Data Integration through Ontologies

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica “Antonio Ruberti”

Università di Roma “La Sapienza”

View-based query processing

Diego Calvanese, Giuseppe De Giacomo, Georg Gottlob, Maurizio Lenzerini, Riccardo Rosati

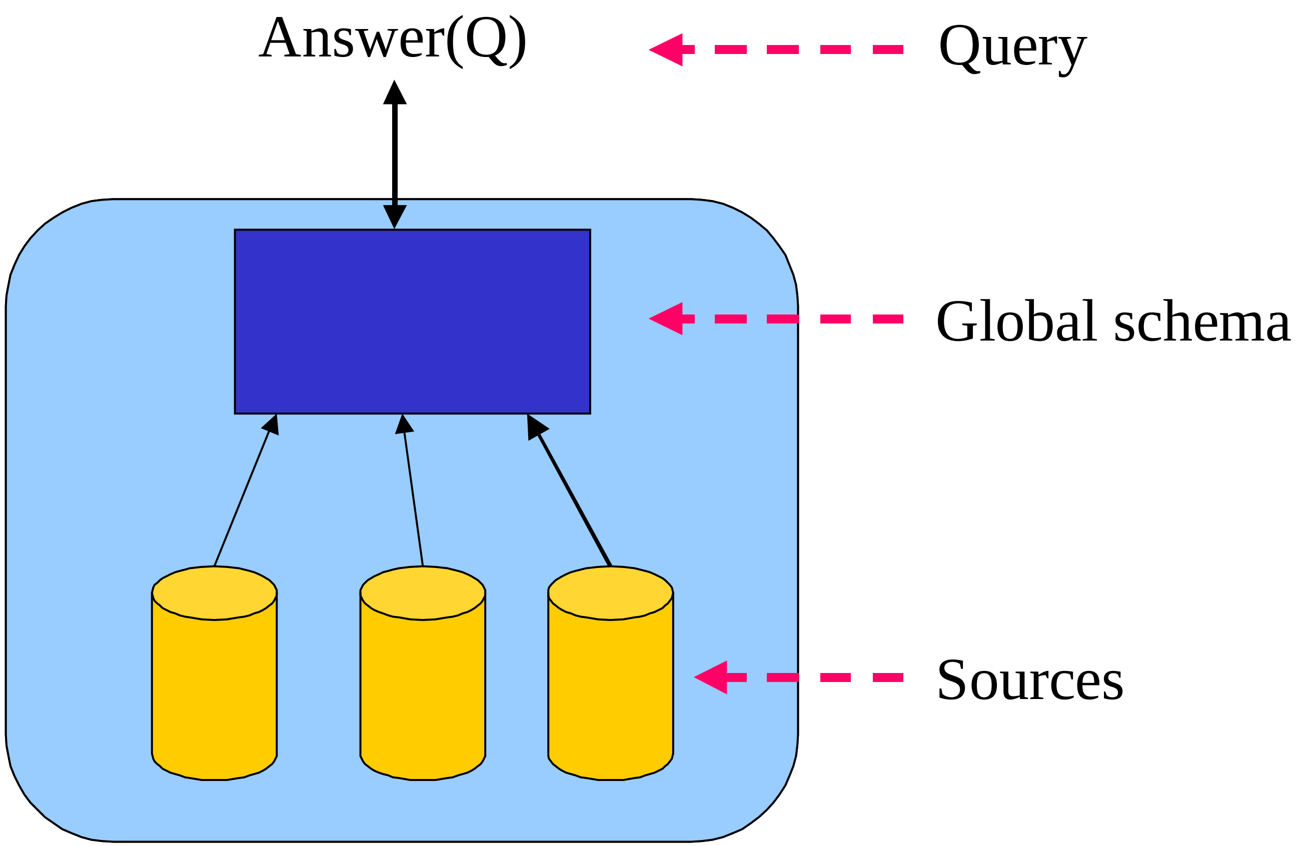
PhD Course at DIS, UNIROMA1

September-October 2005

Data integration through Ontologies: outline

- Introduction to data integration through ontologies
- Ontologies: conceptual schema languages, description logics
- Query answering in description logics
- Data complexity tradeoff: a concrete interpretation
- Quonto

Data integration



Logical transparency

Basic ingredients for achieving logical transparency:

- The global schema provides a conceptual view that is independent from the sources
- The global schema is described with a semantically rich formalism
- The mappings are the crucial tools for realizing the independence of the global schema from the sources
- Obviously, the formalism for specifying the mapping is also a crucial point

All the above aspects are not appropriately dealt with by current tools. This means that data integration cannot be simply addressed on a tool basis.

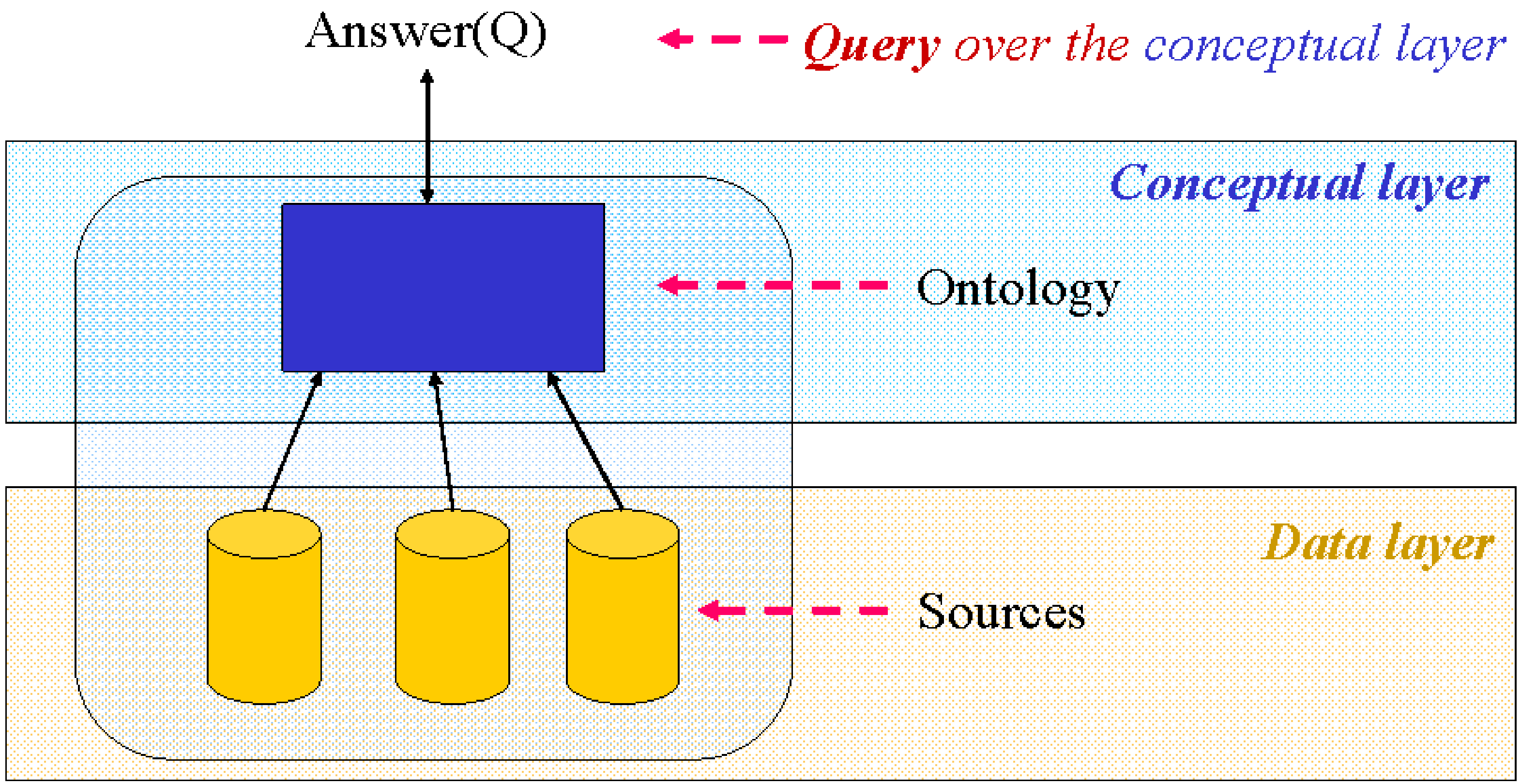
Logical transparency using an ontology

Basic ingredients for achieving logical transparency:

- The **global schema** provides a **conceptual view** -an ontology- that is independent from the sources
- The **global schema** is described with a **semantically rich formalism** -an ontology language-
- The mappings are the crucial tools for realizing the independence of the global schema from the sources
- Obviously, the formalism for specifying the mapping is also a crucial point

All the above aspects are not appropriately dealt with by current tools. This means that data integration cannot be simply addressed on a tool basis.

Data integration through an ontology



Formal framework for data integration

A **data integration system** $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{G}$ is the global schema -now is an ontology-

  The global schema is a logical theory over an alphabet $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{S}$ is the source schema

  The source schema is constituted simply by an alphabet $\mathcal{A}_{\mathcal{S}}$ *disjoint from* $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{G}$ *Different approaches to the specification of mapping*

Semantics of a data integration system (as before)

Which are the databases that satisfy $\mathcal{I}$, i.e., which are the logical models of $\mathcal{I}$?

The databases that satisfy $\mathcal{I}$ are logical interpretations for $\mathcal{A}_\mathcal{G}$ (called global databases). We refer only to databases over a fixed infinite domain Γ of constants.

Let $\mathcal{C}$ be **a source database** over Γ (also called source model), fixing the extension of the predicates of $\mathcal{A}_\mathcal{S}$ (thus modeling the data present in the sources).

The set of models of (i.e., databases for $\mathcal{A}_\mathcal{G}$ that satisfy) $\mathcal{I}$ relative to $\mathcal{C}$ is:

$$sem^\mathcal{C}(\mathcal{I}) = \{\ \mathcal{B} \mid \mathcal{B} \text{ is a } \mathcal{G}\text{-model (i.e., a global database that is legal wrt } \mathcal{G}) \text{ and is an } \mathcal{M}\text{-model wrt } \mathcal{C} \text{ (i.e., satisfies } \mathcal{M} \text{ wrt } \mathcal{C})\ \}$$

What it means to satisfy $\mathcal{M}$ wrt $\mathcal{C}$ depends on the nature of the mapping $\mathcal{M}$.

Semantics of queries to $\mathcal{I}$ (as before)

A **query** q of arity n is a formula with n free variables.

If $\mathcal{D}$ is a database, then $q^{\mathcal{D}}$ denotes the extension of q in $\mathcal{D}$ (i.e., the set of n-tuples that are valuations in Γ for the free variables of q that make q true in $\mathcal{D}$).

If q is a query of arity n posed to a data integration system $\mathcal{I}$ (i.e., a formula over $\mathcal{A}_{\mathcal{G}}$ with n free variables), then the set of **certain answers to q wrt $\mathcal{I}$ and $\mathcal{C}$** is

$$cert(q, \mathcal{I}, \mathcal{C}) = \{(c_1, \ldots, c_n) \in q^{\mathcal{B}} \mid \forall \mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})\}.$$

Note: query answering is logical implication.

Note: complexity will be mainly measured wrt the size of the source database $\mathcal{C}$, and will refer to the problem of deciding whether $\vec{\mathbf{c}} \in cert(q, \mathcal{I}, \mathcal{C})$, for a given $\vec{\mathbf{c}}$.

The mapping

How is the mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{G}$ specified?

- Are the sources defined in terms of the global schema?

  Approach called **source-centric**, or **local-as-view**, or **LAV**

- Is the global schema defined in terms of the sources?

  Approach called **global-schema-centric**, or **global-as-view**, or **GAV**

- A mixed approach?

  Approach called **GLAV**

Note: *Also, we also must take into account mismatch between objects in the ontology and values in the sources!!!* (For lack of time we will not consider it here.)

Data integration through Ontologies: outline

- Introduction to data integration through ontologies
- Ontologies: conceptual schema languages, description logics
- Query answering in description logics
- Data complexity tradeoff: a concrete interpretation
- Quonto

Ontologies

- **Ontologies** are **formal specifications** of a **conceptualization** of a particular domain
- Envisioned to play a major role in supporting **information sharing** across networks by making explicit the **semantics of information** at various sites
- Pioneered in Computer Science by researchers in Artificial Intelligence, where they have become a popular research topic at the beginning of the 1990s (see, e.g., WordNet and CYC). More recently, the notion of ontology has spread across several other research fields such as intelligent information integration, cooperative information systems, information retrieval, knowledge management.
- Married with **Description Logics**, they are advocated as the key technology for realizing the **Semantic Web**. Standardization efforts have started within W3C: RDFS, OWL

Ontologies

- **Ontologies** are used to represent information at the **conceptual level**...
- ... in terms of **classes/concepts/entities** and **relationships** between them
- Observe that such a form of representation is almost universally recognized as the most prominent in Computer Science
  - **UML class diagrams** in Software Engineering
  - **ER diagrams** in databases and information systems
  - **Frame systems** in AI
- Ontologies are typically expressed in **logic**:
  - First Order Logic
  - **Description Logics**: a specialized formalism (typically a fragment of FOL) for expressing knowledge in terms of classes and relationships

UML - FOL - Description Logics

Lets gain an intuitive understanding of the relationships among the above formalisms.

Slides from:

ESSLLI'03 Course on

Description Logics for Conceptual Data Modeling in UML

Requirements: We are interested in building a software application to manage filmed scenes for realizing a movie, by following the so-called “Hollywood Approach”.

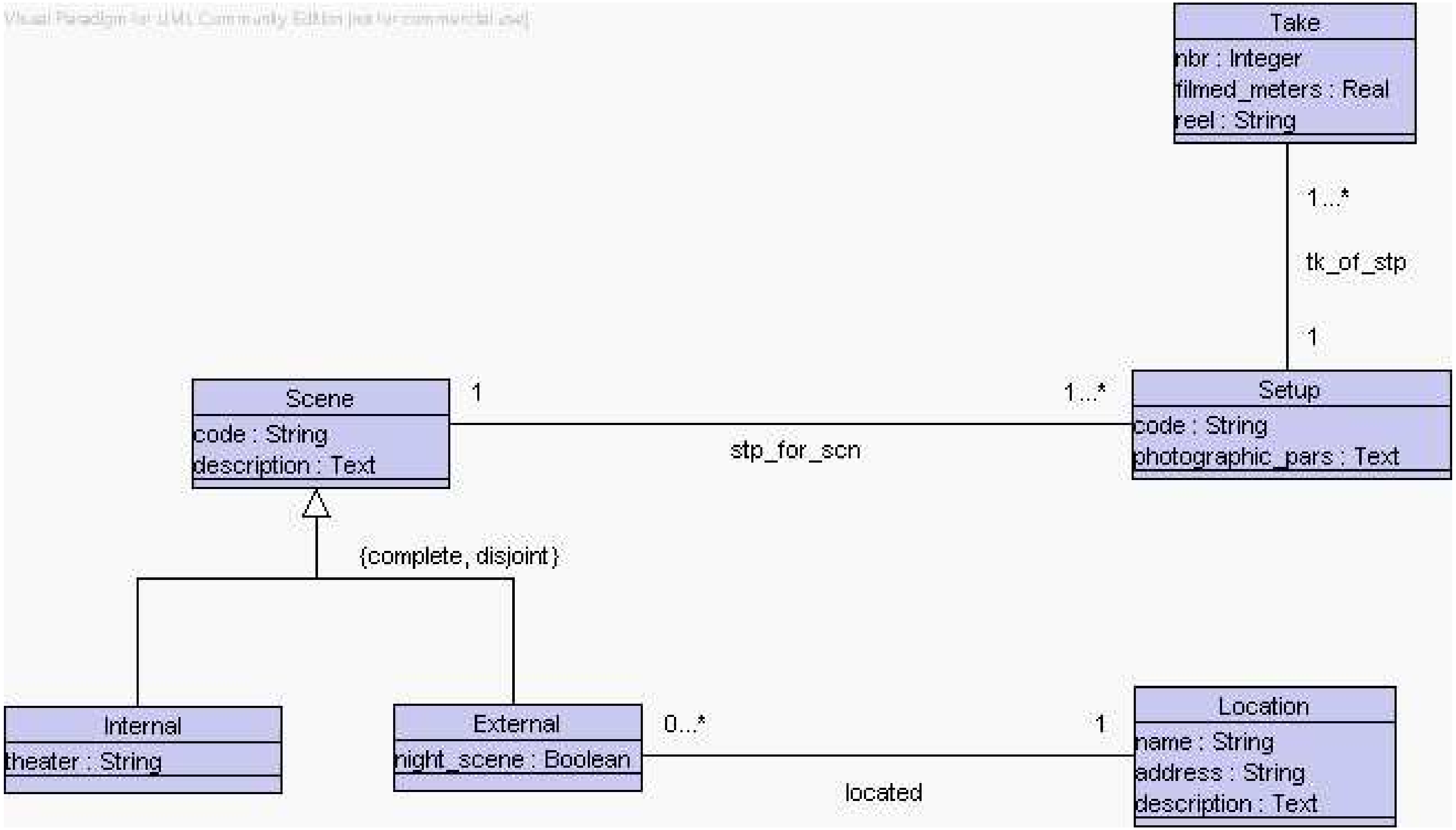
Every **scene** is identified by a code (a string) and it is described by a text in natural language.

Every scene is filmed from different positions (at least one), each of this is called a **setup**. Every setup is characterized by a code (a string) and a text in natural language where the photographic parameters are noted (e.g., aperture, exposure, focal length, filters, etc.). Note that a setup is related to a single scene.

For every setup, several **takes** may be filmed (at least one). Every take is characterized by a (positive) natural number, a real number representing the number of meters of film that have been used for shooting the take, and the code (a string) of the reel where the film is stored. Note that a take is associated to a single setup.

Scenes are divided into **internals** that are filmed in a theater, and **externals** that are filmed in a **location** and can either be “day scene” or “night scene”. Locations are characterized by a code (a string) and the address of the location, and a text describing them in natural language.

Write a precise specification of this domain using any formalism you like.


Take
nbr : Integer
filmed_meters : Real
reel : String
1...*
tk_of_stp
1
Scene
code : String
description : Text
1
stp_for_scn
1...*
Setup
code : String
photographic_pars : Text
{complete, disjoint}
Internal
theater : String
External
night_scene : Boolean
0...*
located
1
Location
name : String
address : String
description : Text

Alphabet:

$Scene(x), Setup(x), Take(x), Internal(x), External(x), Location(x), stp\_for\_scn(x,y), ck\_of\_stp(x,y), located(x,y), \ldots.$

Axioms:

$\forall x, y. (Scene(x) \land code(x,y)) \supset String(y)$

$\forall x, y. (Scene(x) \land description(x,y)) \supset Text(y)$

$\forall x, y. (Setup(x) \land code(x,y)) \supset String(y)$

$\forall x, y. (Setup(x) \land photographic\_pars(x,y)) \supset Text(y)$

$\forall x, y. (Take(x) \land nbr(x,y)) \supset Integer(y)$

$\forall x, y. (Take(x) \land filmed\_meters(x,y)) \supset Real(y)$

$\forall x, y. (Take(x) \land reel(x,y)) \supset String(y)$

$\forall x, y. (Internal(x) \land theater(x,y)) \supset String(y)$

$\forall x, y. (External(x) \land night\_scene(x,y)) \supset Boolean(y)$

$\forall x, y. (Location(x) \land name(x,y)) \supset String(y)$

$\forall x, y. (Location(x) \land address(x,y)) \supset String(y)$

$\forall x, y. (Location(x) \land description(x,y)) \supset Text(y)$

$\forall x. Scene(x) \supset (1 \leq \sharp\{y \mid code(x,y)\} \leq 1)$

$\ldots$

$\forall x, y. stp\_for\_scn(x,y) \supset Setup(x) \land Scene(y)$

$\forall x, y. tk\_of\_stp(x,y) \supset Take(x) \land Setup(y)$

$\forall x, y. located(x,y) \supset External(x) \land Location(y)$

$\forall x. Setup(x) \supset 1 \leq \sharp\{y \mid stp\_for\_scn(x,y)\} \leq 1$

$\forall y. Scene(y) \supset 1 \leq \sharp\{x \mid stp\_for\_scn(x,y)\}$

$\forall x. Take(x) \supset 1 \leq \sharp\{y \mid tk\_of\_stp(x,y)\} \leq 1$

$\forall x. Setup(y) \supset 1 \leq \sharp\{x \mid tk\_of\_stp(x,y)\}$

$\forall x. External(x) \supset 1 \leq \sharp\{y \mid located(x,y)\} \leq 1$

$\forall x. Internal(x) \supset Scene(x)$

$\forall x. External(x) \supset Scene(x)$

$\forall x. Internal(x) \supset \neg External(x)$

$\forall x. Scene(x) \supset Internal(x) \lor External(x)$

Encoding of classes and attributes

$$
\begin{array}{rcl}
\mathsf{Scene} & \sqsubseteq & \forall \mathsf{code}.\mathsf{String} \sqcap \exists \mathsf{code} \sqcap (\leq 1\, \mathsf{code}) \\
\mathsf{Scene} & \sqsubseteq & \forall \mathsf{description}.\mathsf{Text} \sqcap \exists \mathsf{description} \sqcap (\leq 1\, \mathsf{description}) \\
\mathsf{Internal} & \sqsubseteq & \forall \mathsf{theater}.\mathsf{String} \sqcap \exists \mathsf{theater} \sqcap (\leq 1\, \mathsf{theater}) \\
\mathsf{External} & \sqsubseteq & \forall \mathsf{night\_scene}.\mathsf{Boolean} \sqcap \exists \mathsf{night\_scene} \sqcap (\leq 1\, \mathsf{night\_scene}) \\
\mathsf{Take} & \sqsubseteq & \forall \mathsf{nbr}.\mathsf{Integer} \sqcap \exists \mathsf{nbr} \sqcap (\leq 1\, \mathsf{nbr}) \\
\mathsf{Take} & \sqsubseteq & \forall \mathsf{filmed\_meters}.\mathsf{Real} \sqcap \exists \mathsf{filmed\_meters} \sqcap (\leq 1\, \mathsf{filmed\_meters}) \\
\mathsf{Take} & \sqsubseteq & \forall \mathsf{reel}.\mathsf{String} \sqcap \exists \mathsf{reel} \sqcap (\leq 1\, \mathsf{reel}) \\
\mathsf{Setup} & \sqsubseteq & \forall \mathsf{code}.\mathsf{String} \sqcap \exists \mathsf{code} \sqcap (\leq 1\, \mathsf{code}) \\
\mathsf{Setup} & \sqsubseteq & \forall \mathsf{photographic\_pars}.\mathsf{Text} \sqcap \exists \mathsf{photographic\_pars} \sqcap (\leq 1\, \mathsf{photographic\_pars}) \\
\mathsf{Location} & \sqsubseteq & \forall \mathsf{name}.\mathsf{String} \sqcap \exists \mathsf{name} \sqcap (\leq 1\, \mathsf{name}) \\
\mathsf{Location} & \sqsubseteq & \forall \mathsf{address}.\mathsf{String} \sqcap \exists \mathsf{address} \sqcap (\leq 1\, \mathsf{address}) \\
\mathsf{Location} & \sqsubseteq & \forall \mathsf{description}.\mathsf{Text} \sqcap \exists \mathsf{description} \sqcap (\leq 1\, \mathsf{description})
\end{array}
$$

Encoding of hierarchies

$$
\begin{array}{rcl}
\mathsf{Internal} & \sqsubseteq & \mathsf{Scene} \\
\mathsf{External} & \sqsubseteq & \mathsf{Scene} \\
\mathsf{Scene} & \sqsubseteq & \mathsf{Internal} \sqcup \mathsf{External} \\
\mathsf{Internal} & \sqsubseteq & \neg\mathsf{External}
\end{array}
$$

Encoding of associations

$$
\begin{array}{rcl}
\top & \sqsubseteq & \forall \mathsf{stp\_for\_scn}.\mathsf{Setup} \sqcap \forall \mathsf{stp\_for\_scn}^-.\mathsf{Scene} \\
\mathsf{Scene} & \sqsubseteq & (\geq 1\, \mathsf{stp\_for\_scn}) \\
\mathsf{Setup} & \sqsubseteq & (\geq 1\, \mathsf{stp\_for\_scn}^-) \sqcap (\leq 1\, \mathsf{stp\_for\_scn}^-) \\
\top & \sqsubseteq & \forall \mathsf{tk\_of\_stp}.\mathsf{Take} \sqcap \forall \mathsf{tk\_of\_stp}^-.\mathsf{Setup} \\
\mathsf{Setup} & \sqsubseteq & (\geq 1\, \mathsf{tk\_of\_stp}) \\
\mathsf{Take} & \sqsubseteq & (\geq 1\, \mathsf{tk\_of\_stp}^-) \sqcap (\leq 1\, \mathsf{tk\_of\_stp}^-) \\
\top & \sqsubseteq & \forall \mathsf{located}.\mathsf{Location} \sqcap \forall \mathsf{located}^-.\mathsf{External} \\
\mathsf{External} & \sqsubseteq & (\geq 1\, \mathsf{located}) \sqcap (\leq 1\, \mathsf{located})
\end{array}
$$

What are description logics

Description Logics are **logics** ...

- ... specifically designed to represent knowledge in terms of:
  - objects
  - classes – called concepts in DLs
  - (binary) relations – typically binary relations aka roles in DLs
- ... and to **reason automatically** on such a representation – Thoroughly **studied from the computational point of view**

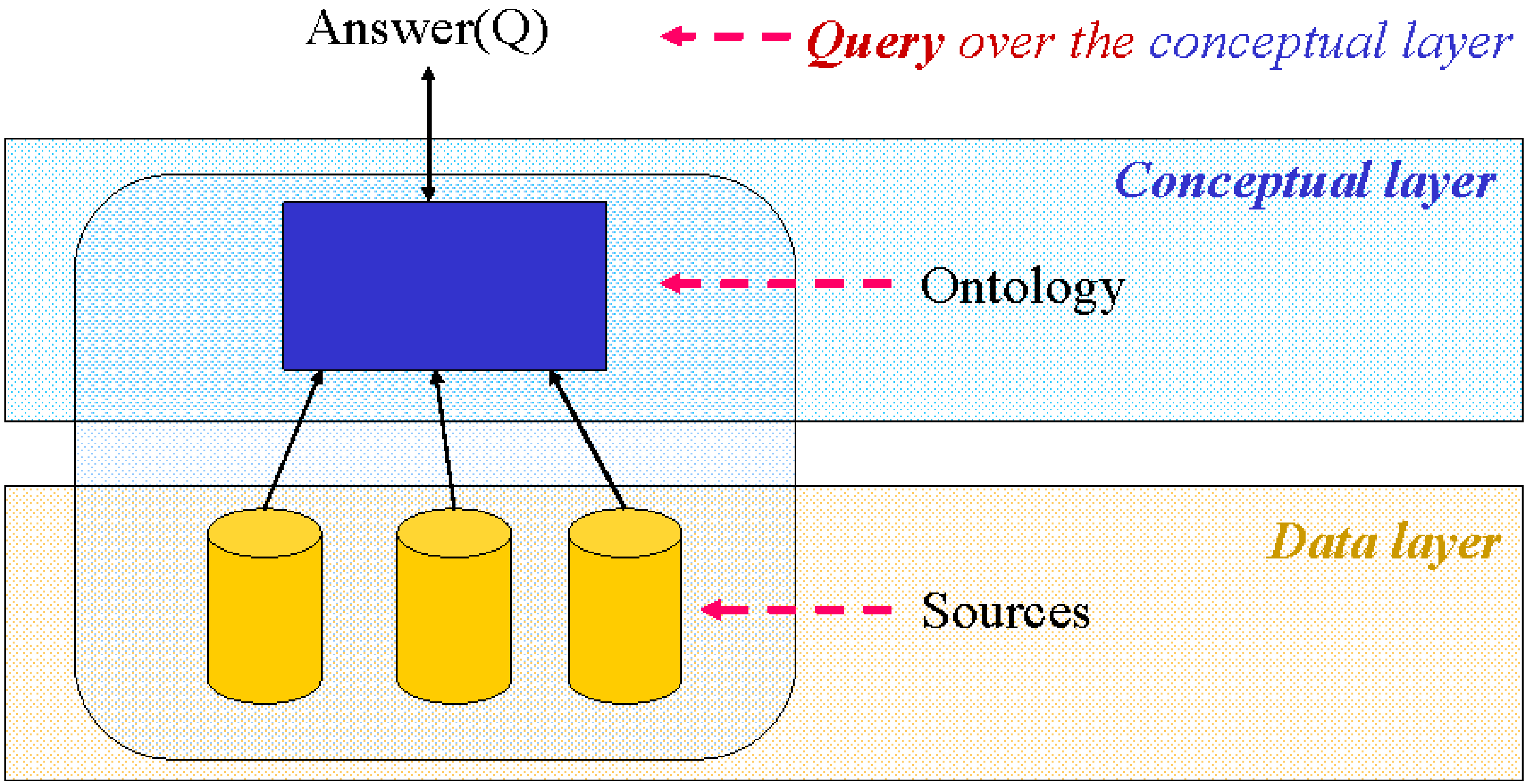
Excellent formal tool for **class-based knowledge representation and reasoning** (*but not for expressing queries!*)

Advocated by the Semantic Web community as "the" formalism for expressing ontologies – W3C OWL

Data integration through Ontologies: outline

- Introduction to data integration through ontologies
- Ontologies: conceptual schema languages, description logics
- Query answering in description logics
- Data complexity tradeoff: a concrete interpretation
- Quonto

Data integration through an DL-based ontology
Query: CQ over ontology; Ontology: expressed in a DL



View-based query answering in (expressive) DL

If we use an expressive description logics such as OWL to express the ontology, is view based query answering (of conjunctive queries) decidable?

YES it can be done in **2EXPTIME** in combined complexity [CDL-AAAI00]!

View-based query answering in full UML class diagrams

If we use UML class diagrams to express the ontology, do we get better bounds?

NO, the only technique known is that of [CDL'AAAI00], hence 2EXPTIME in combined complexity!

Is there any hope of improvement?

NO, not substantial: logical inference (of assertions) and satisfiability of **UML class diagrams is EXPTIME-hard** (and since the can be coded in expressive DLs EXPTIME-complete) [BCD-AIJ05]! Query answering is a service build on top of logical inference so its going to be harder.

Data integration through Ontologies: outline

- Introduction to data integration through ontologies
- Ontologies: conceptual schema languages, description logics
- Query answering in description logics
- Data complexity tradeoff: a concrete interpretation
- Quonto

But what about data complexity?

Slides form:

2005 Description logics Workshop paper:

Data Complexity of Query Answering in Description Logics

Data Complexity of Query Answering in Description Logics

Diego Calvanese[1], *Giuseppe De Giacomo*[2], *Domenico Lembo*[2],
Maurizio Lenzerini[2], *Riccardo Rosati*[2]

[1] **Free University of Bolzano**

[2] **Università di Roma "La Sapienza"**

2005 International Workshop on Description Logics (DL 2005)
Edinburgh, U.K., July 26–28, 2005

Motivations

- Ontologies, often are being used as a conceptual view over data repositories (e.g., in Enterprise Application Integration, Data Integration, Semantic Web)
- DLs are considered the fundamental formal tool for expressing ontologies (e.g., OWL)
- Typical reasoning tasks in DLs are classification, subsumption, instance checking (all based on logical inference)
- When ontologies are used for accessing data, the fundamental task is query answering (still based on logical inference)

The line of research this work belongs to is query answering over ontologies used to access data

Query answering

Considered in several contexts, for example:

- Databases
  - data are completely specified (CWA), and typically large
  - schema not used at run-time (gives only alphabet for queries)
  - queries are complex expressions (e.g., SPJ SQL queries)

$\leadsto$ query answering amounts to query evaluation

- Knowledge bases, e.g., in DLs
  - data (i.e., ABox) are incomplete (but its size is not considered critical)
  - schema (i.e., TBox) is used for query answering (constrains the possible models)
  - queries are atomic (a concept or role name)

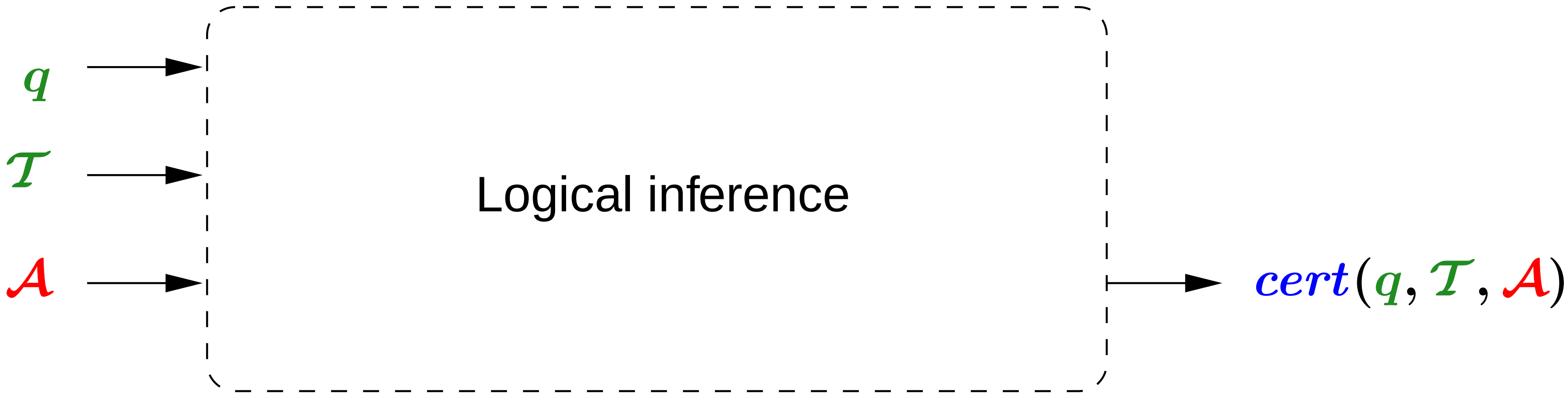
$\leadsto$ query answering amounts to logical inference

Query answering over ontologies

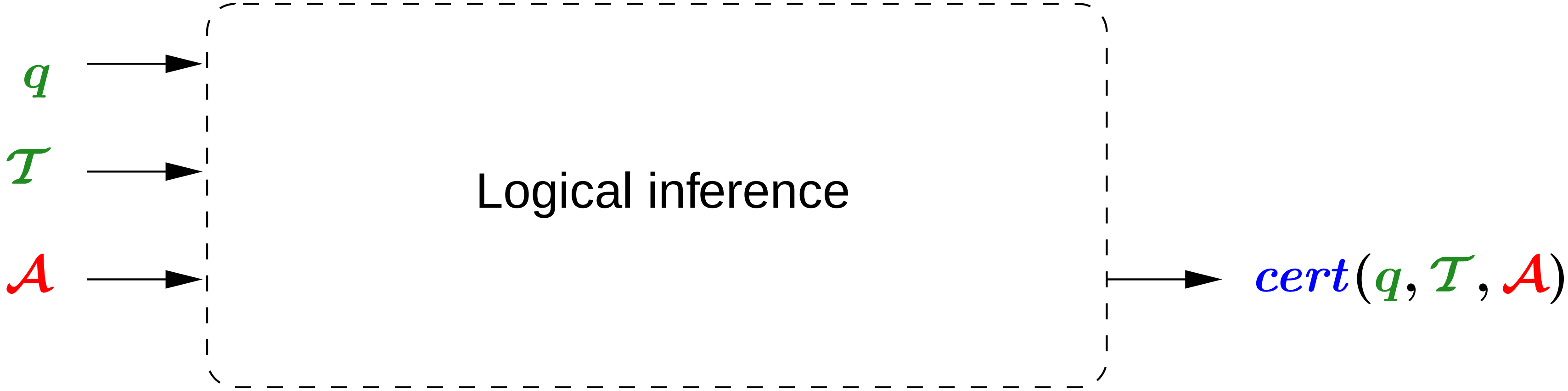
We consider query answering in the following setting:

- data (i.e., ABox $\mathcal{A}$) are incomplete and assumed to be large (their size dominates the size of the schema)
- schema (i.e., TBox $\mathcal{T}$) constrains the possible models
- query q is a complex expressions (conjunctive query)

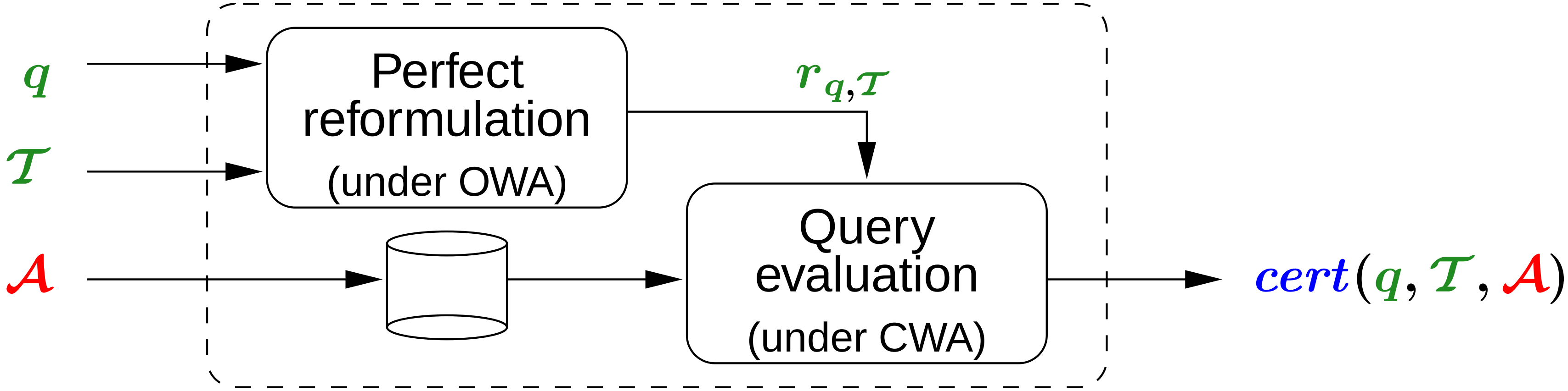
We want to compute $\quad cert(q, \mathcal{T}, \mathcal{A}) = \{\vec{c} \mid \mathcal{T} \cup \mathcal{A} \models q(\vec{c})\}$



Query answering: focus on data



Query answering: focus on data



The critical point in query evaluation is the cost in the size of $\mathcal{A}$ (viewed as a database) $\leadsto$ we have to look at data complexity

Depends on language $\mathcal{L}$ for $r_{q,\mathcal{T}}$, which in turn depends on language for $\mathcal{T}$

Special cases of interest:

- $\mathcal{L}$ is contained in FO (i.e., SQL) $\leadsto$ Query evaluation via a DBMS engine
- $\mathcal{L}$ is NLOGSPACE-hard $\leadsto$ Query evaluation requires linear recursion
- $\mathcal{L}$ is PTIME-hard $\leadsto$ Query evaluation requires recursion (e.g., Datalog)
- $\mathcal{L}$ is coNP-hard $\leadsto$ Query evaluation requires power of Disjunctive Datalog

Previous work on data complexity in DLs

Much of the previous work deals with instance checking (i.e., atomic queries):

[Donini & al. JLC'94] Data and combined complexity for DLs up to $\mathcal{ALC}$

[Hustadt & al. IJCAI'05] Data complexity for very expressive DLs via a reduction to Disjunctive Datalog. Identify also polynomial cases.

Complexity of answering conjunctive queries has been addressed in:

[Levy & Rousset AIJ'98] coNP upper bound for $\mathcal{ALCNR}$ knowledge bases (CARIN setting)

[— & al. AAAI'00] EXPTIME upper bound for $\mathcal{DLR}$ knowledge bases (via reduction to PDL)

[— & al. AAAI'05] Polynomial upper bound for *DL-Lite* knowledge base (using techniques drawn from databases with constraints)

The setting of this work

We have studied data complexity of answering conjunctive queries (CQs) for various DLs containing a subset of the following constructs:

- TBox inclusion assertions: $B \sqsubseteq C$, with:

$$
\begin{array}{lcl}
B & \longrightarrow & A \mid \neg A \mid B_1 \sqcap B_2 \mid \exists R \mid \forall R.A \mid \exists R.A \\
C & \longrightarrow & A \mid \;\bot\; \mid A_1 \sqcup A_2 \mid \exists R \mid \forall R.A \mid \exists R.C \\
R & \longrightarrow & P \mid P^-
\end{array}
$$

- TBox functionality assertions: $(funct\ R)$
- ABox membership assertions: $A(o)$, $P(o_1, o_2)$

  with o, o_1, o_2 constants

Summary of results on data complexity

B	C	R	$(funct\ R)$	Data complexity
$A \mid \exists R \mid B_1 \sqcap B_2$	$A \mid \bot \mid \exists R$	$P \mid P^-$	*allowed*	in LogSpace
$A \mid \exists R \mid B_1 \sqcap B_2$	$A \mid \bot \mid \exists R.C$	$P \mid P^-$	*not allowed*	in LogSpace
$A \mid \exists P.A$	A	P	*not allowed*	NLogSpace-hard
A	$A \mid \forall P.A$	P	*not allowed*	NLogSpace-hard
A	$A \mid \exists P.A$	P	*allowed*	NLogSpace-hard
$A \mid \exists P.A \mid B_1 \sqcap B_2$	A	P	*not allowed*	PTime-hard
$A \mid B_1 \sqcap B_2$	$A \mid \forall P.A$	P	*not allowed*	PTime-hard
$A \mid B_1 \sqcap B_2$	$A \mid \exists P.A$	P	*allowed*	PTime-hard
$A \mid \neg A$	A	P	*not allowed*	coNP-hard
A	$A \mid A_1 \sqcup A_2$	P	*not allowed*	coNP-hard
$A \mid \forall P.A$	A	P	*not allowed*	coNP-hard

Cases in LogSpace

Answering CQs is in LogSpace wrt data complexity for:

$$1.\ \left\{\begin{array}{lcl} B & \longrightarrow & A \mid \exists R \mid B_1 \sqcap B_2 \\ C & \longrightarrow & A \mid \bot \mid \exists R \\ R & \longrightarrow & P \mid P^- \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ allowed}} \end{array}\right.$$

$$2.\ \left\{\begin{array}{lcl} B & \longrightarrow & A \mid \exists R \mid B_1 \sqcap B_2 \\ C & \longrightarrow & A \mid \bot \mid \exists R.C \\ R & \longrightarrow & P \mid P^- \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array}\right.$$

Note: Case 1 extends *DL-Lite* with concept conjunction on the lhs of inclusions

We exploit this result for query answering using DBMS technology:

1. The ABox is stored in a relational database
2. The input CQ q is reformulated as a union $r_{q,\mathcal{T}}$ of CQs
3. $r_{q,\mathcal{T}}$ is evaluated directly over the ABox using DBMS technology

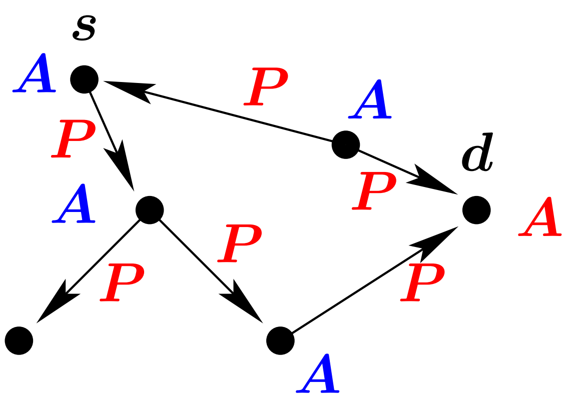
Note: The technique scales up to millions of tuples in the ABox

$\leadsto$ See QuOnto demo

NLogSpace-hard cases

Adding qualified existential on the lhs of inclusions makes instance checking (and hence query answering) NLogSpace-hard:

$$1. \begin{cases} B \;\longrightarrow\; A \mid \exists P.A \\ C \;\longrightarrow\; A \\ R \;\longrightarrow\; P \\ (\mathit{funct}\ R) \text{ not allowed} \end{cases}$$



Hardness proof is by a reduction from reachability in directed graphs:

- TBox $\mathcal{T}$ contains a single inclusion assertion $\exists P.A \sqsubseteq A$
- ABox $\mathcal{A}$ encodes the graph using P and asserts $A(d)$

Result:

$(\mathcal{T}, \mathcal{A}) \models A(s)$ iff d is reachable from s in G

NLogSpace-hard cases

Instance checking (and hence query answering) is NLogSpace-hard in data complexity for:

$$\textbf{1.}\ \left\{\begin{array}{lcl} B & \longrightarrow & A \mid \exists P.A \\ C & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array}\right. \qquad \textbf{2.}\ \left\{\begin{array}{lcl} B & \longrightarrow & A \\ C & \longrightarrow & A \mid \forall P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array}\right.$$

$$\textbf{3.}\ \left\{\begin{array}{lcl} B & \longrightarrow & A \\ C & \longrightarrow & A \mid \exists P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ allowed}} \end{array}\right.$$

1. Reduction from reachability in directed graphs
2. Follows from 1. by replacing $\exists P.A_1 \sqsubseteq A_2$ with $A_1 \sqsubseteq \forall P^-.A_2$
3. Proved by simulating $\exists P.A_1 \sqsubseteq A_2$ via $A_1 \sqsubseteq \exists P^-.A_2$ and $(\mathit{funct}\ P^-)$

PTIME-hard cases

Are obtained from previous cases by adding $B_1 \sqcap B_2$ to lhs of inclusions

Instance checking (and hence query answering) is PTIME-hard in data complexity for:

$$\mathbf{1.}\ \left\{\begin{array}{lcl} B & \longrightarrow & A \mid \exists P.A \mid B_1 \sqcap B_2 \\ C & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array}\right.$$

$$\mathbf{2.}\ \left\{\begin{array}{lcl} B & \longrightarrow & A \mid B_1 \sqcap B_2 \\ C & \longrightarrow & A \mid \forall P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array}\right.$$

$$\mathbf{3.}\ \left\{\begin{array}{lcl} B & \longrightarrow & A \mid B_1 \sqcap B_2 \\ C & \longrightarrow & A \mid \exists P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ allowed}} \end{array}\right.$$

1. Proved via reduction from Path System Accessibility
2. and 3. follow from 1. as in the NLOGSPACE case

Path System Accessibility

Instance of Path System Accessibility: $PS = (N, E, S, t)$ with

- N a set of nodes
- $E \subseteq N \times N \times N$ an accessibility relation
- $S \subseteq N$ a set of source nodes
- $t \in N$ a terminal node

Accessibility of nodes is defined inductively:

- each $n \in S$ is accessible
- if $(n, n_1, n_2) \in E$ and n_1, n_2 are accessible, then also n is accessible

Given PS, checking whether t is accessible, is PTIME-complete

Reduction from Path System Accessibility

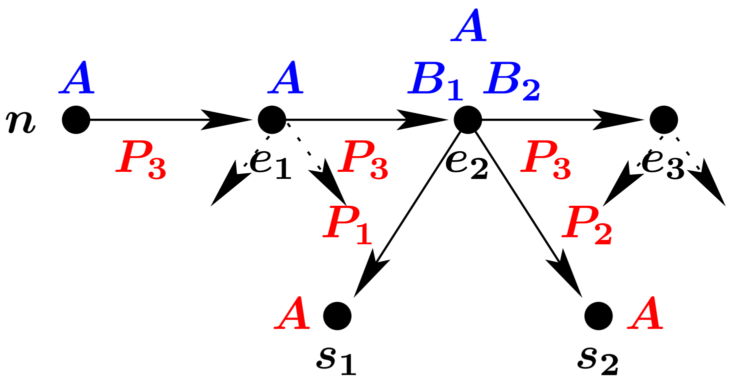
Given an instance $PS = (N, E, S, t)$, we construct

- TBox $\mathcal{T}$ consisting of the inclusion assertions

$$\begin{array}{rclcrcl} \exists P_1.A & \sqsubseteq & B_1 & \qquad & B_1 \sqcap B_2 & \sqsubseteq & A \\ \exists P_2.A & \sqsubseteq & B_2 & & \exists P_3.A & \sqsubseteq & A \end{array}$$

- ABox $\mathcal{A}$ encoding the accessibility relation using P_1, P_2, and P_3, and asserting $A(s)$ for each source node s

$$\begin{aligned} e_1 &= (n, \ .\ , \ .\ ) \\ e_2 &= (n, s_1, s_2) \\ e_3 &= (n, \ .\ , \ .\ ) \end{aligned}$$



Result:

$(\mathcal{T}, \mathcal{A}) \models A(t)$ iff t is accessible in PS

coNP-hard cases

Are obtained when we can use in the query two concepts that cover the whole domain. This forces reasoning by cases on the data

Query answering is coNP-hard in data complexity for:

$$\mathbf{1.}\ \begin{cases} B \;\longrightarrow\; A \mid \neg A \\ C \;\longrightarrow\; A \\ R \;\longrightarrow\; P \\ (\mathit{funct}\ R)\ \text{not allowed} \end{cases}$$

$$\mathbf{2.}\ \begin{cases} B \;\longrightarrow\; A \\ C \;\longrightarrow\; A \mid A_1 \sqcup A_2 \\ R \;\longrightarrow\; P \\ (\mathit{funct}\ R)\ \text{not allowed} \end{cases}$$

$$\mathbf{3.}\ \begin{cases} B \;\longrightarrow\; A \mid \forall P.A \\ C \;\longrightarrow\; A \\ R \;\longrightarrow\; P \\ (\mathit{funct}\ R)\ \text{not allowed} \end{cases}$$

All three cases are proved by adapting the proof of coNP-hardeness of instance checking for $\mathcal{ALE}$ by [Donini & al. JLC 1994]

Conclusions

We have studied the various levels of data complexity for the problem of answering conjunctive queries over a DL knowledge base:

- *DL-Lite* + $\sqcap$ on lhs stays in LogSpace $\leadsto$ relational technology
- with $\exists \boldsymbol{R}.\boldsymbol{A}$ on lhs, we are NLogSpace-hard $\leadsto$ linear recursion needed
- with $\exists \boldsymbol{R}.\boldsymbol{A}$ + $\sqcap$ on lhs, we are PTime-hard $\leadsto$ full recursion needed
- with forms of covering, we are coNP-hard $\leadsto$ Disjunctive Datalog needed

Ongoing work

- Devise tight complexity bounds for the various cases
- Rewriting technique for the cases where recursion is needed
- Data complexity of conjunctive query answering for very expressive DLs. We have now a coNP upper bound for $\mathcal{SHIQ}$ knowledge bases

Data integration through Ontologies: outline

- Introduction to data integration through ontologies
- Ontologies: conceptual schema languages, description logics
- Query answering in description logics
- Data complexity tradeoff: a concrete interpretation
- Quonto

Quonto

- Quonto is a system that performs reasoning, and in particular **query answering over ontologies**.
- It is based on **DL-lite** i.e., the maximal expressive description logic that admits reformulation into FOL (QA is in LOGSPACE).
- It uses variants of the **reformulation techniques** shown in previous lectures by Rosati for dealing with constraints in the relational case.
- Allows for performing sound and complete reasoning (including QA, validation of constraints, etc) over ontologies, and it does this essentially **at the same computational cost of a relational DBMS**

Quonto Demo

Link:

http://............../QUONTOJSP/web/index.jsp