Data Integration

Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica Antonio Ruberti

SAPIENZA
UNIVERSITÀ DI ROMA

Global scientific data infrastructures: The big data challenges

Capri, May 12–13, 2011

Outline

Outline

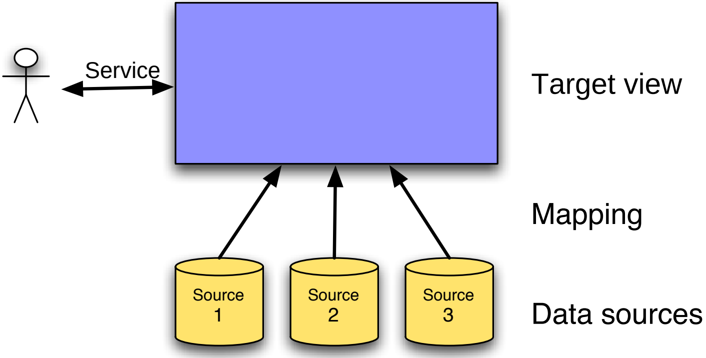
Information integration: general definition

The goal of information integration is to provide a unified and transparent view to a collection of data stored in *multiple*, *autonomous*, and *heterogeneous* data sources.

The unified view is achieved through a target schema (or, global schema), and is realized either through

- a materialized database – data exchange, or data warehousing
- a virtualization mechanism based on querying – virtual data integration, or simply **data integration**
- a mapping mechanism among a set of networked peers – peer-to-peer data exchange and integration

Data integration architecture



Recurring theme: choosing the right language for queries, target schema, mapping assertions.

The distingushing feature of data integration

Other methods/techniques for distributing/moving/merging data:

- Distributed database systems,
- Data replication,
- ETL (Extraction, Trasformation and Loading)
- Data federation
- Data mash-up

Distinguishing feature

A data integration system is based on a **structure** accomodating data in the target view, and such a structure should be

- declaratively specified,
- decoupled from the sources,
- linked to the sources by means of mappings.

Ontology-based data integration

In OBDI, the target schema is expressed in terms of an ontology.

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

- $\mathcal{O}$ is the ontology, expressed as a TBox in OWL 2 DL (or its DL counterpart $\mathcal{SROIQ}(D)$)
- $\mathcal{S}$ is the source database
- $\mathcal{M}$ is a set of GLAV mapping assertions, each one of the form

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

  where

  - $\Phi(\vec{x})$ is a query over $\mathcal{S}$, returning values for $\vec{x}$
  - $\Psi(\vec{x})$ is a 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 $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ if:

- $\mathcal{I}$ is a model of $\mathcal{O}$;
- $\mathcal{I}$ satisfies every assertion in $\mathcal{M}$ wrt $\mathcal{S}$, where $\mathcal{I}$ satisfies the assertion $\Phi(\vec{x}) \leadsto \Psi(\vec{x})$ wrt a database $\mathcal{S}$, if the sentence
$$\forall \vec{x} \; (\Phi(\vec{x}) \rightarrow \Psi(\vec{x}))$$
is true in $\mathcal{I} \cup \mathcal{S}$.

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

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

Some challenges

Challenges depending on the existence of the target structure and its decoupling from the sources:

- Incompleteness (data do not adhere to the target schema because of lack of data)
- Inconsistency (data do not adhere to the target schema because of contradictions)
- Acquisition of intensional knowledge from the sources

We will address these challenges in the context of ontology-based data integration

Outline

Query answering under incomplete informstion

Example

$\mathcal{O}$:

$$\texttt{University} \sqsubseteq \exists\texttt{HasRector}$$

$\mathcal{M}$:

$$\texttt{R1(x,y,z)} \rightsquigarrow \texttt{University(x)}$$
$$\texttt{R2(x,y,z,w)} \rightsquigarrow \texttt{HasRector(x,z)}$$

Query: $\{\ x \mid \exists y\ \texttt{University}(x) \wedge \texttt{HasRector}(x,y)\ \}$

The problem of answering queries under incomplete information shows up

Query language for user queries

- Answering FOL queries is **undecidable**, even if the ontology is empty, and the set of mappings are very simple.
- Unions of conjunctive queries (UCQs) do not suffer from this problem.
- We can go beyond unions of conjunctive queries, but we have to carefully choose the semantics of nonmonotonic queries.

Query languages for the mappings

$\mathcal{O}$	lhs of $\mathcal{M}$	rhs of $\mathcal{M}$	Query language	Query answering
$\emptyset$	single atom	FOL	single atom	undecidable (1)
$\emptyset$	single atom	UCQ	single atom	NP-complete (2)
$\emptyset$	FOL	CQ	UCQ	AC^0 (3)

(1) *(Abiteboul & Duschka, PODS'98)*
(2) *(van Der Meyden, TCS'93; Abiteboul & Duschka, PODS'98)*
(3) *(Duschka & Genesereth, PODS'97; Pottinger & Levy VLDBJ 2001)*

We measure the computational complexity of query answering with respect to the size of the data $\mathcal{S}$

Note: $AC^0 \subseteq$ LogSpace, and going beyond LogSpace means going beyond relational databases

Query answering in Description Logic Ontologies

DL	Data complexity of query answering
$\mathcal{SROIQ}(D)$	? (1)
$\mathcal{SHIQ}(D)$	coNP-complete (2)
?	PTIME (3)
?	AC^0 (3)

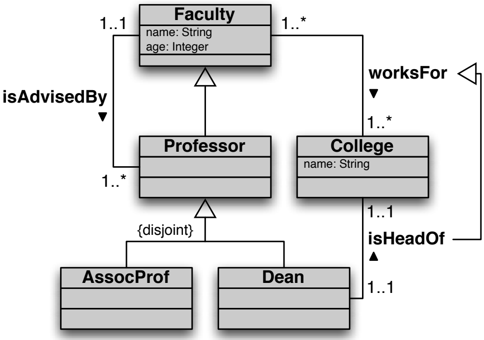
(1) It is in fact open whether answering CQs over OWL 2 DL (i.e., $\mathcal{SROIQ}(D)$) ontologies is decidable.

(2) *(Hustadt & al., IJCAI'05; Glimm & al., JAIR'08; Ortiz & al., JAIR'08)*. In fact, *(Calvanese & al., KR'06)* show coNP-hardness for very simple languages (fragments of OWL 2 DL) allowing for union.

(3) Question: Are there significative fragments of OWL 2 DL for which answering CQs is tractable/has the same complexity as SQL query evaluation?

Tractable Ontology Languages

- Three polynomially tractable profiles of OWL 2 DL:
  - OWL 2 QL (*DL-Lite*$_\mathcal{R}$, one of the members of the *DL-Lite* family)
  - OWL 2 EL
  - OWL 2 RL
- Datalog+–
- Weakly acyclic TGDs
- ...



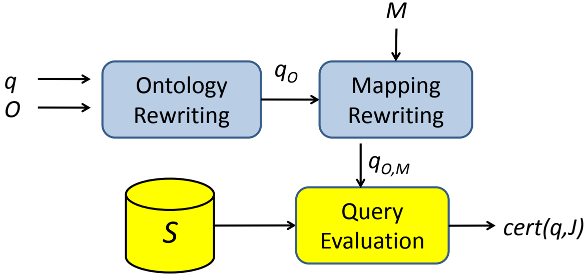
$$\begin{array}{rcl}
\mathsf{Professor} & \sqsubseteq & \mathsf{Faculty} \\
\mathsf{AssocProf} & \sqsubseteq & \mathsf{Professor} \\
\mathsf{Dean} & \sqsubseteq & \mathsf{Professor} \\
\mathsf{AssocProf} & \sqsubseteq & \neg\mathsf{Dean} \\
\exists\mathsf{worksFor} & \sqsubseteq & \mathsf{Faculty} \\
\exists\mathsf{worksFor}^- & \sqsubseteq & \mathsf{College} \\
\mathsf{Faculty} & \sqsubseteq & \exists\mathsf{worksFor} \\
\mathsf{College} & \sqsubseteq & \exists\mathsf{worksFor}^- \\
\exists\mathsf{isHeadOf} & \sqsubseteq & \mathsf{Dean} \\
\exists\mathsf{isHeadOf}^- & \sqsubseteq & \mathsf{College} \\
\mathsf{Dean} & \sqsubseteq & \exists\mathsf{isHeadOf} \\
\mathsf{College} & \sqsubseteq & \exists\mathsf{isHeadOf}^- \\
\mathsf{isHeadOf} & \sqsubseteq & \mathsf{worksFor} \\
 & \vdots &
\end{array}$$

UML attributes can be captured considering the extension of *DL-Lite*$_\mathcal{R}$ to data properties.

Query answering in $DL\text{-}Lite_{\mathcal{R}}$

Based on query rewriting – given an (U)CQ q, and $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$:

1. **Perfect reformulation**: rewrite q into the perfect reformulation $q_{\mathcal{O}}$ of q w.r.t. $\mathcal{O}$, which turns out to be a UCQ – $q_{\mathcal{O}}$ is such that $cert(q, \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle) = cert(q_{\mathcal{O}}, \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle)$
2. **Unfolding**: compute the rewriting $q_{\mathcal{O},\mathcal{M}}$ of $q_{\mathcal{O}}$ w.r.t. $\mathcal{M}$, which is a query over $\mathcal{S}$ – $q_{\mathcal{O},\mathcal{M}}$ is such that $cert(q_{\mathcal{O}}, \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle) = q^{\mathcal{S}}_{\mathcal{O},\mathcal{M}}$
3. **Evaluation**: evaluate $q_{\mathcal{O},\mathcal{M}}$ over the source database $\mathcal{S}$.



Complexity: **$\mathbf{AC}^0$** in the size of the **database** $\mathcal{S}$, in fact FOL-rewritable.

Beyond $DL\text{-}Lite_{\mathcal{R}}$: results on data complexity

	lhs	rhs	funct.	Prop. incl.	Data complexity of query answering
0	$DL\text{-}Lite_{\mathcal{A},id}$		–	$\surd$	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**

- $DL\text{-}Lite_{\mathcal{A},id}$ 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**.

Outline

The problem of inconsistency

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

One popular approach to dealing with inconsistency in data warehousing is data cleaning, but in OBDI data cleaning is unfeasible

Question

How to handle classically-inconsistent data integration systems in a meaningful way?

The notion of repair

Definition

A **repair** r of a database D under integrity constraints IC is a database (over the same schema) such that:

- $r \models IC$,
- there is no database r' such that $r' \models IC$, and r' is preferred to r, relative to some preference order.

Several preference orders have been proposed, based on:

- symmetric difference
- cardinality
- value modification
- metric distance
- ...

Inconsistent-tolerant semantics

Let $\mathcal{M}(\mathcal{S})$ denote the the minimal universal solution of $\mathcal{M}$ with respect to $\mathcal{S}$.

Example

Consider $\mathcal{J} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$, with $\mathcal{O}$:

$$\{\mathsf{Mechanic} \sqsubseteq \mathsf{TeamMbr}, \quad \mathsf{Driver} \sqsubseteq \mathsf{TeamMbr}, \quad \exists \mathsf{drives} \sqsubseteq \mathsf{Driver}, \quad \exists \mathsf{drives}^- \sqsubseteq \mathsf{Car}, \quad \mathsf{Driver} \sqsubseteq \neg \mathsf{Mechanic}\}$$

and

$$\mathcal{M}(\mathcal{S}) = \{\ \mathsf{Driver}(felipe), \mathsf{Mechanic}(felipe), \mathsf{TeamMbr}(felipe), \mathsf{drives}(felipe, ferrari)\ \}.$$

Inconsistent-tolerant semantics

Intuitively, a repair for $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ is an ABox (set of ground facts) $\mathcal{A}_R$ such that $\langle \mathcal{O}, \mathcal{A}_R \rangle$ is satisfiable, and $\mathcal{A}_R$ "minimally" differs from $\mathcal{M}(\mathcal{S})$.

Definition (Repair)

A **repair** of $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ is a set $\mathcal{A}$ of extensional assertions such that:

1. $\mathcal{A} \subseteq \mathcal{M}(\mathcal{S})$
2. $Mod(\langle \mathcal{O}, \mathcal{A} \rangle) \neq \emptyset$
3. no $\mathcal{A}'$ exists such that
    - $\mathcal{A} \subset \mathcal{A}' \subseteq \mathcal{M}(\mathcal{S})$, and
    - $Mod(\langle \mathcal{O}, \mathcal{A}' \rangle) \neq \emptyset$.

The set of repairs for $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ is denoted by $Rep(\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle)$.

Inconsistent-tolerant semantics

Example

$$\mathcal{O} = \{\text{Mechanic} \sqsubseteq \text{TeamMbr}, \quad \text{Driver} \sqsubseteq \text{TeamMbr}, \quad \exists\text{drives} \sqsubseteq \text{Driver}, \quad \exists\text{drives}^- \sqsubseteq \text{Car}, \quad \text{Driver} \sqsubseteq \neg\text{Mechanic}\}$$

$$\mathcal{M}(\mathcal{S}) = \{ \text{Driver}(felipe), \text{Mechanic}(felipe), \text{TeamMbr}(felipe), \text{drives}(felipe, ferrari) \}.$$

Rep$(\mathcal{J})$:

r_1=$\{$Driver$(felipe)$, drives$(felipe, ferrari)$, TeamMbr$(felipe)\}$;
r_2=$\{$Mechanic$(felipe)$, TeamMbr$(felipe)\}$.

Inconsistent-tolerant semantics

Example

$\mathcal{O} = \{\text{Mechanic} \sqsubseteq \text{TeamMbr}, \quad \text{Driver} \sqsubseteq \text{TeamMbr}, \quad \exists\text{drives} \sqsubseteq \text{Driver},$
$\quad \exists\text{drives}^- \sqsubseteq \text{Car}, \quad \text{Driver} \sqsubseteq \neg\text{Mechanic}\}$

$\mathcal{M}(\mathcal{S}) = \{ \text{Driver}(felipe), \text{Mechanic}(felipe), \text{TeamMbr}(felipe),$
$\quad \text{drives}(felipe, ferrari) \}.$

Rep$(\mathcal{J})$:

$r_1 = \{\text{Driver}(felipe), \text{drives}(felipe, ferrari), \text{TeamMbr}(felipe)\};$
$r_2 = \{\text{Mechanic}(felipe), \text{TeamMbr}(felipe)\}.$

$$\mathcal{J} \models_R \text{TeamMbr}(felipe)$$

Inconsistent-tolerant semantics

Problems:

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

Proposition

Let $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDIS, and let α be a ground fact. Deciding whether $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle \models \alpha$ in all possible repairs is coNP-complete with respect to data complexity.

Idea

Take 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 – WIDTIO).

Inconsistent-tolerant query answering

A tractable method for answering queries posed to $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ according to the WIDTIO semantics:

- Avoid computing the intersection, and rewrite the query q into q' in such a way that $\mathcal{J} \models_{WIDTIO} q$ is equivalent to $\mathcal{J} \models q'$.

In our setting, with *DL-Lite*$_{\mathcal{R}}$:

problem	R-semantics	$WIDTIO$-semantics
single atom query	coNP-complete	in AC_0
UCQ answering	coNP-complete	in AC_0

Outline

Example

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

Motivating example

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 $\mathcal{M}$:

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

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

Higher-order Description Logics

Technically, we need higher-order logic – $Hi(\textbf{DL-Lite}_{\mathcal{R}})$

We also need higher-order queries, such as:

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:
  $q(x) \leftarrow x(AB111)$

Mapping both extensional and intensional knowledge

We denote by $\mathcal{M}_{\mathcal{A}}$ the part of the mapping assertions with extensional assertions in the rhs.

Proposition

Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{M} \rangle$ be a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ OBDIS, and let Q be an instance higher-order UCQ. Deciding whether $\mathcal{K} \models Q$ is in AC^0 with respect to the size of $\mathcal{M}_{\mathcal{A}}$, is in PTIME with respect to the size of $\mathcal{M} \setminus \mathcal{M}_A$, and is NP-complete with respect to the size of $\mathcal{K} \cup Q$.

Outline

Many other challenges

- Methodology to build data integration systems
  - How to write "good" ontologies
  - How to write "good" mappings
- Tools supporting data integration
  - Design time
  - Run time (query optimization)
- Privacy preserving query answering
- Object identification (record matching)
- Other types of mapping languages?
- Other inconsistent tolerant semantics?
- Updates/services/processes on the target schema?
- ...