)
$Q \sqsubseteq R$	(*role inclusion*)	$U \sqsubseteq V$	(*attribute inclusion*)
$(id\ B\ \pi_1, ..., \pi_n)$	(*identification assertions*)	(**funct** Q)	(*role functionality*)
(**funct** U)	(*attribute functionality*)		

In identification and functional assertions, roles and attributes cannot specialized, and each π_i denotes a *path* (with at least one path with length 1), which is an expression built according to the following syntax rule:

$$\pi \longrightarrow S \mid B? \mid \pi_1 \circ \pi_2$$

Semantics of DL-Lite$_{A,id}$

Construct	Syntax	Example	Semantics
atomic conc.	A	Doctor	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
exist. restr.	$\exists Q$	$\exists$child$^-$	$\{d \mid \exists e.\, (d,e) \in Q^{\mathcal{I}}\}$
at. conc. neg.	$\neg A$	$\neg$Doctor	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
conc. neg.	$\neg\exists Q$	$\neg\exists$child	$\Delta^{\mathcal{I}} \setminus (\exists Q)^{\mathcal{I}}$
atomic role	P	child	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse role	P^-	child$^-$	$\{(o,o') \mid (o',o) \in P^{\mathcal{I}}\}$
role negation	$\neg Q$	$\neg$manages	$(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$
conc. incl.	$B \sqsubseteq C$	Father $\sqsubseteq \exists$child	$B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
role incl.	$Q \sqsubseteq R$	hasFather $\sqsubseteq$ child$^-$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
funct. asser.	(**funct** Q)	(**funct** succ)	$\forall d,e,e'.(d,e) \in Q^{\mathcal{I}} \wedge (d,e') \in Q^{\mathcal{I}} \rightarrow e = e'$
mem. asser.	$A(c)$	Father(bob)	$c^{\mathcal{I}} \in A^{\mathcal{I}}$
mem. asser.	$P(c_1, c_2)$	child(bob, ann)	$(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in P^{\mathcal{I}}$

DL-Lite$_{A,id}$ (as all DLs of the *DL-Lite* family) adopts the Unique Name Assumption (UNA), i.e., different individuals denote different objects.

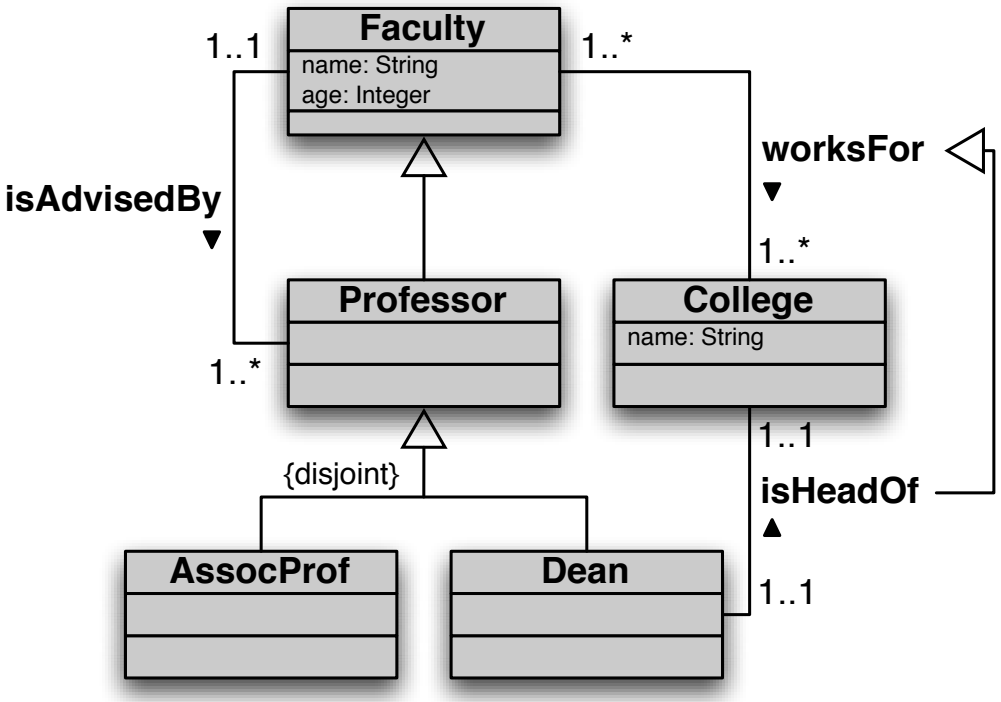
Capturing basic ontology constructs in DL-Lite$_{A,id}$

ISA between classes	$A_1 \sqsubseteq A_2$
Disjointness between classes	$A_1 \sqsubseteq \neg A_2$
Domain and range of properties	$\exists P \sqsubseteq A_1 \quad \exists P^- \sqsubseteq A_2$
Mandatory participation *(min card = 1)*	$A_1 \sqsubseteq \exists P \quad A_2 \sqsubseteq \exists P^-$
Functionality of relations *(max card = 1)*	(**funct** P) (**funct** P^-)
ISA between properties	$Q_1 \sqsubseteq Q_2$
Disjointness between properties	$Q_1 \sqsubseteq \neg Q_2$

Note 1: *DL-Lite*$_{A,id}$ cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., **OR**).

Note 2: *DL-Lite*$_{A,id}$ can be extended to capture also **min cardinality constraints** ($A \sqsubseteq \leq\ n\ Q$), **max cardinality constraints** ($A \sqsubseteq \geq\ n\ Q$) [Artale et al, JAIR 2009], n-**ary** relations, and **denial assertions** (not considered here for simplicity).

Example of DL-Lite$_{A,id}$ ontology



$$
\begin{aligned}
\text{Professor} &\sqsubseteq \text{Faculty}\\
\text{AssocProf} &\sqsubseteq \text{Professor}\\
\text{Dean} &\sqsubseteq \text{Professor}\\
\text{AssocProf} &\sqsubseteq \neg\text{Dean}
\end{aligned}
$$

$$
\begin{aligned}
\text{Faculty} &\sqsubseteq \exists \mathit{age}\\
\exists \mathit{age}^- &\sqsubseteq \texttt{xsd:integer}\\
&(\textbf{funct}\ \mathit{age})
\end{aligned}
$$

$$
\begin{aligned}
\exists\text{worksFor} &\sqsubseteq \text{Faculty}\\
\exists\text{worksFor}^- &\sqsubseteq \text{College}\\
\text{Faculty} &\sqsubseteq \exists\text{worksFor}\\
\text{College} &\sqsubseteq \exists\text{worksFor}^-
\end{aligned}
$$

$$
\begin{aligned}
\exists\text{isHeadOf} &\sqsubseteq \text{Dean}\\
\exists\text{isHeadOf}^- &\sqsubseteq \text{College}\\
\text{Dean} &\sqsubseteq \exists\text{isHeadOf}\\
\text{College} &\sqsubseteq \exists\text{isHeadOf}^-\\
\text{isHeadOf} &\sqsubseteq \text{worksFor}\\
&(\textbf{funct}\ \text{isHeadOf})\\
&(\textbf{funct}\ \text{isHeadOf}^-)
\end{aligned}
$$

$$\vdots$$

Query answering in DL-Lite$_{A,id}$

Possible approaches:

- the chase (used in database theory for reasoning about data dependencies [Maier 1983])
- resolution-based methods
- ...

None of the existing approaches directlty works for our purpose.

$\rightsquigarrow$ So, we designed our own algorithm, called *PerfectRef*, implemented in our OBDA tool, MASTRO

Query answering in DL-Lite$_{A,id}$

Remark

We call **positive inclusions (PIs)** assertions of the form

$$B_1 \sqsubseteq B_2, \quad Q_1 \sqsubseteq Q_2$$

whereas we call negative inclusions (NIs) assertions of the form

$$B_1 \sqsubseteq \neg B_2, \quad Q_1 \sqsubseteq \neg Q_2$$

Theorem

Let q be a boolean UCQs and $\mathcal{O} = \mathcal{O}_{\text{PI}} \cup \mathcal{O}_{\text{NI}} \cup \mathcal{O}_{\text{id}}$ be a TBox s.t.

- $\mathcal{O}_{\text{PI}}$ is a set of PIs
- $\mathcal{O}_{\text{NI}}$ is a set of NIs
- $\mathcal{O}_{\text{id}}$ is a set of identification assertions.

For each $\mathcal{S}$ such that $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ **is satisfiable**, we have that

$$\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle \models q \text{ iff } \langle \mathcal{O}_{\text{PI}}, \mathcal{S}, \mathcal{M} \rangle \models q.$$

In other words, we have that $cert(q, \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle) = cert(q, \langle \mathcal{O}_{\text{PI}}, \mathcal{S}, \mathcal{M} \rangle)$.

Query answering in DL-Lite$_{A,id}$: Query rewriting (cont'd)

Intuition: Use the PIs as basic rewriting rules

$$\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$$

$$\mathsf{AssocProfessor} \sqsubseteq \mathsf{Professor}$$

$$\text{as a logic rule:} \quad \mathsf{Professor}(z) \leftarrow \mathsf{AssocProfessor}(z)$$

Basic rewriting step:

when the atom unifies with the **head** of the rule (with mgu σ).

substitute the atom with the **body** of the rule (to which σ is applied).

Towards the computation of the perfect rewriting, we add to the input query above the following query ($\sigma = \{z/x\}$)

$$\mathsf{q}(x) \leftarrow \mathsf{AssocProfessor}(x)$$

We say that the PI $\mathsf{AssocProfessor} \sqsubseteq \mathsf{Professor}$ **applies** to the atom $\mathsf{Professor}(x)$.

Query answering in DL-Lite$_{A,id}$: Query rewriting (cont'd)

Consider now the query

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y)$$

$$\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$$

as a logic rule: $\mathsf{teaches}(z_1, z_2) \leftarrow \mathsf{Professor}(z_1)$

We add to the reformulation the query ($\sigma = \{z_1/x, z_2/y\}$)

$$\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$$

Query answering in DL-Lite$_{A,id}$: Query rewriting (cont'd)

Conversely, for the query

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, \mathtt{databases})$$

$$\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$$

$$\text{as a logic rule:} \quad \mathsf{teaches}(z_1, z_2) \leftarrow \mathsf{Professor}(z_1)$$

$\mathsf{teaches}(x, \mathtt{databases})$ does not unify with $\mathsf{teaches}(z_1, z_2)$, since the **existentially quantified variable** z_2 in the head of the rule **does not unify** with the constant $\mathtt{databases}$.

In this case the PI **does not apply** to the atom $\mathsf{teaches}(x, \mathtt{databases})$.

The same holds for the following query, where y is **distinguished**

$$\mathsf{q}(x, y) \leftarrow \mathsf{teaches}(x, y)$$

Query answering in DL-Lite$_{A,id}$: Query rewriting (cont'd)

An analogous behavior with join variables

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$$

$$\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$$

$$\text{as a logic rule:} \quad \mathsf{teaches}(z_1, z_2) \leftarrow \mathsf{Professor}(z_1)$$

The PI above does not apply to the atom $\mathsf{teaches}(x, y)$.

Conversely, the PI

$$\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}$$

$$\text{as a logic rule:} \quad \mathsf{Course}(z_2) \leftarrow \mathsf{teaches}(z_1, z_2)$$

applies to the atom $\mathsf{Course}(y)$.

We add to the perfect rewriting the query ($\sigma = \{z_2/y\}$)

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{teaches}(z_1, y)$$

Query answering in DL-Lite$_{A,id}$: Query rewriting (cont'd)

We now have the query

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{teaches}(z, y)$$

The PI $\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$

as a logic rule: $\mathsf{teaches}(z_1, z_2) \leftarrow \mathsf{Professor}(z_1)$

does not apply to $\mathsf{teaches}(x, y)$ nor $\mathsf{teaches}(z, y)$, since y is a join variable.

However, we can transform the above query by unifying the atoms $\mathsf{teaches}(x, y)$, $\mathsf{teaches}(z, y)$. This rewriting step is called **reduce**, and produces the following query

$$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y)$$

We can now apply the PI above ($\sigma\{z_1/x, z_2/y\}$), and add to the reformulation the query

$$\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$$

Query answering in DL-Lite$_{A,id}$: Query rewriting (cont'd)

Algorithm *PerfectRef*$(q, \mathcal{O}_P)$
Input: conjunctive query q, set of *DL-Lite*$_{A,id}$ PIs $\mathcal{O}_P$
Output: union of conjunctive queries PR
$PR := \{q\}$;
repeat
  $PR' := PR$;
  for each $q \in PR'$ **do**
  (a) **for each** g in q **do**
      for each PI I in $\mathcal{O}_P$ **do**
        if I is applicable to g
        then $PR := PR \cup \{\, q[g/(g, I)] \,\}$
  (b) **for each** g_1, g_2 in q **do**
      if g_1 and g_2 unify
      then $PR := PR \cup \{\tau(\mathit{reduce}(q, g_1, g_2))\}$;
until $PR' = PR$;
return PR

Answering by rewriting in DL-Lite$_{A,id}$: The algorithm

1. Rewrite the CQ q into a UCQs: apply to q in all possible ways the PIs in the TBox $\mathcal{O}$.
2. This corresponds to exploiting ISAs, role typings, and mandatory participations to obtain new queries that could contribute to the answer.
3. Unifying atoms can make applicable rules that could not be applied otherwise.

Theorem (Calvanese et al, JAR 2007)

The query resulting from the above process is a UCQ, and is the **perfect rewriting** $r_{q,\mathcal{O}}$.

Note that the same algorithm can be used to check satisfiability of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$

Query answering in DL-Lite$_{A,id}$: Example

TBox: $\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$
$\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}$

Query: $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$

Perfect Rewriting: $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$
$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{teaches}(z, y)$
$\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, z)$
$\mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$

$\mathcal{M}(\mathcal{S})$: $\mathsf{teaches}(\mathtt{John}, \mathtt{databases})$
$\mathsf{Professor}(\mathtt{Mary})$

It is easy to see that the evaluation of $r_{q,\mathcal{O}}$ over $\mathcal{M}(\mathcal{S})$ in this case produces the set $\{\mathtt{John}, \mathtt{Mary}\}$.

Complexity

n : query size

m : number of predicate symbols in $\mathcal{O}$ or query q

The number of distinct conjunctive queries generated by the algorithm is less than or equal to $(m \times (n+1)^2)^n$, which corresponds to the maximum number of executions of the repeat-until cycle of the algorithm.

Query answering for CQs and UCQs is:

- PTIME in the size of TBox.
- AC^0 in the size of the $\mathcal{M}(\mathcal{S})$.
- Exponential in the size of the query.

Can we go beyond *DL-Lite*$_{A,id}$ and remain in AC^0?

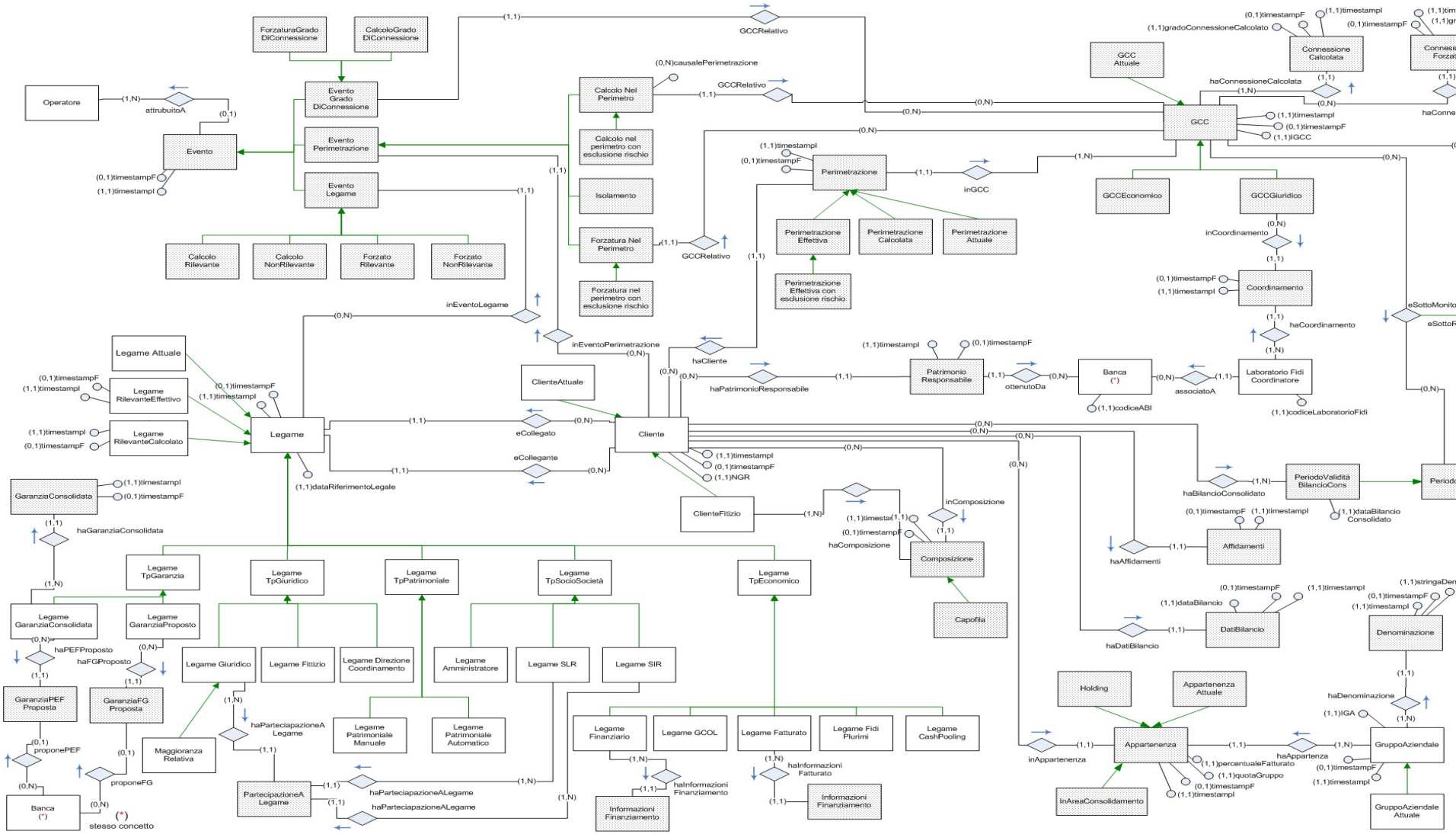
By adding essentially any other DL construct (without limitations) we lose these computational properties.

Beyond DL-Lite$_{A,id}$: results on data complexity

	lhs	rhs	funct.	Prop. incl.	Data complexity of query answering
0	*DL-Lite*$_{\mathcal{A},id}$		–	√	in AC^0
1	$A \mid \exists P.A$	A	–	–	NLogSpace-hard
2	A	$A \mid \forall P.A$	–	–	NLogSpace-hard
3	A	$A \mid \exists P.A$	√	–	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	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	√	–	PTime-hard
7	$A \mid \exists P.A \mid \exists P^-.A$	$A \mid \exists P$	–	–	PTime-hard
8	$A \mid \exists P \mid \exists P^-$	$A \mid \exists P \mid \exists P^-$	√	√	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**

- *DL-Lite*$_{\mathcal{A},id}$ is the most expressive DL of the *DL-Lite* family
- NLogSpace and PTime hardness holds already for instance checking.
- For coNP-hardness in line 10, a TBox with a single assertion $A_L \sqsubseteq A_T \sqcup A_F$ suffices! $\rightsquigarrow$ **No** hope of including **covering constraints**.

Complexity counts



Sources of complexity

For realistic ontologies, systems based on *PerfectRef* works for queries with at most 7-8 atoms.

Two sources of complexity wrt query:

- conjunctive query evaluation is NP-complete – complexity comes from the need of matching the query and the data
  $\rightsquigarrow$ **unavoidable!**
- the rewritten query has exponential size wrt the original query – complexity comes from the need of "expanding" the query w.r.t. the ontology
  $\rightsquigarrow$ **avoidable?**

Avoiding exponential blow-up: first attempt

Idea: avoid rewriting whatsoever!

Unfortunately, this idea does not work:

Theorem (Calvanese et al, JAR 2007)

Given $\mathcal{M}(\mathcal{S})$, there is no finite database B such that for every query q, $cert(q, \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle) = q^B$

Avoiding exponential blow-up: second attempt

Idea: try to apply the chase only partially, so as to obtain a finite database (possibly with variables) such that only "small rewritings" of conjunctive queries are needed [Kontchakov et al, KR 2010]

Two problems:

- it works only without role inclusions
- (partial) materialization not always appropriate in OBDA

Avoiding exponential blow-up: third attempt

REQUIEM [Pérez-Urbina et al, 2009] is a query rewriting algorithm based on resolution for $\mathcal{ELHIO}$ that rewrites in UCQ for DL-lite, and reduces the number of "reduce" steps wrt *PerfectRef*. Still the size of the rewritten query is exponential in the worst case.

Presto [Rosati, KR 2010] is the current rewriting algorithm used in MASTRO (OBDA tool developed at University of Roma La Sapienza), based on the following ideas for improving the performance of *PerfectRef*:

- centering the rewriting around the query variables rather than the query atoms – this allows for
  - collapsing sequences of rewriting steps into single steps and
  - dramatically pruning the solution space of the algorithm
- going beyond the disjunctive normal form (UCQ) of the rewritten query – nonrecursive datalog queries (UCQ with an additional unfolding step)

Avoiding exponential blow-up: third attempt

Presto replaces the "atom-rewrite" and "reduce" rules of *PerfectRef* with a rule (based on MGS) that eliminates existential join variables, where elimination of an existential join variable means that the variable turns into a non-join existential variable (through unification steps)

This makes the rewriting produced by Presto exponential with respect to the number of eliminable existential join variables in the query (notice: in practice, often the majority of existential join variables in a CQ are not eliminable)

$\rightsquigarrow$ dramatic reduction of the size of the query generated. Presto can handle queries with about 30 atoms.

Avoiding exponential blow-up: fourth attempt

- [Calvanese et al, KR 2012] shows how to exploit knowledge about inclusion dependencies on $\mathcal{M}(\mathcal{S})$ in order to produce more compact rewritings (See QUEST, an OBDA tool developed at the Free University of Bolzano)
- Prexto [Rosati, ESWC 2012] applies this idea to Presto.

In the worst case, Presto still rewrites into a union of conjunctive queries of exponential size wrt the original query. Is it avoidable?

Theorem (Kikot et al, DL 2011)

Cheking whether $\vec{t} \in cert(q, \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle)$ is NP-complete in combined complexity, even if $\mathcal{O}$ is in *DL-Lite*$_\mathcal{R}$, and $\mathcal{M}(\mathcal{S})$ is constituted by just one atomic assertion $A(c)$.

Since answering a FOL query over a database $A(c)$ can be done in linear time, it follows that no algorithm can construct a FOL rewriting in polynomial time, unless P = NP.

Avoiding exponential blow-up: other attempts

- Polynomial FOL rewritings exist in the case of *DL-Lite*$_{core}$ (*DL-Lite*$_{\mathcal{R}}$ without role inclusions) [Kikot et al, DL 2011].
- Polynomial rewritings exist if we allow rewriting to be expressed in nonrecursive Datalog queries using additional symbols [Gottlob and Schwentick, DL 2011].
- Many recent papers carry out deep investigations on the problem from various points of view [Kikot et al, AAAI 2011], [Gottlob et al, ICDE 2011], [Kikot et al, KR 2012], [Kikot et al, DL 2012], etc.

Outline

Example: an inconsistent DL-Lite ontology

$\mathcal{O}$

$\mathsf{RedWine} \sqsubseteq \mathsf{Wine}$ $\quad$ $\mathsf{WhiteWine} \sqsubseteq \mathsf{Wine}$

$\mathsf{RedWine} \sqsubseteq \neg\, \mathsf{WhiteWIne}$ $\quad$ $\mathsf{Wine} \sqsubseteq \neg\, \mathsf{Beer}$

$\mathsf{Wine} \sqsubseteq \exists \mathsf{producedBy}$ $\quad$ $\exists \mathsf{producedBy} \sqsubseteq \mathsf{Wine}$

$\mathsf{Wine} \sqsubseteq \neg\, \mathsf{Winery}$ $\quad$ $\mathsf{Beer} \sqsubseteq \neg\, \mathsf{Winery}$

$\exists \mathsf{producedBy}^{-} \sqsubseteq \mathsf{Winery}$ $\quad$ $(\mathit{funct}\ \mathsf{producedBy})$

$\mathcal{M}$

$\mathsf{R1(x,y,`white')} \rightsquigarrow \mathsf{WhiteWine(x)}$ $\quad$ $\mathsf{R1(x,y,`red')} \rightsquigarrow \mathsf{RedWine(x)}$

$\mathsf{R2(x,y)} \rightsquigarrow \mathsf{Beer(x)}$ $\quad$ $\mathsf{R1(x,y,z)} \vee \mathsf{R2(x,y)} \rightsquigarrow \mathsf{producedBy(x,y)}$

$\mathcal{S}$

$\mathsf{R1(grechetto,p1,`white')}$ $\quad$ $\mathsf{R1(grechetto,p1,`red')}$

$\mathsf{R2(guinnes,p2)}$ $\quad$ $\mathsf{R1(falanghina,p1,`white')}$

The problem

One popular approach to dealing with inconsistency in data management is data cleaning

However, even with data cleaning, inconsistencies may remain, and we would like our system to provide meaningful answers to queries.

The problem is that query answering based on classical logic becomes meaningless in the presence of inconsistency (ex falso quodlibet)

Question

How to handle classically-inconsistent OBDA systems in a more meaningful way?

Inconsistent-tolerant semantics

The semantics we propose [Lembo et al, RR 2010] for inconsistent OBDA systems is based on the following principles:

- We assume that $\mathcal{O}$ and $\mathcal{M}$ are always consistent (this is true if $\mathcal{O}$ is expressed in *DL-Lite*$_{\mathcal{A},id}$)
- Inconsistencies are caused by the interaction between the data at $\mathcal{S}$ and the other components of the system, i.e., between $\mathcal{M}(\mathcal{S})$ and $\mathcal{O}$
- We resort to the notion of *repair* [Arenas, Bertossi, Chomicki, PODS 1999]. Intuitively, a repair for $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ is an ontology $\langle \mathcal{O}, \mathcal{A} \rangle$ that is consistent, and "minimally" differs from $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$.

  See [Leopoldo Bertossi, "Database Repairing and Consistent Query Answering", *Synthesis Lectures on Data Management*, Vol. 3, No. 5, Morgan and Claypool].

Inconsistent-tolerant semantics

What does it mean for $\mathcal{A}$ to be "minimally different" from $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$? We base this concept on the notion of symmetric difference.

We write $S_1 \oplus S_2$ to denote the symmetric difference between S_1 and S_2, i.e.,

$$S_1 \oplus S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$$

Definition (Repair)

Let $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA system. A **repair** of $\mathcal{K}$ is an ABox $\mathcal{A}$ such that:

1. $Mod(\langle \mathcal{O}, \mathcal{A} \rangle) \neq \emptyset$,
2. no set of facts $\mathcal{A}'$ exists such that
   - $Mod(\langle \mathcal{O}, \mathcal{A}' \rangle) \neq \emptyset$,
   - $\mathcal{A}' \oplus \mathcal{M}(\mathcal{S}) \subset \mathcal{A} \oplus \mathcal{M}(\mathcal{S})$

Example: Repairs

Rep_1

{WhiteWine(grechetto), Beer(guinnes), WhiteWine(falanghina)}

Rep_2

{RedWine(grechetto), Beer(guinnes), WhiteWine(falanghina)}

Rep_3

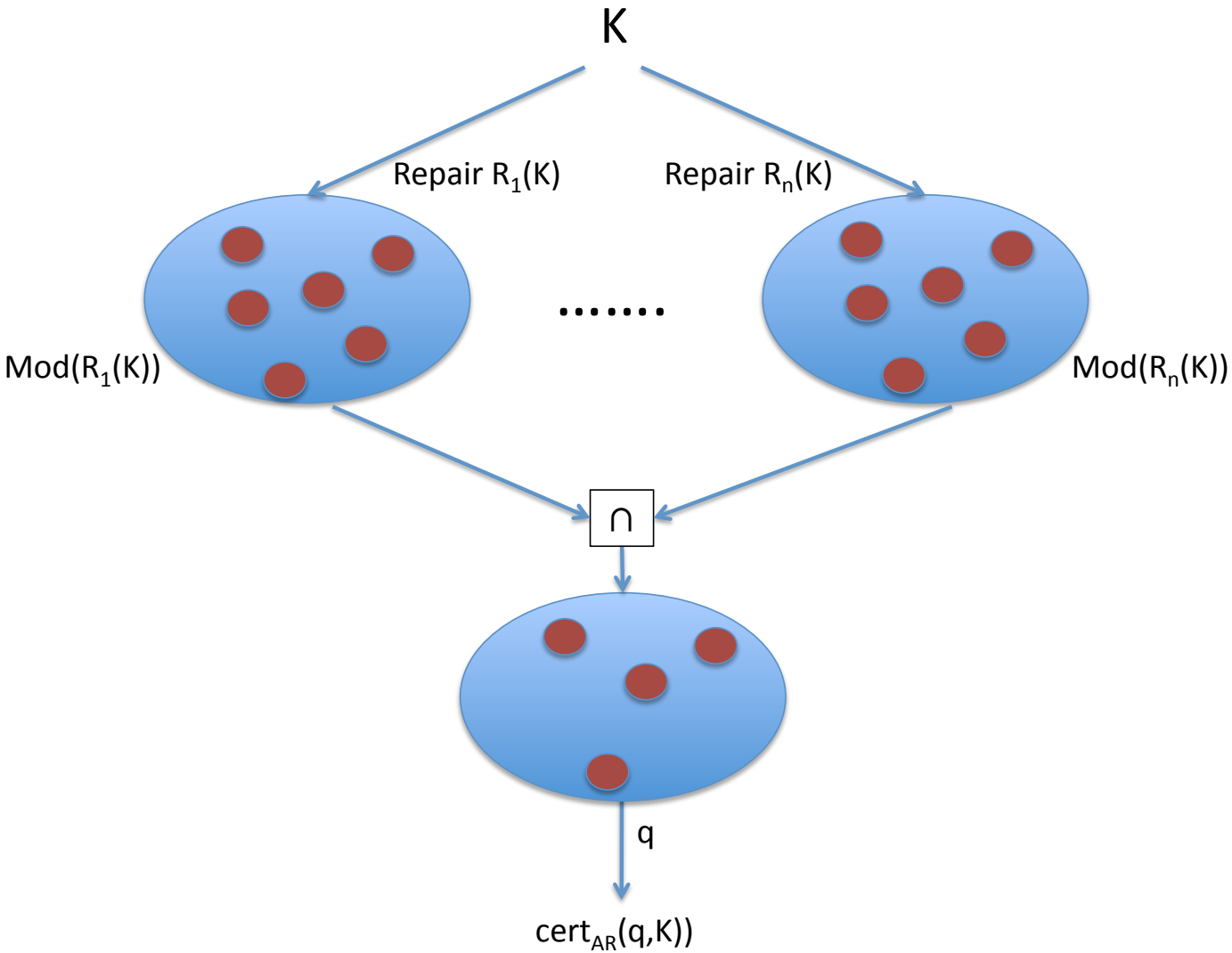
{WhiteWine(grechetto), producedBy(guinnes, p2),
WhiteWine(falanghina)}

Rep_4

{RedWine(grechetto), producedBy(guinnes, p2),
WhiteWine(falanghina)}

Reasoning wih all repairs: the AR semantics

An interpretation $\mathcal{I}$ is a model of $\mathcal{K}$ if $\mathcal{I}$ is a model of **every** repair of $\mathcal{K}$.



Reasoning wih all repairs: the AR semantics

Problems:

- Many repairs in general
- What is the complexity of reasoning about all such repairs?

Theorem

Let $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA system, and let α be a ground atom. Deciding whether α is logically implied by every repair of $\mathcal{K}$ is coNP-complete with respect to data complexity.

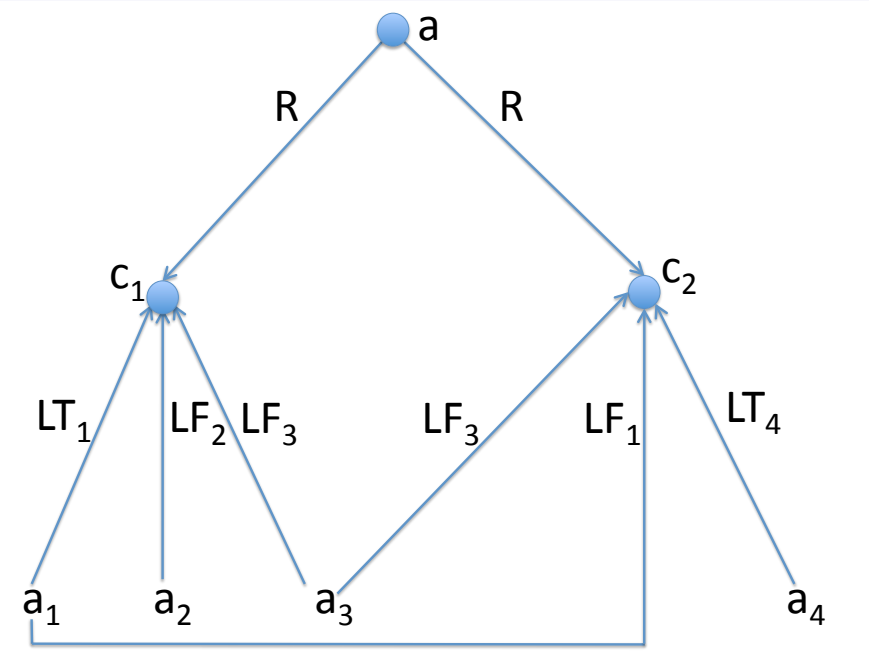
coNP hardness of the AR-semantics

Ontology $\mathcal{O}$

$$\exists R \sqsubseteq Unsat$$
$$\exists R^- \sqsubseteq \neg\exists LT_1^-$$
$$\exists R^- \sqsubseteq \neg\exists LF_1^-$$
$$\exists R^- \sqsubseteq \neg\exists LT_2^-$$
$$\exists R^- \sqsubseteq \neg\exists LF_2^-$$
$$\exists R^- \sqsubseteq \neg\exists LT_3^-$$
$$\exists R^- \sqsubseteq \neg\exists LF_3^-$$
$$\exists LT_1 \sqsubseteq \neg\exists LF_1$$
$$\exists LT_1 \sqsubseteq \neg\exists LF_2$$
$$\exists LT_1 \sqsubseteq \neg\exists LF_3$$
$$\exists LF_1 \sqsubseteq \neg\exists LT_2$$
$$\exists LF_1 \sqsubseteq \neg\exists LT_3$$
$$\exists LT_2 \sqsubseteq \neg\exists LF_2$$
$$\exists LT_2 \sqsubseteq \neg\exists LF_3$$
$$\exists LF_2 \sqsubseteq \neg\exists LT_3$$
$$\exists LT_3 \sqsubseteq \neg\exists LF_3$$

3-CNF formula ϕ: $(a_1 \vee \neg a_2 \vee \neg a_3) \wedge (\neg a_3 \vee a_4 \vee \neg a_1)$

ABox $\mathcal{A}$ corresponding to ϕ



ϕ satisfiable iff $\langle \mathcal{O}, \mathcal{A} \rangle \not\models_{AR} Unsat(a)$

When in doubt, throw it out: the IAR semantics

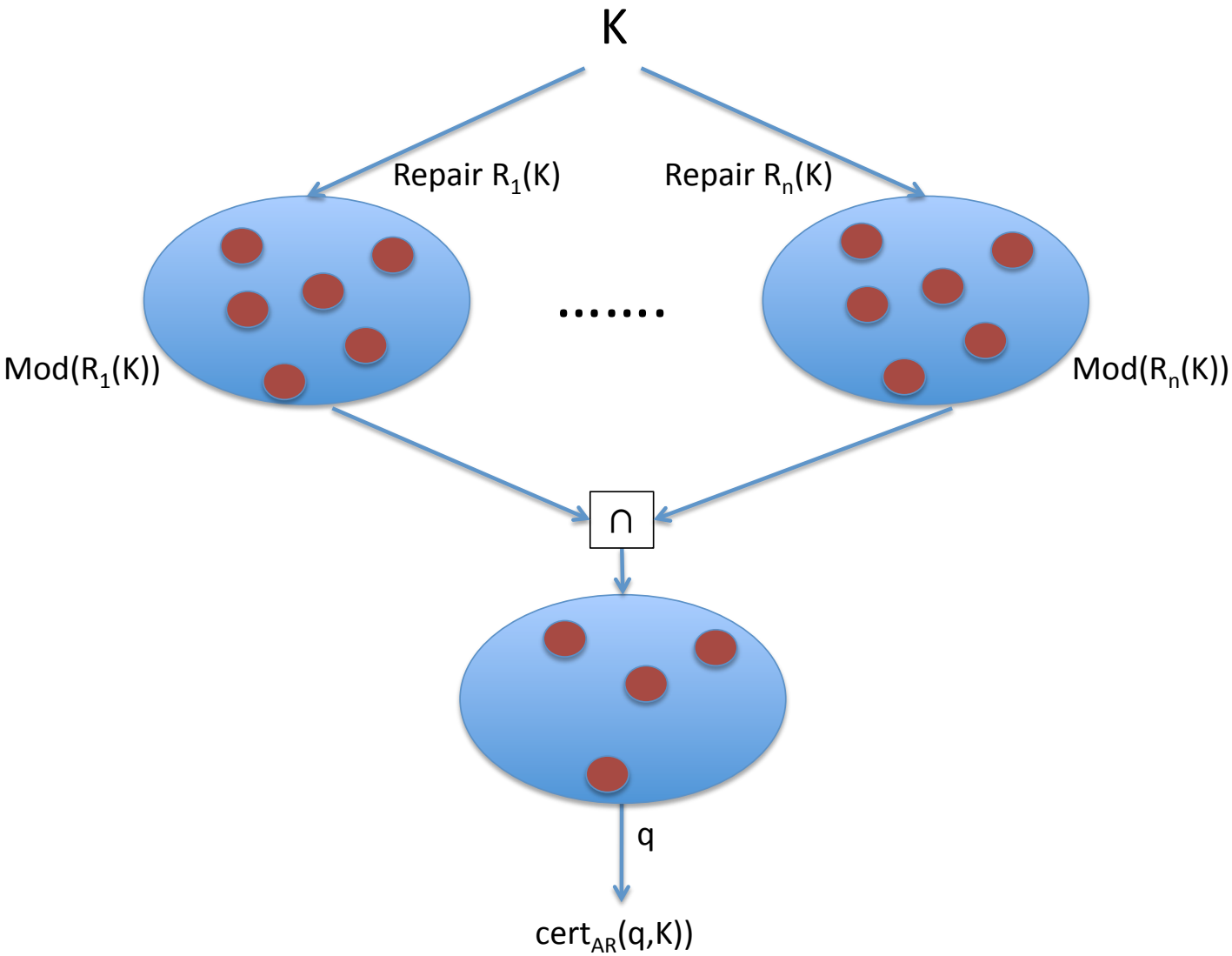
Other intractability results of the AR semantics, even for simpler languages (e.g., [Bienvenu, DL 2012])

Idea: The IAR semantics

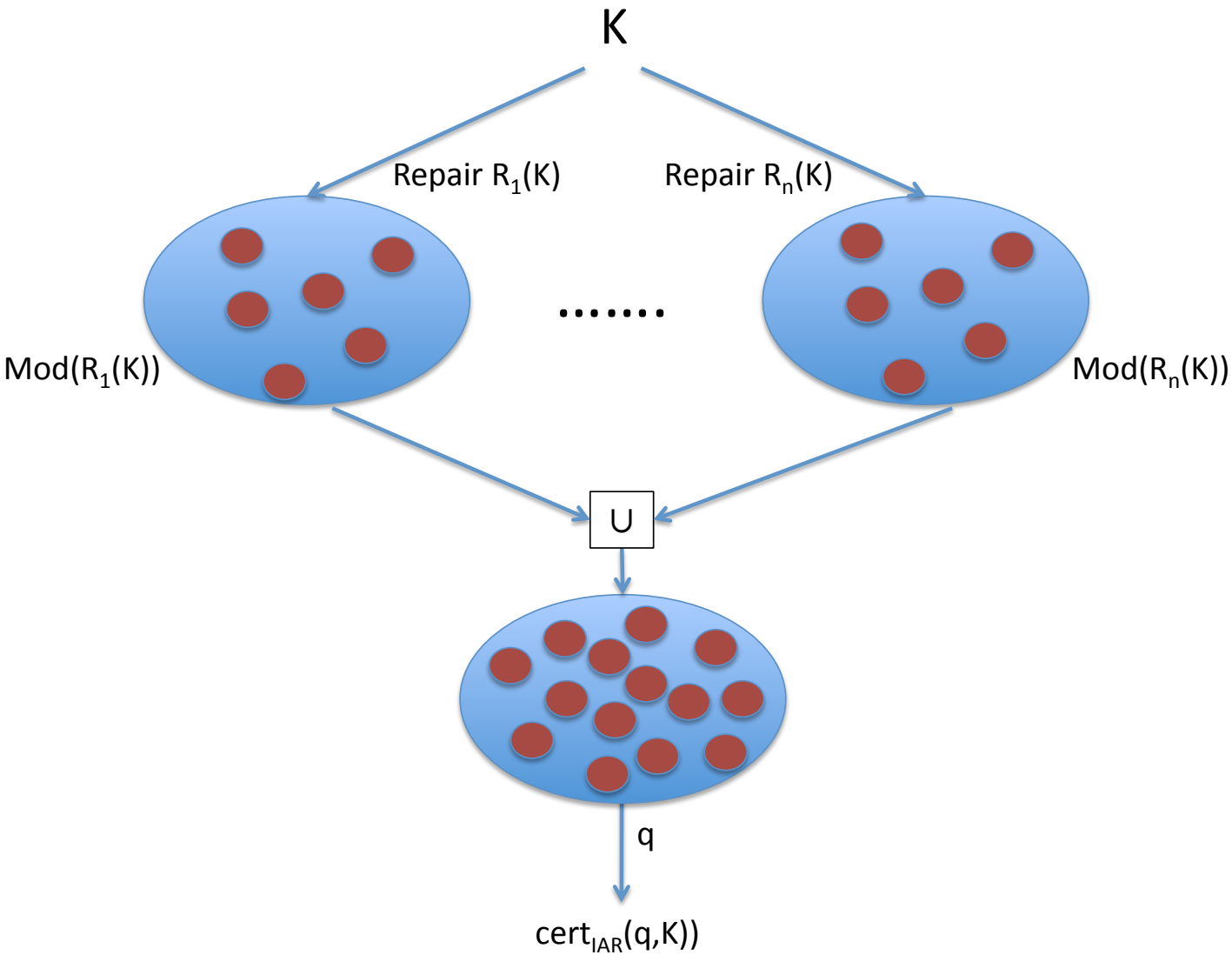
An interpretation $\mathcal{I}$ is a model of $\mathcal{K}$ according to the IAR semantics if $\mathcal{I}$ is a model of **at least one** repair of $\mathcal{K}$.
In other words, consider the "intersection of all repairs", and consider the set of models of such intersection as the semantics of the system (When in Doubt, Throw It Out).

Note that the IAR semantics is an approximation of the AR semantics

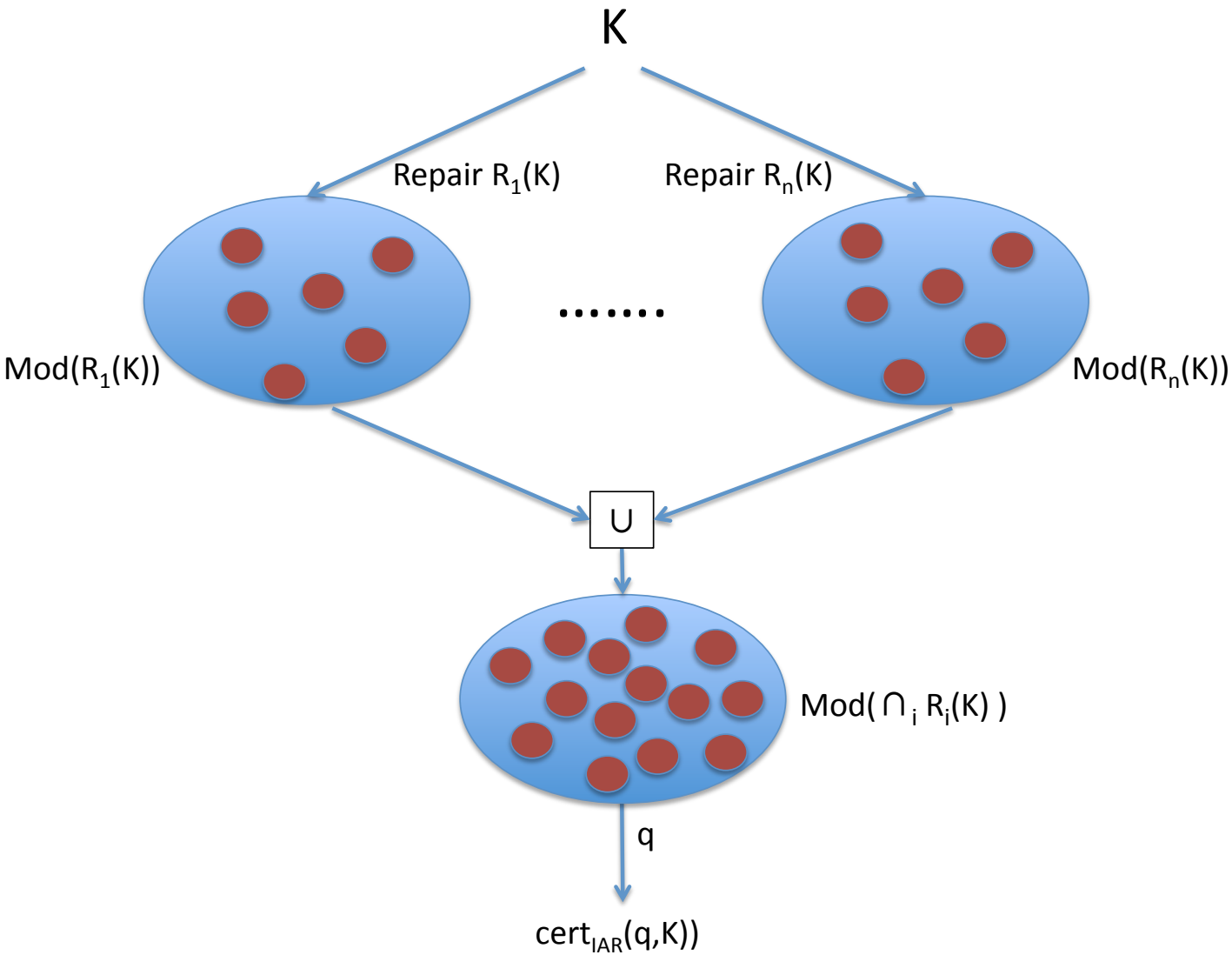
Reasoning wih all repairs: the AR semantics



When in doubt, throw it out: the IAR semantics



When in doubt, throw it out: the IAR semantics



Inconsistent-tolerant query answering

Two possible methods for answering queries posed to $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ according to the inconsistency-tolerant semantics:

- Compute the intersection $\mathcal{A}$ of all repairs of $\mathcal{K}$, and then compute $\vec{t}$ such that $\langle \mathcal{O}, \mathcal{A} \rangle \models q(\vec{t})$
- Rewrite the query q into q' in such a way that, for all $\vec{t}$, we have that $\mathcal{K} \models_{IAR} q(\vec{t})$ is equivalent to $\vec{t} \in q'(\mathcal{M}(\mathcal{S}))$. Then, evaluate q' over $\mathcal{M}(\mathcal{S})$.

We have devised a rewriting technique which encodes a UCQ q into a FOL query q' which, evaluated against the original $\mathcal{M}(\mathcal{S})$ retrieves only the certain answers of q w.r.t the IAR semantics [Lembo et al, DL 2012].

Rewriting technique

We provide a rewriting technique which encodes a UCQ Q into a FOL query Q' which evaluated against the original $\mathcal{S}$ retrieves only the certain answers of Q w.r.t the IR semantics

Rewriting technique

Given a UCQ $Q = q_1 \vee q_2 \vee \ldots \vee q_n$ over $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$

- we compute $\text{PerfectRef}_{IAR}(Q, \mathcal{O}, \mathcal{M})$ as
  $$\text{MapRewriting}_{\mathcal{M}}(\text{IncRewritingUCQ}_{IAR}(\text{PerfectRef}(Q, \mathcal{O}), \mathcal{O}))$$
- we evaluate $\text{PerfectRef}_{IAR}(Q, \mathcal{O}, \mathcal{M})$ over $\mathcal{S}$

where

- $\text{PerfectRef}(Q, \mathcal{O})$ rewrites Q taking care of $\mathcal{O}$
- $\text{IncRewritingUCQ}_{IAR}(Q, \mathcal{O}) = \bigvee_{i=1}^{n} \text{IncRewriting}(q_i, \mathcal{O})$ rewrites Q taking care of inconsistencies
- $\text{MapRewriting}_{\mathcal{M}}(Q)$ rewrites Q taking care of $\mathcal{M}$

Inconsistency on disjointness assertions

Given a disjointness assertion $B \sqsubseteq \neg C$ implied by $\mathcal{O}$, an inconsistency may arise if both $B(a)$ and $C(a)$ are in $\mathcal{M}(\mathcal{S})$, for some constant a.
Analogously, given $B \sqsubseteq \neg\exists P$ (resp. $B \sqsubseteq \neg\exists P^-$), an inconsistency may arise if $B(a)$ and $P(a, b)$ (resp. $P(b, a)$) belong to $\mathcal{M}(\mathcal{S})$.
In order to characterize such inconsistencies, given a concept B we define $\text{NotDisjClash}_B^{\mathcal{O}}(t)$ as the following FOL formula:

$$\bigwedge_{C \in DC(B,\mathcal{O})} \neg C(t) \quad \bigwedge_{P \in DRD(B,\mathcal{O})} \neg\exists y.P(t, y) \quad \bigwedge_{P \in DRR(B,\mathcal{O})} \neg\exists y.P(y, t)$$

where
$DC(B, \mathcal{O}) = \{C \mid C \text{ is an atomic concept s.t. } \mathcal{O} \models B \sqsubseteq \neg C\}$
$DRD(B, \mathcal{O}) = \{P \mid P \text{ is an atomic role s.t. } \mathcal{O} \models B \sqsubseteq \neg\exists P\}$
$DRR(B, \mathcal{O}) = \{P \mid P \text{ is an atomic role s.t. } \mathcal{O} \models B \sqsubseteq \neg\exists P^-\}$

Inconsistency on disjointness assertions: example

$\mathcal{O}$

RedWine $\sqsubseteq$ Wine
RedWine $\sqsubseteq \neg$ WhiteWine
Wine $\sqsubseteq \exists$producedBy
$\exists$producedBy$^- \sqsubseteq$ Winery
Wine $\sqsubseteq \neg$ Winery
(*funct* producedBy)

WhiteWine $\sqsubseteq$ Wine
Beer $\sqsubseteq \neg$ Wine
$\exists$producedBy $\sqsubseteq$Wine
Beer $\sqsubseteq \neg \exists$producedBy
Beer $\sqsubseteq \neg$ Winery

Due to RedWine $\sqsubseteq \neg$RedWine we have that:

$$\mathsf{NotDisjClash}^{\mathcal{O}}_{\mathsf{RedWine}}(t) = \neg\mathsf{WhiteWine}(t) \wedge \neg\mathsf{Winery}(t) \wedge \neg\mathsf{Beer}(t)$$

and due to Beer $\sqsubseteq \neg\exists$ProducedBy we have that:

$$\mathsf{NotDisjClash}^{\mathcal{O}}_{Beer}(t) = \neg\exists y.\mathsf{producedBy}(t, y) \wedge \neg\mathsf{Wine}(t) \wedge$$
$$\neg\mathsf{RedWine}(t) \wedge \neg\mathsf{WhiteWine}(t) \wedge \neg\mathsf{Winery}(T)$$

Inconsistency on functionalities

Given a functionality assertion $(funct\ P)$ over a role P, an inconsistency may arise if the assertions $P(a, b)$ and $P(a, c)$ belong to the ABox A.

Analogously, given a functionality assertion $(funct\ P^-)$ over a role P^-, an inconsistency may arise if $P(a, b)$ and $P(a, c)$ belong to the ABox A.

In order to detect such inconsistencies, given a role P we define $\mathsf{NotFunctClash}_P^{\mathcal{O}}(t, t')$ as the FOL formula:

- $\neg(\exists y.P(t, y) \wedge y \neq t')$, if $(funct\ P)$ exists in the ontology, and
- $\neg(\exists y.P(y, t') \wedge y \neq t)$, if $(funct\ P^-)$ exists in the ontology
- $\neg(\exists y.P(t, y) \wedge y \neq t') \wedge \neg(\exists y.P(y, t') \wedge y \neq t)$, if both the functionalities are present.

Other inconsistency causes

Other less intuitive cases which the algorithm takes into account:

- inconsistency deriving form irreflexive roles (that is, roles for which the ontology implies $P \sqsubseteq \neg P^-$)
- inconsistency deriving from roles such that $T \models \exists P \sqsubseteq \neg \exists P^-$

$$P(a, a)$$

- inconsistencies deriving from attribute assertion in which the type of the range is not respected

$$T = \{\rho(U) \sqsubseteq xsd : string\} \qquad A = \{U(a, 12)\}$$

- "false" inconsistencies.

Skipping false inconsistencies

An atomic concept B is called empty (unsatisfiable) in $\mathcal{O}$ if $\mathcal{O} \models B \sqsubseteq \neg B$.

An ABox assertion $B(a)$ over an empty concept B, violating a disjointness assertion $B \sqsubseteq \neg C$ together with $C(a)$, is not a real inconsistency because it does not belong to any repair.

In order to skip false inconsistencies we define the condition

$$\text{ConsAtom}_B^{\mathcal{O}}(t) = \begin{cases} \textit{false} & \text{if } \mathcal{O} \models B \sqsubseteq \neg B \\ \textit{true} & \text{otherwise} \end{cases}$$

and put it in conjunction with NotDisjClash and NotFunctClash formulas.
A similar conditions holds for roles, which leads to the definition of the formula $\text{ConsAtom}_P^{\mathcal{O}}(t, t')$ for every empty role P.

Skipping false inconsistencies (2)

Considering the ConsAtom conditions, the check on disjointness becomes:

$$\mathsf{NotDisjClash}_B^{\mathcal{O}}(t) = \bigwedge_{C \in DC(B,T)} \neg C(t) \wedge \mathsf{ConsAtom}_C^{\mathcal{O}}(t)$$

$$\bigwedge_{P \in DRD(B,\mathcal{O})} \neg \exists y.P(t,y) \wedge \mathsf{ConsAtom}_P^{\mathcal{O}}(t,y)$$

$$\bigwedge_{P \in DRR(B,\mathcal{O})} \neg \exists y.P(y,t) \wedge \mathsf{ConsAtom}_P^{\mathcal{O}}(y,t)$$

and the check on functionalities (with both $(funct\ P)$ and $(funct\ P^-)$) becomes:

$$\mathsf{NotFunctClash}_P^{\mathcal{O}}(t,t') = \neg(\exists y.P(t,y) \wedge y \neq t' \wedge \mathsf{ConsAtom}_P^{\mathcal{O}}(t,y)) \wedge$$

$$\neg(\exists y.P(y,t') \wedge y \neq t \wedge \mathsf{ConsAtom}_P^{\mathcal{O}}(y,t))$$

Put it all together: $\mathbf{IncRewriting}_{IAR}(q, \mathcal{O})$

Let q be a CQ of the form

$$\exists x_1, \ldots, x_k. \bigwedge_{i=1}^{n} B_i(t_i^1) \wedge \bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3)$$

For every concept B_i and every role P_i appearing in q we define the following conditions:

$$\mathsf{NotClash}_B^{\mathcal{O}}(t) = \mathsf{NotDisjClash}_B^{\mathcal{O}}(t)$$

$$\mathsf{NotClash}_P^{\mathcal{O}}(t, t') = \mathsf{NotDisjClash}_P^{\mathcal{O}}(t) \wedge \mathsf{NotFunctClash}_P^{\mathcal{O}}(t, t')$$

and use them to build the rewriting:

$$\exists x_1, \ldots, x_k. \bigwedge_{i=1}^{n} B_i(t_i^1) \wedge \mathsf{ConsAtom}_{B_i}^{\mathcal{O}}(t_i^1) \wedge \mathsf{NotClash}_{B_i}^{\mathcal{O}}(t_i^1) \wedge$$
$$\textstyle\bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) \wedge \mathsf{ConsAtom}_{P_i}^{\mathcal{O}}(t_i^2, t_i^3) \wedge \mathsf{NotClash}_{P_i}^{\mathcal{O}}(t_i^2, t_i^3)$$

Example

Let us consider the CQ

$$q = \exists x.\mathsf{RedWine}(x)$$

We have that $\mathsf{IncRewriting}_{IAR}(q, \mathcal{O})$ is

$$\exists x.\mathsf{RedWine}(x) \wedge \neg\mathsf{WhiteWine}(x) \wedge \neg\mathsf{Beer}(x) \wedge \neg\mathsf{Winery}(x) \wedge$$

$$\neg(\exists y.\mathsf{producedBy}(x, y) \wedge x \neq y)$$

Notice that $\mathsf{ConsAtom}^{\mathcal{O}}_{\mathsf{RedWine}} = true$ because RedWine is not an empty concept.

Contribution

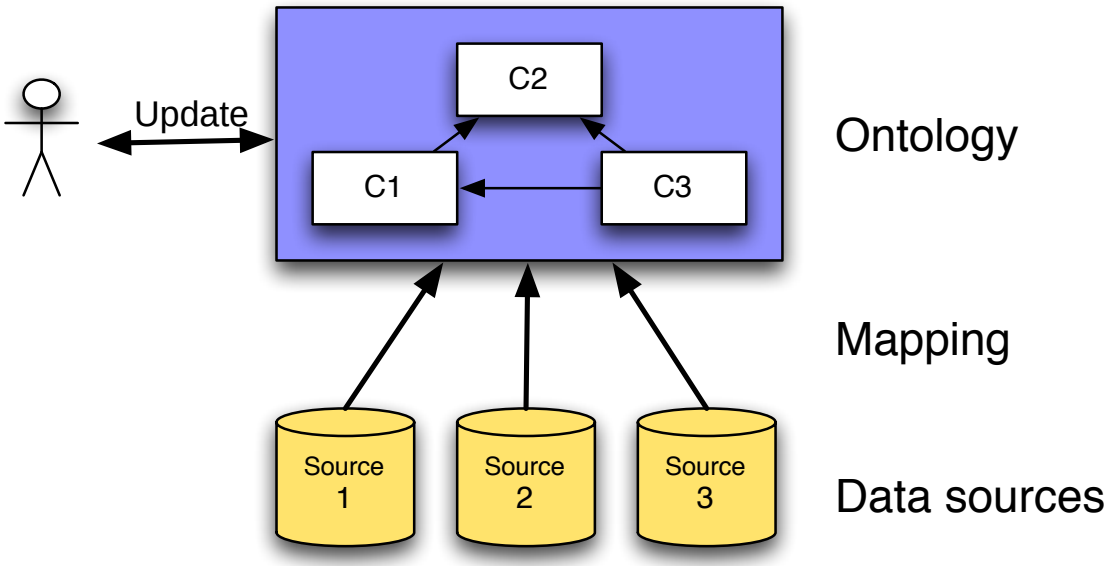
Theorem

Let Q be a UCQ over $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$. Deciding whether $\vec{t} \in cert_{IAR}(Q, \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle)$ is in AC^0 in data complexity.

problem	AR-semantics	IAR-semantics
instance checking	coNP-complete	in AC_0
UCQ answering	coNP-complete	in AC_0

Outline

Ontology-based update: challenges



- Which is a reasonable semantics for updates expressed over an ontology?
- How to "push" updates espressed over the ontology to updates over the sources?

An old problem

Knowledge/belief revision/update is a longstanding problem.

Main principle: keep the distance between the original KB $\mathcal{K}$ and the KB resulting from the application of an evolution operator minimal.

Two main approaches to define such a distance, called

1. model-based: the result is defined in terms of a set of models, with the idea that such a set should be as close as possible to the models of $\mathcal{K}$. Challenge: is the set of models expressible in $\mathcal{L}$? (see, for instance, [Liu et al, KR 2006, Kharlamov et al, DL 2011])
2. formula-based: the result is defined in terms of a formula, by resorting to some minimality criterion with respect to the formula expressing $\mathcal{K}$.
   Challenge: is the result unique? (see, for instance, [Fagin, Ullman, Vardi 1983], [Eiter, Gottlob 1992])

Our goal and assumptions

Our main goal was to come up with a method for instance-based evolution to add to MASTRO.
We have studied evolution operators under the following asssumptions:

- we have a fixed DL $\mathcal{L}$ (an ABox contains only ground atoms)
- we are interested in both insertion and deletion of chunks of knowledge
- evolution affects only the instance level of the KB
- the result should be expressed in $\mathcal{L}$
- the result shoud be independent of the syntactic form of the original KB

Our goal and assumptions

Our main goal was to come up with a method for instance-based evolution to add to MASTRO.
We have studied evolution operators under the following asssumptions:

- we have a fixed DL $\mathcal{L}$ (an ABox contains only ground atoms)
- we are interested in both insertion and deletion of chunks of knowledge
  $\Rightarrow$ **two operators**
- evolution affects only the instance level of the KB
- the result should be expressed in $\mathcal{L}$
- the result shoud be independent of the syntactic form of the original KB

Our goal and assumptions

Our main goal was to come up with a method for instance-based evolution to add to MASTRO.
We have studied evolution operators under the following asssumptions:

- we have a fixed DL $\mathcal{L}$ (an ABox contains only ground atoms)
- we are interested in both insertion and deletion of chunks of knowledge
  $\Rightarrow$ **two operators**
- evolution affects only the instance level of the KB
  $\Rightarrow$ **the TBox remains unchanged**
- the result should be expressed in $\mathcal{L}$

- the result shoud be independent of the syntactic form of the original KB

Our goal and assumptions

Our main goal was to come up with a method for instance-based evolution to add to MASTRO.
We have studied evolution operators under the following asssumptions:

- we have a fixed DL $\mathcal{L}$ (an ABox contains only ground atoms)
- we are interested in both insertion and deletion of chunks of knowledge
  $\Rightarrow$ **two operators**
- evolution affects only the instance level of the KB
  $\Rightarrow$ **the TBox remains unchanged**
- the result should be expressed in $\mathcal{L}$
  $\Rightarrow$ **formula-based approach**
- the result shoud be independent of the syntactic form of the original KB

Our goal and assumptions

Our main goal was to come up with a method for instance-based evolution to add to MASTRO.
We have studied evolution operators under the following asssumptions:

- we have a fixed DL $\mathcal{L}$ (an ABox contains only ground atoms)
- we are interested in both insertion and deletion of chunks of knowledge
  $\Rightarrow$ **two operators**
- evolution affects only the instance level of the KB
  $\Rightarrow$ **the TBox remains unchanged**
- the result should be expressed in $\mathcal{L}$
  $\Rightarrow$ **formula-based approach**
- the result shoud be independent of the syntactic form of the original KB
  $\Rightarrow$ **we base our semantics on $\mathbf{cl}_{\mathcal{O}}(\mathcal{A})$, where**
  $\mathbf{cl}_{\mathcal{O}}(\mathcal{A}) = \{\alpha \mid \alpha$ **is an ABox assertion and** $\langle \mathcal{O}, \mathcal{A} \rangle \models \alpha\}$

Accomplishing an update minimally

We base our notions on [Fagin, Ullman, Vardi 1983].

Definition

Let $\mathcal{A}'$ be an ABox.

- $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{O}, \mathcal{A} \rangle$ if $\mathcal{A}'$ is $\mathcal{O}$-consistent, and $\langle \mathcal{O}, \mathcal{A}' \rangle \models F$ (i.e., $F \subseteq \mathsf{cl}_{\mathcal{O}}(\mathcal{A}')$).
- $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{O}, \mathcal{A} \rangle$ if $\mathcal{A}'$ is $\mathcal{O}$-consistent, and $\langle \mathcal{O}, \mathcal{A}' \rangle \not\models F$ (i.e., $F \not\subseteq \mathsf{cl}_{\mathcal{O}}(\mathcal{A}')$).

We assume that $\mathcal{A}$ is $\mathcal{O}$-consistent.

Definition

Let $\mathcal{A}'$ be an ABox. Then $\mathcal{A}'$ accomplishes the insertion (deletion) of F into (from) $\langle \mathcal{O}, \mathcal{A} \rangle$ minimally if

- $\mathcal{A}'$ accomplishes the insertion (deletion) of F into (from) $\langle \mathcal{O}, \mathcal{A} \rangle$, and
- there is no $\mathcal{A}''$ that accomplishes the insertion (deletion) of F into (from) $\langle \mathcal{O}, \mathcal{A} \rangle$, and $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}')$ with respect to $\mathsf{cl}_{\mathcal{O}}(\mathcal{A})$.

The problem of multiple results

Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

The problem of multiple results

Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

The problem of multiple results

Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

$\mathcal{A}_1 = \{R(a_1, a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_1) = \mathcal{A}_1$

$\mathcal{A}_2 = \{C(a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_2) = \mathcal{A}_2$

The problem of multiple results

Example

$\mathcal{O}: \quad \exists R.C \sqsubseteq B, \quad B \sqsubseteq \neg D, \quad B \sqsubseteq E$

$\mathcal{A}: \quad \{R(a_1, a_2), C(a_2)\}, \quad$ with

$\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) = \{R(a_1, a_2), C(a_2), B(a_1), E(a_1)\}$

insert $F = \{D(a_1)\}$

$\mathcal{A}_1 = \{R(a_1, a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_1) = \mathcal{A}_1$

$\mathcal{A}_2 = \{C(a_2), D(a_1), E(a_1)\}, \quad$ with $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}_2) = \mathcal{A}_2$

Several approaches to deal with this problem are possible, including:

- Keep all of them, so that the result is a set of ABoxes [Fagin, Ullman, Vardi 1983]
- Choose one ABox nondeterministically [Calvanese, Kharlamov, Nutt, Zheleznyakov, 2010]
- Adopt a "When In Doubt Throw It Out" (WIDTIO) approach

The result of inserting and deleting [L. and Savo, DL 2011]

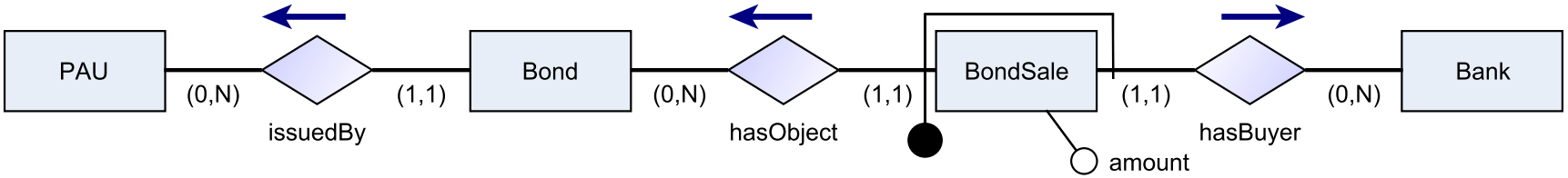
Definition

Let $\mathcal{U}$ be the set of all ABoxes accomplishing the insertion (deletion) of F into (from) $\langle \mathcal{O}, \mathcal{A} \rangle$ minimally, and let $\mathcal{A}'$ be an ABox. Then, $\langle \mathcal{O}, \mathcal{A}' \rangle$ is the result of changing $\langle \mathcal{O}, \mathcal{A} \rangle$ with the insertion (deletion) of F if

- $\mathcal{U}$ is empty, and $\langle \mathcal{O}, \mathsf{cl}_{\mathcal{O}}(\mathcal{A}') \rangle = \langle \mathcal{O}, \mathsf{cl}_{\mathcal{O}}(\mathcal{A}) \rangle$, or
- $\mathcal{U}$ is nonempty, and $\langle \mathcal{O}, \mathsf{cl}_{\mathcal{O}}(\mathcal{A}') \rangle = \langle \mathcal{O}, \bigcap \{ \mathsf{cl}_{\mathcal{O}}(\mathcal{A}_i) \mid \mathcal{A}_i \in U \} \rangle$.

- Up to logical equivalence, the result of changing $\langle \mathcal{O}, \mathcal{A} \rangle$ with the insertion or the deletion of F is unique.
- By definition, in the case where F is $\mathcal{O}$-inconsistent, the result of changing $\langle \mathcal{O}, \mathcal{A} \rangle$ with both the insertion and the deletion of F is logically equivalent to $\langle \mathcal{O}, \mathcal{A} \rangle$ itself.

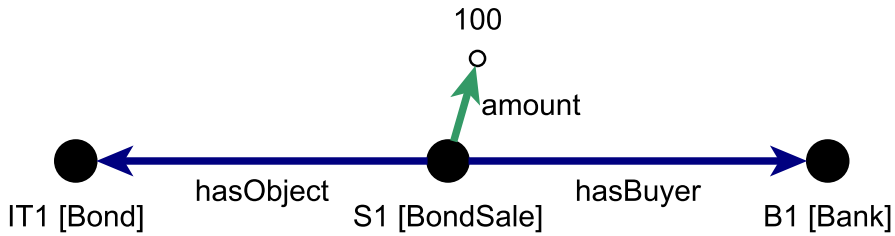
Example



The new TBox is:

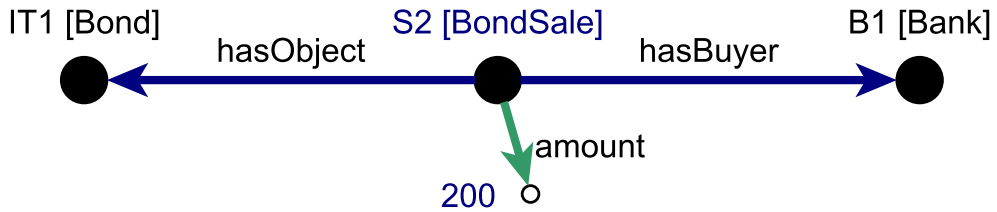
$\exists \texttt{issuedBy} \sqsubseteq \texttt{Bond}$
$\exists \texttt{issuedBy}^- \sqsubseteq \texttt{PAU}$
$\texttt{Bond} \sqsubseteq \exists \texttt{issuedBy}$
$\exists \texttt{hasObject} \sqsubseteq \texttt{BondSale}$
$\exists \texttt{hasObject}^- \sqsubseteq \texttt{Bond}$
$\texttt{BondSale} \sqsubseteq \exists \texttt{hasObject}$
$\exists \texttt{hasBuyer} \sqsubseteq \texttt{BondSale}$
$\exists \texttt{hasBuyer}^- \texttt{Bank}$
$\texttt{BondSale} \sqsubseteq \exists \texttt{hasBuyer}$
$\delta(\texttt{amount}) \sqsubseteq \texttt{BondSale}$
$\texttt{BondSale} \sqsubseteq \delta(\texttt{amount})$
$(funct\ \texttt{amount})$
$(funct\ \texttt{issuedBy})$
$(funct\ \texttt{hasObject})$
$(funct\ \texttt{hasBuyer})$
$(id\ \texttt{BondSale hasObject, hasBuyer})$

Consider the following ABox:



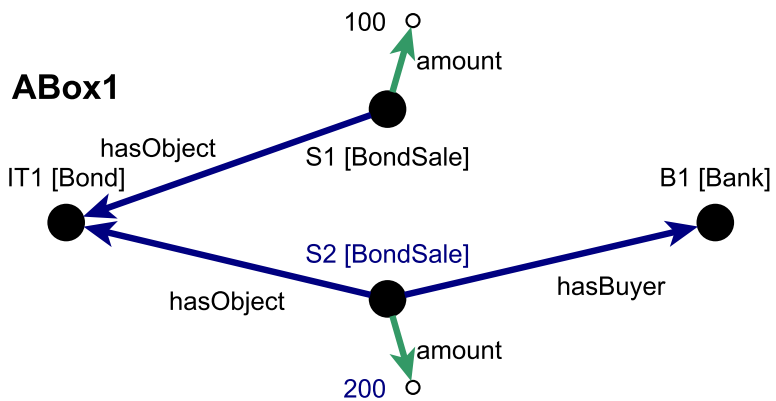
Example

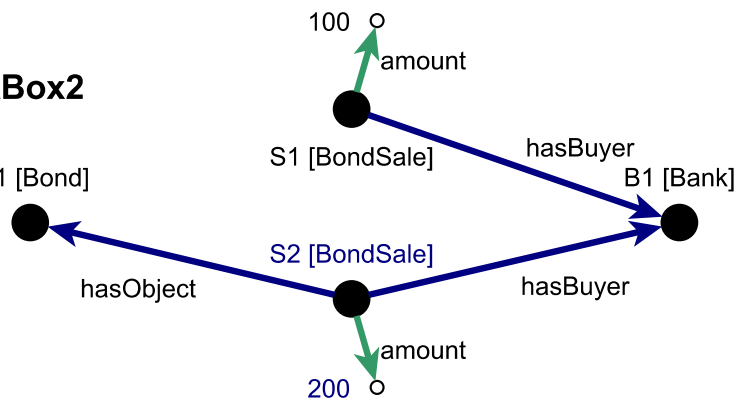
We want to update the instance level of our KB with the new information which states that the *bond* $IT1$ is sold to the *bank* B1 at the price 200 by means of the bond sale $S2$.



The new facts violate the identification assertion

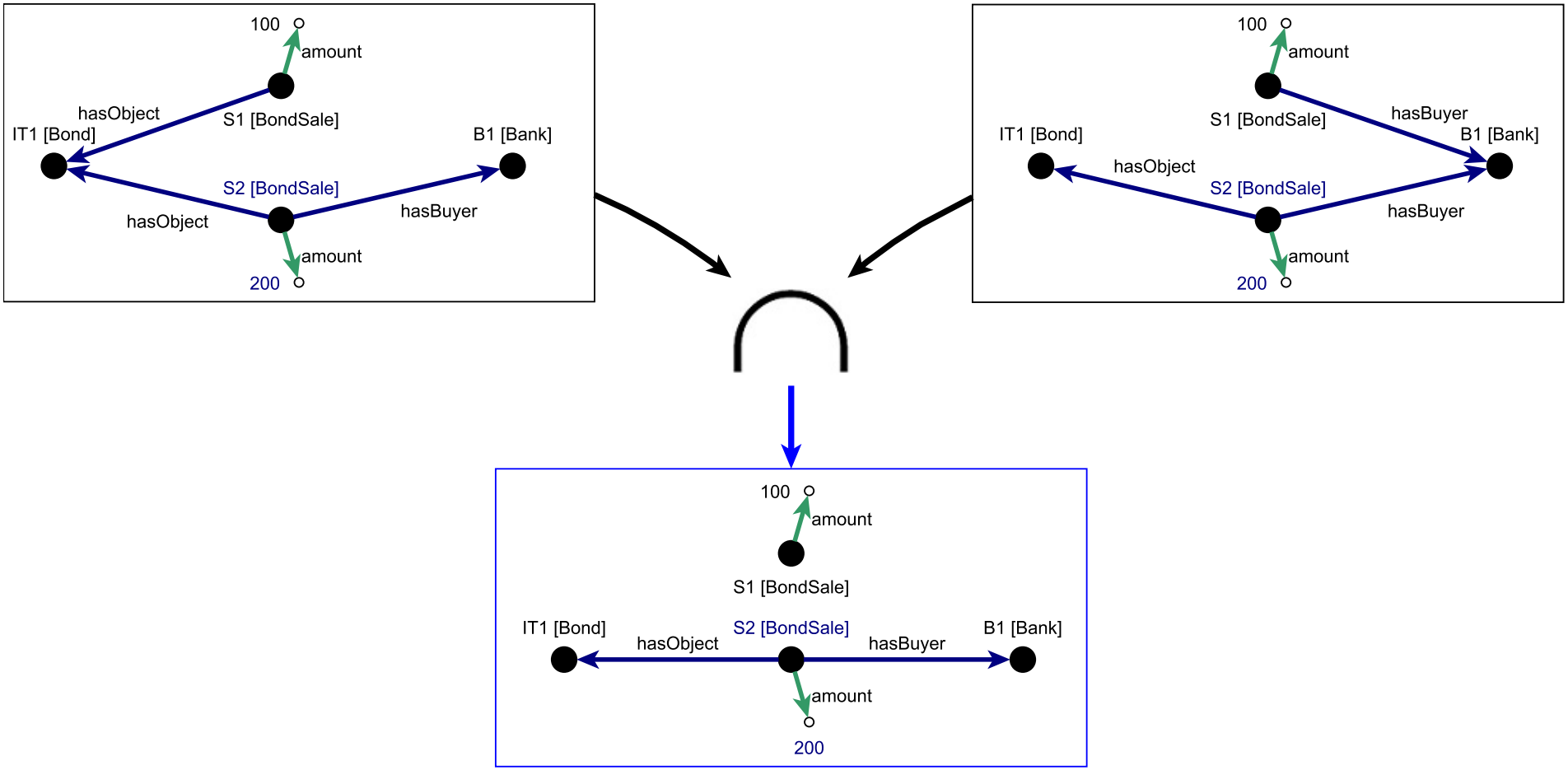
$$(id\ \texttt{BondSale}\ \texttt{hasObject}, \texttt{hasBuyer})$$





Example

With the *When In Doubt Throw It Out* (WIDTIO) approach:



Characterization of insertion in DL-Lite$_{A,id}$

We assume that $\mathcal{A}$ is $\mathcal{O}$-consistent, i.e., $\langle \mathcal{O}, \mathcal{A} \rangle$ is satisfiable.

Theorem

Let $\mathcal{A}'$ be an ABox. Then $\mathcal{A}'$ accomplishes the insertion of F from $\langle \mathcal{O}, \mathcal{A} \rangle$ minimally if and only if $\mathrm{cl}_{\mathcal{O}}(\mathcal{A}') = \mathrm{cl}_{\mathcal{O}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathrm{cl}_{\mathcal{O}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{O}$-consistent.

The above theorem immediately suggests an algorithm for computing the result of an update by insertion:

1. compute all ABoxes accomplishing the insertion minimally,
2. compute their intersection.

Challenge: compute the result without computing all ABoxes accomplishing the insertion minimally.

Basic idea for the insertion algorithm

Idea: look for atoms α that do not belong to at least one ABox accomplishing the insertion minimally, i.e., that do not belong to at least one maximal subset $\mathcal{A}''$ of $\mathrm{cl}_{\mathcal{O}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{O}$-consistent, and remove such atoms from $\mathrm{cl}_{\mathcal{O}}(\mathcal{A})$.

The next theorem is the key to our solution.

Theorem

Let α be an assertion in $\mathrm{cl}_{\mathcal{O}}(\mathcal{A}) \setminus \mathrm{cl}_{\mathcal{O}}(F)$. There exists a maximal subset Σ of $\mathrm{cl}_{\mathcal{O}}(\mathcal{A})$ such that $\Sigma \cup F$ is $\mathcal{O}$-consistent, and Σ does not contain α if and only if there is an $\mathcal{O}$-inconsistent set V in $\mathrm{cl}_{\mathcal{O}}(\mathcal{A}) \cup \mathrm{cl}_{\mathcal{O}}(F)$ such that $\alpha \in V$, and $F \cup (V \setminus \{\alpha\})$ is $\mathcal{O}$-consistent.

The algorithm for insertion

Algorithm *ComputeInsertion*$(\langle \mathcal{O}, \mathcal{A} \rangle, \mathcal{F})$
In a satisfiable *DL-Lite*$_{A,id}$ KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$, a finite set of ABox assertions F such that $\langle \mathcal{O}, F \rangle$ is satisfiable
Out a *DL-Lite*$_{A,id}$ KB

begin
  $F' = \emptyset$;
  foreach $\alpha \in \mathsf{cl}_{\mathcal{O}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{O}}(F)$ **do**
    if there exists an $\mathcal{O}$-inconsistent set V in $\mathsf{cl}_{\mathcal{O}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{O}}(F)$ s.t. $\alpha \in V$ and $\langle \mathcal{O}, F \cup (V \setminus \{\alpha\}) \rangle$ is satisfiable
    then $F' \leftarrow F' \cup \{\alpha\}$;
  return $\langle \mathcal{O}, F \cup \mathsf{cl}_{\mathcal{O}}(\mathcal{A}) \setminus F' \rangle$;
end

Theorem

ComputeInsertion$(\langle \mathcal{O}, \mathcal{A} \rangle, \mathcal{F})$ terminates, and computes the result of the insertion of F into $\langle \mathcal{O}, \mathcal{A} \rangle$ in polynomial time with respect to $|\mathcal{O} \setminus \mathcal{O}_{id}|$, $|\mathcal{A}|$, and $|F|$, and in exponential time with respect to $|\mathcal{O}_{id}|$.

Outline

Many challenges

- Still a lot to do for efficient query answering
  - Rewriting wrt mapping (even GAV mapping are problematic)
- Pushing the updates to the data sources
- Updating inconsistent OBDA systems [L. and Savo, ECAI 2012]
- *Ontology-based business intelligence* (more sophisticated queries)
- *Ontology-based data quality* (explanation)
- *Ontology-based data governance* (provenance, more sophisticated mapping languages)
- *Ontology-based service and process management* (process modeling – see the ACSI project)
- Natural language interface for querying
- Desperate need of effective tools for modeling both the ontology and the mapping, and for supporting their evolution
- Optique: A new European project on OBDA

Enriching the mapping languages: mapping intensional knowledge

Source $\mathcal{S}$:

`T-CarTypes`

Code	Name
T1	Coupé
T2	SUV
T3	Sedan
T4	Estate

`T-Cars`

CarCode	CarType	EngineSize	BreakPower	Color	TopSpeed
AB111	T1	2000	200	Silver	260
AF333	T2	3000	300	Black	200
BR444	T2	4000	400	Grey	220
AC222	T4	2000	125	Dark Blue	180
BN555	T3	1000	75	Light Blue	180
BP666	T1	3000	600	Red	240

Example

Ontology $\mathcal{O}$: $\texttt{Car} \sqsubseteq \texttt{Vehicle}$

Source $\mathcal{S}$:

T-CarTypes

Code	Name
T1	Coupé
T2	SUV
T3	Sedan
T4	Estate

T-Cars

CarCode	CarType	EngineSize	BreakPower	Color	TopSpeed
AB111	T1	2000	200	Silver	260
AF333	T2	3000	300	Black	200
BR444	T2	4000	400	Grey	220
AC222	T4	2000	125	Dark Blue	180
BN555	T3	1000	75	Light Blue	180
BP666	T1	3000	600	Red	240

Mapping $\mathcal{M}$:

- $\{y \mid \texttt{T-CarTypes}(x, y)\} \rightsquigarrow y \sqsubseteq \texttt{Car}$
- $\{(x, v, z) \mid \texttt{T-Cars}(x, y, t, u, v, q) \wedge \texttt{T-CarTypes}(y, z)\} \rightsquigarrow z(x)$
- $\{(x, y) \mid \texttt{T-CarTypes}(z_1, x) \wedge \texttt{T-CarTypes}(z_2, y) \wedge x \neq y\} \rightsquigarrow x \sqsubseteq \neg y$

The ontology $\mathcal{O}$ is enriched through $\mathcal{M}$ and $\mathcal{S}$.

Higher-order Description Logics

Technically, we need higher-order logic (e.g., $Hi(\textbf{DL-Lite}_{\mathcal{R}})$ [De Giacomo et al, AAAI 2011, Di Pinto et al, AAAI 2012])

Higher-order queries become natural, e.g.:

Example

Interesting queries that can be posed to $\langle \mathcal{S}, \mathcal{M} \rangle$ exploit the higher-order nature of the system:

- Return all the instances of *Car*, each one with its own type:
  $q(x, y) \leftarrow y(x), \texttt{Car}(x)$
- Return all the concepts which car *AB111* is an instance of:
  $q(x) \leftarrow x(AB111)$

Acknowledgements

- Grant from IBM, European Project TONES, Italian project TOCAI
- The original group
  - Diego Calvanese (now at the Free University of Bozen-Bolzano)
  - Giuseppe De Giacomo
  - Domenico Lembo
  - Riccardo Rosati
- Additional people in the current group
  - Antonella Poggi
  - Marco Ruzzi
  - Domenico Fabio Savo
  - Cristina Civili
  - Floriana Di Pinto
  - Valerio Santarelli
  - Riccardo Mancini
  - Lorenzo Lepore
- Many students

Well my ship's been split to splinters and it's sinking fast
I'm drowning in the poison, got no future, got no past
But my heart is not weary, it's light and it's free
I've got nothing but affection for all those who've sailed with me
Bob Dylan

Acknowledgements

We miss Marco

Marco Cadoli
(5/12/1965 – 21/11/2006)

I was deeply saddened to learn that Marco died yesterday. I looked at his web page and again saw this young, intelligent and handsome man whose work I admired. I looked over his vitae and was again impressed by the research he contributed to the computational logic community.

Jack Minker November 22, 2006