Ontology Mediated Data Access

Giuseppe De Giacomo

Università di Roma “La Sapienza”

Seminari di Ingegneria del Software:
integrazione di dati e servizi
Corso di laurea Specialistica in Ingegneria Informatica
A.A. 2005/06

Outline

- Ontologies & ontologies mediated data access
- Description logics & query answering
- *DL-Lite* & QuOnto
- Reasoning in *DL-Lite*
- Beyond *DL-Lite*: a complexity analysis

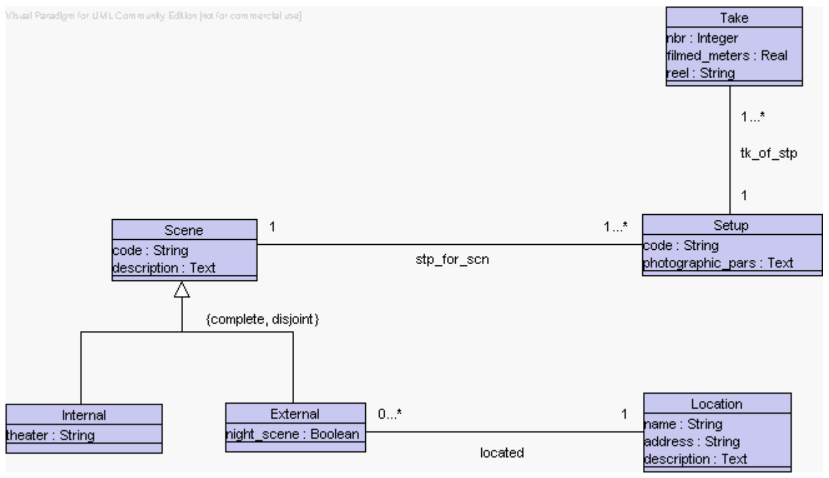
Ontologies in Computer Science

- 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 in Computer Science

- 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

An example of ontology – in UML



An example of ontology – in FOL

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:

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

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

$\forall x, y. (Take(x) \land nbr(x,y)) \supset Integer(y)$
$\forall x, y. (Take(x) \land filmed\_meters(x,y)) \supset Real(y)$
$\forall x, y. (Take(x) \land reel(x,y)) \supset String(y)$

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

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

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

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

$\forall x, y. stp\_for\_scn(x,y) \supset Setup(x) \land Scene(y)$
$\forall x, y. tk\_of\_stp(x,y) \supset Take(x) \land Setup(y)$
$\forall x, y. located(x,y) \supset External(x) \land Location(y)$

$\forall x. Setup(x) \supset 1 \leq \sharp\{y \mid stp\_for\_scn(x,y)\} \leq 1$
$\forall y. Scene(y) \supset 1 \leq \sharp\{x \mid stp\_for\_scn(x,y)\}$
$\forall x. Take(x) \supset 1 \leq \sharp\{y \mid tk\_of\_stp(x,y)\} \leq 1$
$\forall x. Setup(y) \supset 1 \leq \sharp\{x \mid tk\_of\_stp(x,y)\}$
$\forall x. External(x) \supset 1 \leq \sharp\{y \mid located(x,y)\} \leq 1$

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

An example of ontology – in DL

Encoding of classes and attributes

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

Encoding of hierarchies

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

Encoding of associations

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

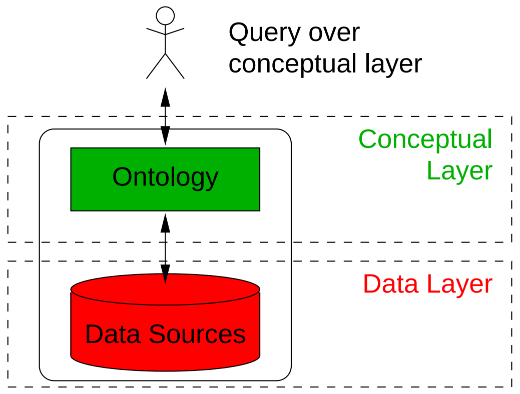
Ontology based data access

Desiderata: achieving logical transparency

- hide to the user where and how data are stored;
- present to the user a conceptual view of the data;
- use a semantically rich formalism for the conceptual view.

Similar to data integration, but with a rich conceptual description as the global view.

Ontology mediated data access



Formal framework for ontology mediated data access

An ontology mediated data access system is a triple $\mathcal{K} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{O}$ is the conceptual view (an ontology) exported to the users
  a logical theory
- $\mathcal{S}$ is a data source schema
  constituted simply by a relational schema (whose alphabet is disjoint from $\mathcal{O}$)
- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{O}$
  different approaches to the specification of mappings

Semantics of an ontology mediated data access system

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

Let $\mathcal{D}$ be a source database over $\mathcal{S}$.

The set of models of $\mathcal{K}$ relative to $\mathcal{D}$ is:

$$sem^{\mathcal{D}}(\mathcal{K}) = \{\ \mathcal{B} \ \mid\ \mathcal{B} \text{ is a model of ontology } \mathcal{O} \text{ and is a model of mapping } \mathcal{M} \text{ wrt } \mathcal{D}\ \}$$

Note: the notion of a model of $\mathcal{M}$ wrt $\mathcal{D}$ depends on the nature of the mapping $\mathcal{M}$.

Semantics of queries to $\mathcal{K}$

If q is a query posed to an ontology mediated data access system $\mathcal{K}$, then the set of certain answers to q wrt $\mathcal{K}$ and $\mathcal{D}$ is

$$cert(q, \mathcal{K}, \mathcal{D}) = \{\vec{c} \in q^{\mathcal{B}} \mid \forall \mathcal{B} \in sem^{\mathcal{D}}(\mathcal{K})\}.$$

Note: query answering is logical implication.

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

The mapping

How is the mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{O}$ specified? We can draw from data integration!!

- Are the sources defined in terms of the ontology?

  Approach called source-centric, or local-as-view, or LAV
- Is the extension of (some of the) concepts in the ontology 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 the rest of the lecture . . .

We will assume that through the mapping we have (virtually) retrieved the data from the sources and have stored them as an incomplete database, i.e. as a set of facts in the same alphabet as the ontology $\mathcal{O}$.

What are description logics

Description Logics are logics . . .

- . . . specifically designed to represent knowledge in terms of:
  - classes – called concepts in DLs
  - relations – typically binary relations aka roles in DLs
- . . . by means of a set of universal axioms, called TBox, and a set of facts, called ABox . . .
- . . . 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

Query answering over ontologies

- Data layer
  - Seen as a DL ABox $\mathcal{A}$ (over atomic concepts and roles only)

    $\leadsto$ Open World Assumption: not all facts are represented explicitly
  - Very large

    $\leadsto$ Stored in a database
- Conceptual layer
  - Represented as a DL TBox $\mathcal{T}$
  - Constrains the possible models
- Conjunctive queries q over the ontology

Query answering amounts to computing certain answers:

$$cert(q, \mathcal{T}, \mathcal{A}) = \{\vec{c} \mid \mathcal{T} \cup \mathcal{A} \models q(\vec{c})\}$$

Conjunctive query answering in (expressive) DL

If we use an expressive description logics such as OWL to express the ontology, is answering conjunctive queries decidable?

YES it can be done in 2EXPTIME in combined complexity [CDL-PODS98, CDL-AAAI00]!

Conjunctive 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 techniques known are 2EXPTIME in combined complexity!

Is there any hope of improvement?

Not substantial: logical inference (of assertions) and satisfiability of UML class diagrams are EXPTIME-hard (and since they can be coded in expressive DLs EXPTIME-complete) [BCD-AIJ05]! Query answering is a service built on top of logical inference so it's going to be harder.

But what about data complexity?

In the above cases is coNP-complete: for hardness see later, for membership [Calvanese-Eiter-Ortiz-2006].

DL-Lite program

Aim:

- Design a DL that is able to express basic ontology constructs (e.g., most of UML class diagrams) . . .
- . . . and where conjunctive query answering is LOGSPACE (as SQL) in the size of the ABox, so as to use the current relational technology for the data layer . . .
- . . . realize an ontology mediated data access system based on such a DL (see the QuOnto demo)

Idea: use query reformulation techniques developed in databases for query containment under inclusion and functional dependencies

$\Rightarrow$ *DL-Lite*

DL-Lite– basic constructs

- Concepts constructs:

$$\begin{array}{lll} Cl & ::= & A \mid \exists P \mid \exists P^- \\ Cr & ::= & Cl \mid \neg Cl \end{array}$$

basic concepts
general concepts

- TBox assertions:

$Cl \sqsubseteq Cr$ inclusion assertions

$(funct\ P)$ $(funct\ P^-)$ functionality assertions

- ABox assertions:

$Cl(a)$ $P(a,b)$ with a, b constants

Semantics of *DL-Lite*

Construct	Syntax	Example	Semantics
atom. conc.	A	$Doctor$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atom. role	P	$child$	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
exist. res.	$\exists P$	$\exists child$	$\{\, d \mid \exists e. (d,e) \in P^{\mathcal{I}} \,\}$
exist. res.	$\exists P^-$	$\exists child^-$	$\{\, e \mid \exists d. (d,e) \in P^{\mathcal{I}} \,\}$
negation	$\neg Cl$	$\neg Doctor$	$\Delta^{\mathcal{I}} \setminus Cl^{\mathcal{I}}$
incl. asser.	$Cl \sqsubseteq Cr$	$Father \sqsubseteq \exists child$	$Cl^{\mathcal{I}} \subseteq Cr^{\mathcal{I}}$
funct. asser.	$(funct\ P)$	$(funct\ succ)$	$\forall d,e,e'.(d,e) \in P^{\mathcal{I}} \wedge (d,e') \in P^{\mathcal{I}} \supset e = e'$
funct. asser.	$(funct\ P^-)$	$(funct\ child^-)$	$\forall e,e',d.(e,d) \in P^{\mathcal{I}} \wedge (e',d) \in P^{\mathcal{I}} \supset e = e'$
mem. asser.	$Cl(a)$	$Father(bob)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
mem. asser.	$P(a,b)$	$child(bob,ann)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$

Note:

- inclusion assertions $\longrightarrow$ inclusion dependencies or disjointness constraints
- functionality assertions $\longrightarrow$ functional dependencies
- membership assertions $\longrightarrow$ tuples on an incomplete database

DL-Lite: observations

- Very basic concept constructs
- Inclusion assertions + functionality assertions
- ABox assertions

Cyclic assertions are allowed.
For example $A \sqsubseteq \exists P,\ \exists R^- \sqsubseteq A$

It does not enjoy the finite model property (in the variant shown here).
For example the TBox $\mathcal{T} = \{A \sqsubseteq \exists P, \exists P^- \sqsubseteq A, B \sqsubseteq \neg A, (funct\ P^-)\}$ and the ABox $\mathcal{A} = \{B(a)\}$ admit only infinite models.

Capturing basic ontology constructs in *DL-Lite*

- ISA between classes

$$A_1 \sqsubseteq A_2$$

- class disjointness

$$A_1 \sqsubseteq \neg A_2$$

- role typing

$$\exists P \sqsubseteq A_1 \qquad \exists P^- \sqsubseteq A_2$$

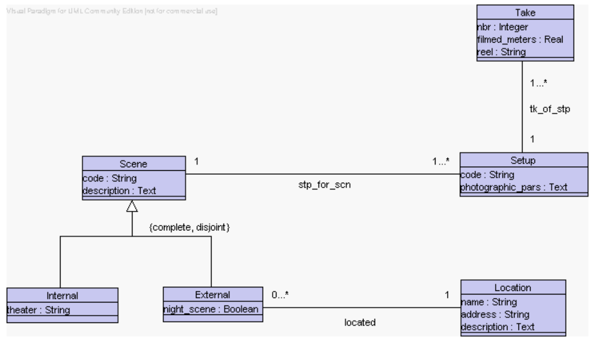
- mandatory participation

$$A_1 \sqsubseteq \exists P \qquad A_2 \sqsubseteq \exists P^-$$

- functionality of roles

$$(funct\ P) \qquad (funct\ P^-)$$

Example: expressing UML Class Diagram in *DL-Lite*



$Internal \sqsubseteq Scene$
$External \sqsubseteq Scene$
$Internal \sqsubseteq \neg External$

$External \sqsubseteq \exists located$
$(funct\ located)$
$\exists located \sqsubseteq External$
$\exists located^- \sqsubseteq Location$

$Setup \sqsubseteq \exists stpForScn$
$Scene \sqsubseteq \exists stpForScn^-$
$(funct\ stpForScn)$
$\exists stpForScn \sqsubseteq Setup$
$\exists stpForScn^- \sqsubseteq Scene$

$Take \sqsubseteq \exists tkOfStp$
$Setup \sqsubseteq \exists tkOfStp^-$
$(funct\ tkOfStp)$
$\exists tkOfStp \sqsubseteq Take$
$\exists tkOfStp^- \sqsubseteq Setup$

$\ldots$

What's missing in *DL-Lite*

Several modeling features are missing in *DL-Lite*, e.g.,:

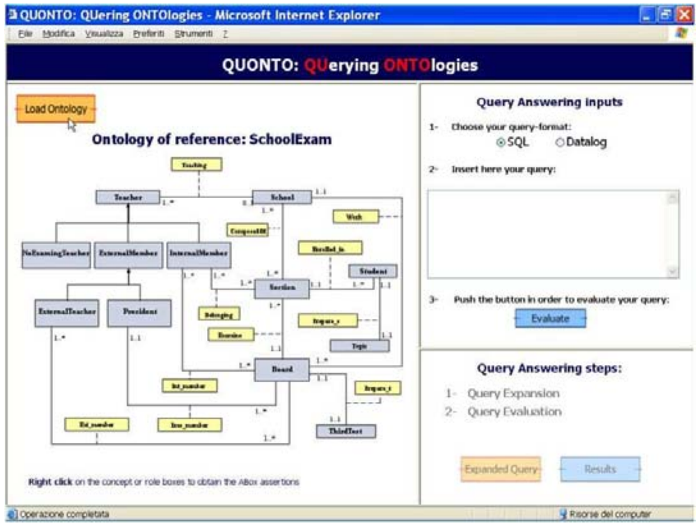
- covering constraints, stating that each instance of a class must be an instance of (at least) one of its subclasses
- subset constraints between associations, stating that the extension of an association (a role) is a subset of the extension of another one

These features are missing exactly to get the nice computational characteristics that we are after – see later.

QuOnto

- Quonto is a system that performs reasoning, and in particular query answering over ontologies.
- It is based on *DL-Lite* (QA is in LOGSPACE – reducible to SQL).
- It uses reformulation techniques originally introduced for dealing with constraints in the relational case – see later – [Johnson&Klug85], [Cali-Lembo-Rosati-PODS03].
- 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


QUONTO: QUering ONTOlogies - Microsoft Internet Explorer

QUONTO: QUerying ONTOlogies

Ontology of reference: SchoolExam

Right click on the concept or role boxes to obtain the ABox assertions

Query Answering inputs

1- Choose your query-format: [✓ SQL] Datalog

2- Insert here your query:

3- Push the button in order to evaluate your query:

Query Answering steps:

1- Query Expansion

2- Query Evaluation

Operazione completata

online at `http://www.dis.uniroma1.it/~quonto`

Ontology mediated data access

- Data layer
  - Seen as a Description Logic ABox $\mathcal{A}$ (over atomic concepts and roles)
    $\rightsquigarrow$ Open World Assumption: not all facts are represented explicitly
  - Very large
    $\rightsquigarrow$ Stored in a database
- Conceptual layer
  - Represented as a Description Logic TBox $\mathcal{T}$
  - Constrains the possible models
- Conjunctive query q is a conjunction of atoms over basic concepts and roles of $\mathcal{T}$

$$q = \{\vec{x} \mid \exists\vec{y}.conj(\vec{x},\vec{y})\}$$

Example: $\{x \mid \exists y.Manager(x) \land Member(x,y) \land \exists Director(x)\}$

Reasoning services

- Knowledge base satisfiability: check whether a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ has a model.
- Query answering amounts to computing certain answers to q wrt $\mathcal{K}$:

$$cert(q,\mathcal{K}) = \{\vec{c} \mid \mathcal{K} \models q(\vec{c})\}$$

i.e., the tuples that are answers to the query in every model of $\mathcal{K}$.

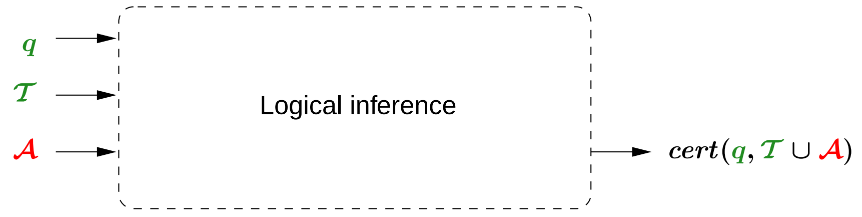
We concentrate on query answering, and specifically on efficiency in the size of the data.

Managing very large data

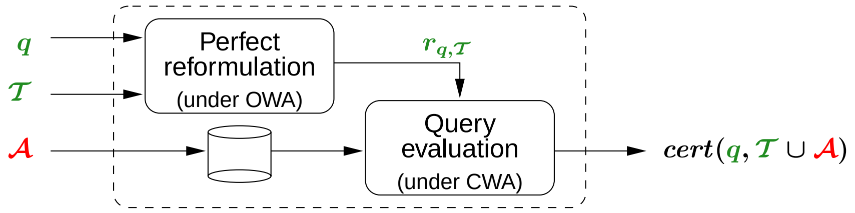
- Best currently available technology: Relational DBMS
- RDBMSs are good at evaluating FOL (i.e., SQL) queries over relational databases
- RDBMSs are specifically optimized for conjunctive queries (considered the most common kinds of queries)

Basic Question: *For which ontology languages (i.e., DLs) can we rephrase query answering over an ontology into query answering over a relational database?*

$\mathcal{Q}$-reducibility



$\mathcal{Q}$-reducibility



q
$\mathcal{T}$
$\mathcal{A}$
Perfect reformulation (under OWA)
$r_{q,\mathcal{T}}$
Query evaluation (under CWA)
$cert(q, \mathcal{T} \cup \mathcal{A})$

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

1. Perfect reformulation: producing the query $r_{q,\mathcal{T}}$, namely the function $cert[q, \mathcal{T}](\cdot)$
2. Query evaluation: evaluating $r_{q,\mathcal{T}}$ over the ABox $\mathcal{A}$ seen as a database, and forgetting about the TBox $\mathcal{T}$ – produces $cert(q, \mathcal{T} \cup \mathcal{A})$

For a query language $\mathcal{Q}$, query answering in a DL is $\mathcal{Q}$-reducible if $r_{q,\mathcal{T}}$ is in $\mathcal{Q}$.
Special case of interest: FOL-reducibility

$\mathcal{Q}$-reducibility and data complexity

$\mathcal{Q}$-reducibility is tightly related to data complexity, i.e., complexity of evaluating $r_{q,\mathcal{T}}$ measured in the size of the ABox $\mathcal{A}$

Special cases of interest:

- $\mathcal{Q}$ is FOL – the DL enjoys FOL-reducibility
  $\leadsto$ Query evaluation via RDBMS
  $\leadsto$ $\mathcal{Q}$ is in LOGSPACE
- $\mathcal{Q}$ is NLOGSPACE-hard $\leadsto$ Query evaluation requires linear recursion
- $\mathcal{Q}$ is PTIME-hard $\leadsto$ Query evaluation requires recursion (e.g., Datalog)
- $\mathcal{Q}$ is coNP-hard $\leadsto$ Query evaluation requires power of Disjunctive Datalog

Previous work on data complexity for DL query answering

Much of the previous work deals with atomic queries only (instance checking in DLs):

[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 reduction to Disjunctive Datalog. Identify also polynomial cases (Horn-$\mathcal{SHIQ}$)

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)

Previous work on query answering under dependencies

Query answering (and query containment) under dependencies has been studied extensively in databases:

[Johnson & Klug JCSS'84] query containment under inclusion dependencies

[Calì & Lembo& Rosati '03] query answering under keys and non-key-conflicting inclusion dependencies

[Fagin & al.'03] recent work on data exchange

DL-Lite family

- Is a family of DLs optimized according to the tradeoff between expressive power and data complexity
- Two maximal languages that enjoy FOL-reducibility: *DL-Lite*$_\mathcal{F}$, *DL-Lite*$_\mathcal{R}$ (we use simply *DL-Lite* to refer to both languages)
- With minimal additions to *DL-Lite*$_\mathcal{F}$ or *DL-Lite*$_\mathcal{R}$, data complexity jumps to NLOGSPACE or above
  $\rightsquigarrow$ We lose FOL-reducibility

Provides an answer to our basic question: *For which DLs can we rephrase query answering over an ontology into query answering over a relational database?*

DL-Lite$_\mathcal{F}$

TBox Language:

- Concept inclusion assertions: $Cl \sqsubseteq Cr$, with:

$$\begin{array}{lcl} Cl & \longrightarrow & A \mid \exists R \mid Cl_1 \sqcap Cl_2 \mid Cl_1 \sqcup Cl_2 \mid \bot \\ Cr & \longrightarrow & A \mid \exists R \mid Cr_1 \sqcap Cr_2 \mid \bot \mid \top \\ R & \longrightarrow & P \mid P^- \end{array}$$

- Functionality assertions: $(funct\ R)$

Observations:

- Captures all the basic constructs of Entity Relationship Diagrams and UML Class Diagrams
- Notable exception: covering constraints in generalizations – if we add them, query answering becomes coNP-hard in data complexity

DL-Lite$_\mathcal{R}$

TBox Language:

- Concept inclusion assertions: $Cl \sqsubseteq Cr$, with:

$$\begin{array}{lcl} Cl & \longrightarrow & A \mid \exists R \mid Cl_1 \sqcap Cl_2 \mid Cl_1 \sqcup Cl_2 \mid \bot \\ Cr & \longrightarrow & A \mid \exists R.Cr \mid Cr_1 \sqcap Cr_2 \mid \bot \mid \top \\ R & \longrightarrow & P \mid P^- \end{array}$$

- Role inclusion assertions: $R_1 \sqsubseteq R_2$

Properties:

- Drops functional restrictions in favor of ISA between roles
- Extends (the DL fragment of) RDFS

Query answering in *DL-Lite*

Given a CQ q and a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$, we compute $cert(q, \mathcal{T} \cup \mathcal{A})$ as follows:

1. Store ABox $\mathcal{A}$ in a relational database
2. Close TBox $\mathcal{T}$ and check for satisfiability wrt $\mathcal{A}$
3. Using $\mathcal{T}$, reformulate CQ q as a union $r_{q,\mathcal{T}}$ of CQs
4. Evaluate $r_{q,\mathcal{T}}$ directly over $\mathcal{A}$ using RDBMS technology

Correctness of this algorithm shows FOL-reducibility of query answering in *DL-Lite*.

$\rightsquigarrow$ Query answering over *DL-Lite* ontologies can be done using RDBMS technology.

$\rightsquigarrow$ Prototype system implemented: QUONTO

Query answering: 1. ABox storage

ABox $\mathcal{A}$ stored as a relational database in a standard DBMS as follows:

- Expand ABox by closing it under the following rule:
  - for each $R(a,b)$ in ABox, add also $\exists R(a)$ and $\exists R^-(b)$
- For each basic concept B used in ABox:
  - define a unary relational table $\mathtt{tab}_B$
  - populate $\mathtt{tab}_B$ with each $\langle a\rangle$ such that $B(a)$ is in ABox
- For each role R used in ABox,
  - define a binary relational table $\mathtt{tab}_R$
  - populate $\mathtt{tab}_R$ with each $\langle a,b\rangle$ such that $R(a,b)$ is in ABox

Query answering: 2. KB satisfiability

To check that $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ is satisfiable (this works for *DL-Lite*$_\mathcal{F}$, slightly more complicated for *DL-Lite*$_\mathcal{R}$):

1. Close the TBox $\mathcal{T}$ by computing all disjointness assertions that are implied according to the rule:
   - if $Cl_1 \sqsubseteq B$ and $B \sqcap Cl_2 \sqsubseteq \bot$, then add $Cl_1 \sqcap Cl_2 \sqsubseteq \bot$.
2. Verify that the ABox $\mathcal{A}$ does not explicitly violate any disjointness or functionality assertion of the closed TBox.

   This can be done by issuing suitable conjunctive queries over the database tables storing $\mathcal{A}$, e.g.:
   - $\mathcal{A}$ violates $A_1 \sqcap A_2 \sqsubseteq \bot$ iff $q(\mathcal{A}) \neq \emptyset$, where
     $q = \{\langle\rangle \mid A_1(x), A_2(x)\}$
   - $\mathcal{A}$ violates $(\mathit{funct}\ P)$ iff $q(\mathcal{A}) \neq \emptyset$, where
     $q = \{\langle\rangle \mid P(x,y), P(x,z), y \neq z\}$

Query answering: 3. Query reformulation

Reformulate the CQ q into a set of queries: apply to q in all possible ways the inclusion assertions in the TBox:

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

($\_$ denotes an unbound variable, i.e., a variable that appears only once)

This corresponds to exploiting ISAs, role typing, and mandatory participation to obtain new queries that could contribute to the answer.

Query answering: 3. Query reformulation (cont'd)

After each reformulation step, try to perform unification:

1. check for atoms g_1, g_2 that unify, and
2. apply to the query the most general unifier between g_1 and g_2

Unification may make variables unbound.

Note: disjointness assertions and functionality assertions can be ignored during reformulation!

Query answering: 4. Evaluation of reformulated query

The resulting union of CQs is evaluated over the ABox stored as relational database.

Query answering in *DL-Lite* – Observations

Our technique is based on rewriting (i.e., inverse chase), rather than chasing the database.

What if we wanted to chase the database?

- We are in a case where the chase would be infinite in general (no weakly acyclic tgds).
- Note that *DL-Lite*$_{\mathcal{F}}$ does not even have the finite model property.
- Could we find a bound on the size of the chase that guarantees correctness of query answering?
  - No! For any bound we fix for the chase, can give a query that, when evaluated on the chase does not provide the certain answers.
  - We could find a bound that depends on the size of the query.

DL-Lite: complexity results

- KB satisfiability is
  - polynomial in the size of TBox and of ABox
- Query answering is
  - exponential in the size of the query (NP-complete)
  - polynomial in the size of TBox and of ABox (in fact LOGSPACE in the ABox)

Can we further extend these results to more expressive ontology languages / DLs?

Summary of results on data complexity

	Cl	Cr	$\mathcal{F}$	$\mathcal{R}$	Data complexity of query answering
1	*DL-Lite*$_{\mathcal{F}}$		√	–	in LOGSPACE
2	*DL-Lite*$_{\mathcal{R}}$		–	√	in LOGSPACE
3	*DLR-Lite*$_{\mathcal{F}}$		√	–	in LOGSPACE
4	*DLR-Lite*$_{\mathcal{R}}$		–	√	in LOGSPACE
5	$A \mid \exists P.A$	A	–	–	NLOGSPACE-hard
6	A	$A \mid \forall P.A$	–	–	NLOGSPACE-hard
7	A	$A \mid \exists P.A$	√	–	NLOGSPACE-hard
8	$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTIME-hard
9	$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTIME-hard
10	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	√	–	PTIME-hard
11	$A \mid \exists P.A \mid \exists P^-.A$	$A \mid \exists P$	–	–	PTIME-hard
12	A	$A \mid \exists P.A \mid \exists P^-.A$	√	–	PTIME-hard
13	$A \mid \exists P.A$	$A \mid \exists P.A$	√	–	PTIME-hard
14	$A \mid \neg A$	A	–	–	coNP-hard
15	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-hard
16	$A \mid \forall P.A$	A	–	–	coNP-hard

All NLOGSPACE and PTIME hardness results hold already for atomic queries

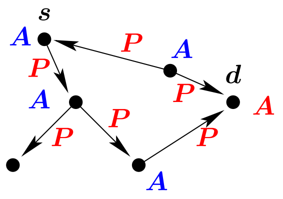
Observations

- Results on FOL-reducibility can be extended to n-ary relations $\leadsto$ *DLR-Lite*$_{\mathcal{F}}$ and *DLR-Lite*$_{\mathcal{R}}$
- RDFS is a subset of *DL-Lite*$_{\mathcal{R}}$ $\leadsto$ enjoys FOL-reducibility
- Horn-$\mathcal{SHIQ}$ [Hustadt & al. IJCAI'05] is PTime-hard even for instance checking (line 13) $\leadsto$ does not enjoy FOL-reducibility
- DLP [Grosof & al. WWW'03] is PTime-hard (line 8) $\leadsto$ does not enjoy FOL-reducibility
- Although used in ER and UML, no hope of including covering constraints, since we get coNP-hardness for trivial DLs (line 15)

NLogSpace-hard cases

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

$$5 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \mid \exists P.A \\ Cr & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is not allowed}} \end{array} \right.$$



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:

$$5 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \mid \exists P.A \\ Cr & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is not allowed}} \end{array} \right. \qquad 6 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \\ Cr & \longrightarrow & A \mid \forall P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is not allowed}} \end{array} \right.$$

$$7 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \\ Cr & \longrightarrow & A \mid \exists P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is allowed}} \end{array} \right.$$

5: reduction from reachability in directed graphs
6: follows from 5 by replacing $\exists P.A_1 \sqsubseteq A_2$ with $A_1 \sqsubseteq \forall P^-.A_2$
7: proved by simulating $\exists P.A_1 \sqsubseteq A_2$ via $A_1 \sqsubseteq \exists P^-.A_2$ and $(funct\ P^-)$

PTime-hard cases

Are obtained from previous cases by adding $A_1 \sqcap A_2$ to lhs of inclusions

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

$$8 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \mid \exists P.A \mid A_1 \sqcap A_2 \\ Cr & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is not allowed}} \end{array} \right. \qquad 9 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \mid A_1 \sqcap A_2 \\ Cr & \longrightarrow & A \mid \forall P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is not allowed}} \end{array} \right.$$

$$10 \left\{ \begin{array}{lcl} Cl & \longrightarrow & A \mid A_1 \sqcap A_2 \\ Cr & \longrightarrow & A \mid \exists P.A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(funct\ R) \text{ is allowed}} \end{array} \right.$$

8: proved via reduction from Path System Accessibility
9 and 10 follow from 8 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

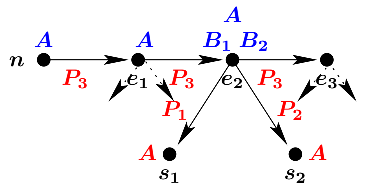
Given an instance $PS = (N, E, S, t)$, we construct

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

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

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

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



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:

$$14 \left\{ \begin{array}{lcl} B & \longrightarrow & A \mid \neg A \\ C & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array} \right. \qquad 15 \left\{ \begin{array}{lcl} B & \longrightarrow & A \\ C & \longrightarrow & A \mid A_1 \sqcup A_2 \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array} \right.$$

$$16 \left\{ \begin{array}{lcl} B & \longrightarrow & A \mid \forall P.A \\ C & \longrightarrow & A \\ R & \longrightarrow & P \\ \multicolumn{3}{l}{(\mathit{funct}\ R) \text{ not allowed}} \end{array} \right.$$

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

2+2-SAT

2+2-SAT: satisfiability of a 2+2-CNF formula, i.e., a CNF formula where each clause has exactly 2 positive and 2 negative literals

Example: $\varphi = c_1 \wedge c_2 \wedge c_3$, with

$$\begin{aligned} c_1 &= \ell_1 \vee \ell_2 \vee \neg\ell_3 \vee \neg\ell_4 \\ c_2 &= \mathit{false} \vee \mathit{false} \vee \neg\ell_1 \vee \neg\ell_4 \\ c_3 &= \mathit{false} \vee \ell_4 \vee \neg\mathit{true} \vee \neg\ell_2 \end{aligned}$$

2+2-SAT is NP-complete [Donini &al. JLC 1994]

Reduction from 2+2-SAT

2+2-CNF formula $\varphi = c_1 \wedge \cdots \wedge c_k$ over letters $\ell_1, \ldots, \ell_n, \mathit{true}, \mathit{false}$

- ABox $\mathcal{A}_\varphi$ constructed from φ (concepts L, T, F, roles P_1, P_2, N_1, N_2):
  - for each letter ℓ_i: $L(\ell_i)$
  - for each clause $c = \ell_1 \vee \ell_2 \vee \neg\ell_3 \vee \neg\ell_4$:
    $P_1(c, \ell_1)$, $P_2(c, \ell_2)$, $N_1(c, \ell_3)$, $N_2(c, \ell_4)$
  - $T(\mathit{true})$, $F(\mathit{false})$
- TBox $\mathcal{T} = \{\, L \sqsubseteq T \sqcup F \,\}$
- $$Q = \{\, \langle\rangle \mid \exists c, \ell_1, \ell_2, \ell_3, \ell_4.\, P_1(c,\ell_1), P_2(c,\ell_2), N_1(c,\ell_3), N_2(c,\ell_4), F(\ell_1), F(\ell_2), T(\ell_3), T(\ell_4) \,\}$$

We have: $\mathcal{T} \cup \mathcal{A}_\varphi \models Q$ iff φ is not satisfiable.

Intuition: each model of $\mathcal{T}$ partitions L into T and F, and corresponds to a truth assignment to $\ell_1, \ldots, \ell_n$. Q asks for a false clause.

What to bring home?

- Ontologies based data access is an important problem we have to consider
- Expressive power of ontology language heavily influences complexity of query answering
- Good news: reasonable expressiveness in the ontology and efficiency of query answering can be reconciled $\leadsto$ *DL-Lite*

Other issues

- Extensions of *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$ with additional constructs
- Tight complexity bounds for the various cases (missing upper bounds)
- Rewriting technique for the cases where recursion is needed
- Data complexity of conjunctive query answering for very expressive DLs
- We need to address the issue of updates through an ontology
- What if we want to restrict the attention to finite models only?
- Address the values vs. objects mismatch, in the mapping between the conceptual and the data layer