Description Logics for Integration

Y. Angélica Ibáñez-García

KRDB Research Centre,
Faculty of Computer Science
Free University of Bozen-Bolzano

DEIS 2010
8-12 Nov.

Outline

Outline

Ontology-based Data Integration Framework

OB Data integration:

- unified and transparent access,
- global (or target) schema
- collection of data stored in multiple, autonomous, and heterogeneous data sources

More formally:

$$\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$$

where

- $\mathcal{G}$: *global schema*: viewed as a conceptual schema, expressed in logic (ontology)
- $\mathcal{S}$: *data sources*: wrapped as relational databases
- $\mathcal{M}$: *mappings*: semantically link data at the sources ($\mathcal{S}$) with the ontology ($\mathcal{G}$)

Problems in OB Data Integration

- How to model the global schema:
  - provide a description of the data of interest in semantic terms,
  - represent the global view as a conceptual schema;
  - formalize it as logical theory (ontology)
  - use the resulting logical theory for reasoning, (e.g. query answering)
- How to model the the sources, and the mappings
- How to answer queries expressed on the global schema

Outline

Description Logics in a Nutshell

- Logics specifically designed to represent and reason on structured knowledge:
  - Concepts: sets of objects
  - Roles: binary relations between (instances of) concepts
- Knowledge Bases, aka Ontologies
  - Intentional Knowledge: TBoxes, general properties of concepts
  - Extensional Knowledge: ABoxes, assertions about individuals/objects
- Nice computational properties: decidability, tractability (in some cases)
- Trade-off between expressive power and computational complexity of reasoning

Current applications of Description Logics

DLs have evolved from being used "just" in KR.

Novel applications of DLs:

- Databases:
  - schema design, schema evolution
  - query optimization
  - integration of heterogeneous data sources, data warehousing
- Conceptual modeling
- Foundation for the Semantic Web (variants of OWL correspond to specific DLs) ...

Reasoning over an Ontology

Reasoning Services

- **Ontology Satisfiability:** $\mathcal{O}$ admits at least one model.
- **Concept Instance Checking:** c is an instance of a concept C in every model of $\mathcal{O}$.
- **Role Instance Checking:** a pair (a_1, a_2) of individuals is an instance of a role R in every model of $\mathcal{O}$.
- **Query Answering:** computing the certain answers to a query over $\mathcal{O}$.

Query answering on Ontologies

- An ontology imposes constraints on the data.
- Actual data may be incomplete or inconsistent w.r.t. such constraints.

- To be able to deal with data efficiently: separate the contribution of $\mathcal{A}$ from the contribution of q and $\mathcal{T}$.
  - $\rightsquigarrow$ Query answering by query rewriting
  - $\rightsquigarrow$ Query answering by data completion

Queries over ontologies

A Conjunctive Query (CQ) over an Ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$

has the form:

$$q(\vec{x}) \leftarrow conj(\vec{x}, \vec{y})$$

where $\vec{x}$ denotes the distinguished variables, $\vec{y}$ the non-distinguished variables, $conj(\vec{x}, \vec{y})$ is a conjunction of atoms

The predicates in atoms are concepts and roles of the ontology.

Union of Conjunctive queries (UCQ)

Datalog notation

$$Q(\vec{x}) \leftarrow conj_1(\vec{x}, \vec{y}_1)$$
$$\ldots\ldots$$
$$Q(\vec{x}) \leftarrow conj_n(\vec{x}, \vec{y}_n)$$

Semantics of Queries

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an interpretation of $\mathcal{O}$, and $q(\vec{x}) \leftarrow \varphi(\vec{x}, \vec{y})$ a CQ.

An answer to $q(\vec{x}) \leftarrow \varphi(\vec{x}, \vec{y})$ over $\mathcal{I}$, denoted $q^{\mathcal{I}}$

is the set of tuples $\vec{c}$ of constants of $\mathcal{A}$ such that there exists a tuple $\vec{o} \in \Delta^{\mathcal{I}} \times \ldots \times \Delta^{\mathcal{I}}$; and the formula $\varphi(\vec{c}, \vec{y})$ evaluates to true in $\mathcal{I}_{[\vec{y}/\vec{o}]}$,

The certain answers to $q(\vec{x})$ over $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, denoted $cert(q, \mathcal{O})$

are the tuples $\vec{c}$ of constants of $\mathcal{A}$ such that $\vec{c}$ is an answer of q ($\vec{c} \in q^{\mathcal{I}}$) in every model $\mathcal{I}$ of $\mathcal{O}$

Tractable Description Logics

- *DL-Lite*:
  - family of DLs optimized according to the tradeoff between expressive power and complexity of query answering, with emphasis on data
  - Nice computational properties for answering UCQs
    - ⋆ same data complexity as relational databases
    - ⋆ query answering can be delegated to a relational DB engine
  - Captures conceptual modeling formalism
  - Is at the basis of the OWL2 QL profile of OWL2
- $\mathcal{EL}$:
  - is particularly suitable for applications employing ontologies that define very large numbers of classes and/or properties
  - ontology consistency, class expression subsumption, and instance checking can be decided in polynomial time
  - e.g. very large biomedical ontology SNOMED CT (≈ 400.000axioms)

DL-Lite$_{\mathcal{A}}$ Syntax

- **Concept expressions:**

$$\begin{array}{lcl} B & ::= & A \mid \exists Q \mid \delta(U_C) \\ C & ::= & \top_C \mid B \mid \neg B \mid \exists Q.C \end{array}$$

- **Value-domain expressions:**

$$\begin{array}{lcl} E & ::= & \rho(U_C) \\ F & ::= & \top_D \mid T_1 \mid \cdots \mid T_n \end{array}$$

- **Role expression:**

$$\begin{array}{lcl} Q & ::= & P \mid P^- \\ R & ::= & Q \mid \neg Q \end{array}$$

- **Attribute expressions:**

$$V_C ::= U_C \mid \neg U_C$$

Semantics of *DL-Lite*$_{\mathcal{A}}$: objects vs. values

Definition (An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$)

	Objects	Values
Domain: $\Delta^{\mathcal{I}}$	$\Delta_O^{\mathcal{I}}$	$\Delta_V^{\mathcal{I}}$
Constants: Γ	$c \in \Gamma_O$, $c^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$	$d \in \Gamma_V$, $d^{\mathcal{I}} \in \Delta_V^{\mathcal{I}}$
Concepts /Types	Concept C, $C^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}}$	RDF datatype T_i, $T_i^{\mathcal{I}} \subseteq \Delta_V^{\mathcal{I}}$
Roles/ Attributes	Role R, $R^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}$	Attribute V, $V^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$

Semantics of *DL-Lite*$_{\mathcal{A}}$ constructs

Construct	Syntax	Semantics
top concept	$\top_C$	$\Delta_O^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential restriction	$\exists Q$	$\{o \mid \exists o' \mid (o, o') \in Q^{\mathcal{I}}\}$
attribute domain	$\delta(U)$	$\{o \mid \exists v.\, (o, v) \in U^{\mathcal{I}}\}$
inverse role	P^-	$\{(b, a) \mid (a, b) \in P^{\mathcal{I}}\}$
role negation	$\neg Q$	$(\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$
top domain	$\top_D$	$\Delta_V^{\mathcal{I}}$
attribute range	$\rho(U)$	$\{v \mid \exists o.\, (o, v) \in U^{\mathcal{I}}\}$
attribute negation	$\neg U$	$(\Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus U^{\mathcal{I}}$

DL-Lite$_\mathcal{A}$ Ontologies

TBox $\mathcal{T}$

$B \sqsubseteq C$	concept inclusion	$E \sqsubseteq F$	value-domain inclusion
$Q \sqsubseteq R$	role inclusion	$U_C \sqsubseteq V_C$	attribute inclusion
(funct Q)	role functionality	(funct U_C)	attribute functionality
(id $BI_1, \ldots, I_n$)	identification constraints		

where each I_i is a role name, an inverse role or an attribute

—

No functional or identifying role or attribute can be specialized by using it in the right-hand side of a role or attribute inclusion assertion.

ABox $\mathcal{A}$

$$A(a),\ P(a,b),\ U_C(a,d)$$

where a, b are object constants, and d is a value constant

Semantics of *DL-Lite*$_{\mathcal{A}}$ assertions

Syntax	Semantics
$B \sqsubseteq C$	$B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
$Q \sqsubseteq R$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
$E \sqsubseteq F$	$E^{\mathcal{I}} \subseteq F^{\mathcal{I}}$
$U \sqsubseteq V$	$U^{\mathcal{I}} \subseteq V^{\mathcal{I}}$
(funct Q)	$\forall o, o_1, o_2.\, (o, o_1) \in Q^{\mathcal{I}} \wedge (o, o_2) \in Q^{\mathcal{I}} \rightarrow o_1 = o_2$
(funct U)	$\forall o, v_1, v_2.\, (o, v_1) \in U^{\mathcal{I}} \wedge (o, v_2) \in U^{\mathcal{I}} \rightarrow v_1 = v_2$
(id B $I_1, \ldots, I_n$)	$I_1, \ldots, I_n$ identify instances of B
$A(c)$	$c^{\mathcal{I}} \in A^{\mathcal{I}}$
$P(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$
$U(c, d)$	$(c^{\mathcal{I}}, val(d)) \in U^{\mathcal{I}}$

Query answering in *DL-Lite*$_{\mathcal{A}}$

- Based on query reformulation
- Given a (U)CQ $q(x)$, and a satisfiable ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$,
- rewrite $q(x)$ into an FO query $q^{\mathcal{T}}(x)$ (independently of $\mathcal{A}$) such that

$$\text{for all } \vec{a}, \langle \mathcal{T}, \mathcal{A} \rangle \models q[\vec{a}] \text{ iff } \mathcal{A} \models q^{\mathcal{T}}[\vec{a}]$$

- evaluate the query $q^{\mathcal{T}}$ over $\mathcal{A}$, seen as a complete DB

\+ Off-the-shelf RDBMSs can be used for evaluating $q^{\mathcal{T}}$

\- rewritten queries can be of size $(\mid \mathcal{T} \mid \cdot \mid q \mid)^{|q|}$

\- not scalable when $\mid \mathcal{T} \mid$ is large (even if $\mid q \mid$) is relatively small

\- This rewriting approach is not applicable to other tractable DLs, e.g. $\mathcal{EL}$

Perfect rewriting in *DL-Lite*$_{\mathcal{A}}$

To compute the perfect rewriting, starting from the original (U)CQ:
Iteratively get a CQ to be processed and either:

- expand positive inclusions & simplify redundant atoms, or
- unify atoms in the CQ to obtain a more specific CQ to be further expanded.

Each result of the above steps is added to the queries to be processed, until no further CQ can be added.

—

Note: negative inclusions, functionalities, and identification constraints play a role in ontology satisfiability, but not in query answering (i.e., we have separability)

- Use the PIs as basic rewriting rules:
  - when an atom in the query unifies with the head of the rule, substitute the atom with the body of the rule.
- Apply in all possible ways unification between atoms in a query. Unifying atoms can make rules applicable that were not so before, and is required for completeness of the method.

Algorithm PerfectRef(Q, $\mathcal{T}_P$)

Input: union of conjunctive queries Q, set of *DL-Lite*$_\mathcal{A}$ PIs $\mathcal{T}_P$

Output: union of conjunctive queries PR $PR := Q$;

repeat

$PR' := PR$;

for each $q \in PR'$ **do**

for each g in q **do for each** PI I in $\mathcal{T}_P$ **do**

if I is applicable to g **then** $PR := PR \cup \{ApplyPI(q, g, I)\}$;

for each g_1, g_2 in q **do**

if g_1 and g_2 unify **then** $PR := PR \cup \{\tau(Reduce(q, g_1, g_2))\}$;

until $PR' = PR$;

return PR

$DL\text{-}Lite_{\mathcal{A}}$ TBox

Example

$$\begin{array}{ll} manager \sqsubseteq employee & manager(x) \rightarrow employee(x) \\ employee \sqsubseteq person & employee(x) \rightarrow person(x) \\ employee \sqsubseteq \exists WORKS\text{-}FOR & employee(x) \rightarrow WORKS\text{-}FOR(x, \_) \\ \exists WORKS\text{-}FOR^{-} \sqsubseteq project & WORKS\text{-}FOR(\_, y) \rightarrow project(y) \end{array}$$

Query:

$$q(x) \leftarrow WORKS\text{-}FOR(x, y), project(y)$$

Perfect Reformulation:

$$\begin{array}{lcl} q(x) & \leftarrow & WORKS\text{-}FOR(x, y), project(y) \\ q(x) & \leftarrow & WORKS\text{-}FOR(x, y), WORKS\text{-}FOR(\_, y) \\ q(x) & \leftarrow & WORKS\text{-}FOR(x, \_) \\ q(x) & \leftarrow & employee(x) \\ q(x) & \leftarrow & manager(x) \end{array}$$

Complexity of reasoning in *DL-Lite*$_{\mathcal{A}}$

	ABox + TBox	**data complexity**	**TBox + query**
Ontology satisfiability	PTIME	AC^0	
Query answering for CQs and UCQs	PTIME	AC^0	NP-**complete**

this is exactly as in relational DBs.

In fact, reasoning (e.g. ontology satisfiability) can be done by constructing suitable FOL/SQL queries and evaluating them over the ABox: FOL-rewritability.

The $\mathcal{EL}$ family

	construct	syntax	semantics
concepts	top	$\top$	$\Delta^{\mathcal{I}}$
	bottom	$\bot$	$\emptyset$
	atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
	qualified existential restriction	$\exists P.C$	$\{o \mid \exists o'.(o,o') \in P^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\}$
	conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
roles	atomic role	P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
TBox	concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
ABox	membership assertions	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
		$P(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$

Data Completion / Combined approach

- Extend ABox to the canonical model of $(\mathcal{T}, \mathcal{A})$, $\mathcal{I}_\mathcal{K}$
- Encode it as a finite structure, $\mathcal{C}_\mathcal{K}$
- Rewrite q into $q^\dagger$ to ensure that the answers to q over $\mathcal{C}_\mathcal{K}$ are correct
- $\mathcal{C}_\mathcal{K}$ can be constructed by first-order queries:
- Avoid exponential blow up: polynomial rewritings for $\mathit{DL\text{-}Lite}^{\mathcal{N}}_{horn}$
- Applicable to other DLs of the *DL-Lite* family, exponential rewriting
- Needs access to the data

Query rewriting for $\mathcal{EL}$

Rewrite a given CQ $q(\vec{x}) \leftarrow \varphi(\vec{x}, \vec{y})$ into an FO query $q^\dagger$ such that

- the answers to q over $\mathcal{I}_\mathcal{K}$ are the same as the answers to $q^\dagger$ over $\mathcal{C}_\mathcal{K}$
- $\mid q^\dagger \mid = O(\mid q \mid \cdot \mid T \mid)$

$$q^\dagger(\vec{x}) \leftarrow \varphi \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3$$

φ_1: answer variables and variables in cycles in q must be mapped to ABox

φ_2: if $R(x_1, x_2), R(x_3, x_2)$ in q and x_2 is mapped outside the ABox then $x1 = x3$

φ_3: if $R(x_1, x_2), S(x_3, x_2)$ in q and $R \neq S$ then x_2 must be mapped to ABox

Query rewriting, open questions

- is the exponential blowup unavoidable for role inclusions?
- is the exponential blowup unavoidable for positive existential queries?
- for which DLs pure rewriting can be polynomial?
- Alternative query rewriting techniques based on resolution for more expressive logics (with recursive rewritings) [Pérez-Urbina et al., 2010].

Outline

Ontology-based data integration Systems

Ontology-based data integration System

is a triple $\mathcal{O}\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ where:

- $\mathcal{T}$ is a TBox
- $\mathcal{S}$ is a relational database representing the sources
- $\mathcal{M}$ is a set of mapping assertions between $\mathcal{T}$ and $\mathcal{S}$

The mapping assertions are a crucial part of an Ontology-Based Data Integration System:
they are used to extract the data from the sources to "populate" the ontology
$\rightsquigarrow$ virtual ABox

Ontology-based data integration: the *DL-Lite*$_\mathcal{A}$ solution

- the data sources are assumed to be wrapped and presented as relational sources.
- data federation tools such as IBM Information Integrator can be used to integrate the sources into a single relational
- Use *DL-Lite*$_\mathcal{A}$ ontologies (with mappings) for the conceptual view on the data.
- Exploit effectiveness of query answering,
- Take advantage of the distinction between objects and values in *DL-Lite*$_\mathcal{A}$ to deal with the notorious impedance mismatch problem.

Impedance mismatch problem

- In RDBs, information is represented in forms of tuples of values
- Ontologies, use both objects and values

- Use an alphabet Λ of function symbols, each with an associated arity.
- Values are denoted by constants from an alphabet Γ_V
- Instances of concepts are denoted by terms built out of Γ_V

$$f(d_1, \ldots, d_n), \text{ with } f \in \Lambda, \text{ and } d_i \in \Gamma_V$$

Example

If a person is identified by her SSN, we can introduce a function symbol **pers**/1. If IBN81B24 is a SSN, then **pers**(IBN81B24) denotes a person.

Mappings

A mapping assertion in $\mathcal{M}$ has the form:

$$\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{t}, \vec{y})$$

where

- Φ is am arbitrary SQL query of arity $n > 0$ over $\mathcal{S}$,
- Ψ is a conjunctive query over $\mathcal{T}$ of arity $n' > 0$ without non-distinguished variables
- $\vec{x}, \vec{y}$ are variables with $\vec{y} \subseteq \vec{x}$,
- $\vec{t}$ are terms of the form $f(\vec{z})$, with $f \in \Lambda$ and $\vec{z} \subseteq \vec{x}$

Split version of $\mathcal{M}$

For each $X \in \Psi$

$$\Phi' \rightsquigarrow X$$

where Φ' is the projection of Φ over the variables occurring in X.

Semantics of mappings

$\mathcal{I}$ satisfies a mapping assertion $\Phi \rightsquigarrow \Psi$ w.r.t. $\mathcal{S}$

if for each tuple of values $\vec{v} \in Eval(\Phi, \mathcal{S})$, and for each ground atom X in $\Psi[\vec{x}/\vec{v}]$,

if X has the form

$$\begin{cases} A(s) & \text{then } s^{\mathcal{I}} \in A^{\mathcal{I}} \\ T(s) & \text{then } s^{\mathcal{I}} \in T^{\mathcal{I}} \\ P(s_2, s_2) & \text{then } (s_1^{\mathcal{I}}, s_2^{\mathcal{I}}) \in P^{\mathcal{I}} \\ U(s_1, s_2) & \text{then } (s_1^{\mathcal{I}}, s_2^{\mathcal{I}}) \in U^{\mathcal{I}} \end{cases}$$

Example

$$D_1[\texttt{SSN} : \textbf{STRING}, \texttt{PROJ} : \textbf{STRING}, \texttt{D} : \textbf{DATE}],$$
$$D_2[\texttt{SSN} : \textbf{STRING}, \texttt{NAME} : \textbf{STRING}]$$

M_1 : SELECT SSN, PROJ,D FROM D_1 $\rightsquigarrow$ $tempEmp(\mathbf{pers}(\texttt{SSN}))$, $WORKS\_FOR(\mathbf{pers}(\texttt{SNN}), \mathbf{proj}(\texttt{PROJ}))$, $\text{ProjName}(\mathbf{proj}(\texttt{PROJ}), \texttt{PROJ})$, $\text{until}(\mathbf{pers}(\texttt{SNN}), \texttt{D})$

M_2 : SELECT SSN, NAME FROM D_2 $\rightsquigarrow$ $employee(\mathbf{pers}(\texttt{SSN}))$, $\text{PersName}(\mathbf{pers}(\texttt{SSN}), \texttt{NAME})$

Semantics of OBDI systems

Model of an OBDI system

An interpretation $\mathcal{I}$ is a model of $\mathcal{O}\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ if:

- $\mathcal{I}$ is a model of $\mathcal{T}$,
- $\mathcal{I}$ satisfies $\mathcal{M}$ w.r.t. $\mathcal{S}$, i.e., $\mathcal{I}$ satisfies every assertion in $\mathcal{M}$ w.r.t. $\mathcal{S}$.

An OBDI system $\mathcal{O}$ is satisfiable if it admits at least one model.

Query Answering on OB Data integration systems

Virtual ABox

Let $M \in \mathcal{M}$, $M = \Phi \rightsquigarrow X$.

$$\mathcal{A}_{M,\mathcal{S}} = \{X[\vec{x}/\vec{v}] \mid \vec{v} \in Eval(\Phi, \mathcal{S})\}$$

$$\mathcal{A}_{\mathcal{M},\mathcal{S}} = \{\mathcal{A}_{M,\mathcal{S}} \mid M \in \mathcal{M}\}$$

bottom-up approach:

- querying over $\mathcal{A}_{\mathcal{M},\mathcal{S}}$
- not really efficient in practice
- materializing the ABox is a PTIME process
- requires mechanisms for updating the ABox w.r.t. the database evolution

Top-Down Approach

Given an OBDI system $\mathcal{O}\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ the computation of the certain answers to an UCQ q consists of three steps:

1. Rewriting: Compute the perfect rewriting $q_{pr} = \mathit{PerfectRew}(q, \mathcal{T})$ of the original query q, using the inclusion assertions of the TBox $\mathcal{T}$.
2. Unfolding: Compute from q_{pr} a new query q_{unf} by unfolding q_{pr} using (the split version of) the mappings $\mathcal{M}$.
   q_{unf} is such that:.

$$Eval(q_{unf}, \mathcal{S}) = Eval(q_{pr}, \mathcal{A}_{\mathcal{M},\mathcal{S}})$$

3. Evaluation: Delegate the evaluation of q_{unf} to the relational DBMS managing $\mathcal{S}$.

Unfolding

The unfolding step is crucial for avoiding materializing the virtual ABox

To unfold a query q_{pr} with respect to a set of mapping assertions:

1. For each non-split mapping assertion $\Phi_i(\vec{x}) \rightsquigarrow \Psi_i(\vec{t}, \vec{y})$:

$$Aux_i(\vec{x}) \leftarrow \Phi_i(\vec{x}) \qquad \text{(view definition)}$$

2. For each split version $\Phi_i(\vec{x}) \rightsquigarrow X_j(\vec{t}, \vec{y})$ of a mapping assertion,

$$X_j(\vec{t}, \vec{y}) \leftarrow Aux_i(\vec{x}) \qquad \text{(clause)}$$

3. unify each atom $X(\vec{z})$ in the body of q_{pr} (in all possible ways) with the head of a clause $X(\vec{t}, \vec{y}) \leftarrow Aux_i(\vec{x})$.
4. Substitute each atom $X(\vec{z})$ with $\theta(Aux_i(\vec{x}))$,
5. The unfolded query q_{unf} is the union of all queries q_{aux} obtained, together with the view definitions for Aux_i appearing in q_{aux}.

Computational complexity of Query answering

From the top-down approach to query answering, and the complexity results for *DL-Lite*$_{\mathcal{A}}$, query answering in a $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is:

- Very efficiently tractable in the size of the database $\mathcal{S}$ (i.e., AC^0, and in fact FOL-rewritable).
- Efficiently tractable in the size of the TBox $\mathcal{T}$ and the mappings $\mathcal{M}$(i.e., PTIME).
- Exponential in the size of the query (i.e., NP-complete).

—

Can we move to LAV or GLAV mappings? No, if we want to stay in AC^0 [Calvanese et al., 2008].

Outline

The theoretical results indicate a good computational behavior in the size of the data. However, performance is a critical issue in practice:

- The rewriting consists of a large number of CQs. Query containment can be used to prune the rewriting. This is already implemented in the QuOnto system, but requires further optimizations.
- The SQL queries generated by the mapping unfolding are not easy to process by the DBMS engine (e.g., they may contain complex joins on skolem terms computed on the fly).
- Different mapping unfolding strategies have a strong impact on computational complexity. Experimentation is ongoing to assess the tradeoff.
- Further extensive experimentations are ongoing:
  - on artificially generated data;
  - on real-world use cases.

CWA or OWA?

Datalog$^{\pm}$

- Generalizes the *DL-Lite* family of DLs
- $+$ stratified negation while keeping Ontology querying tractable (polynomial in data complexity)
- Datalog alone can neither express *disjointness* nor *functionality*
- lack of value creation (e.g. $\mathit{employee} \sqsubseteq \exists \mathit{WORKS\text{-}FOR}$)

Additions to Datalog:

- Existentially quantified variables in rule heads
  $\rightsquigarrow$ tuple generating dependencies (TGDs)
- Rule bodies of TGDs are guarded
  $\rightsquigarrow$ guarded TGDs

$$P(X) \wedge R(X,Y) \wedge Q(Y) \rightarrow \exists Z.R(Y,Z)$$

- Bodies contain single atoms only $\rightsquigarrow$ linear TGDs
- Negative constraints and keys, e.g.

$$\mathit{employee}(X,Y) \wedge \mathit{retired}(X,Z) \rightarrow \bot$$

A Normal TGD (NTGD)

has the form

$$\forall X \forall Y \quad \underbrace{\Phi(X,Y)}_{\substack{\text{conj. of atoms} \\ \text{and neg. atoms}}} \quad \rightarrow \quad \exists Z \quad \underbrace{\Psi(X,Y)}_{\text{conj. of atoms}}$$

- guarded: a positive atom in its body contains X, Y
- linear: is guarded, and has exactly one positive atom in its body

A normal Boolean conjunctive query (NBCQ) $\mathcal{Q}$

is an existentially closed conjunction of atoms and negated atoms

$$\exists \vec{X} p_1(\vec{X}) \wedge \ldots \wedge p_m(\vec{X}) \wedge \neg p_{m+1}(\vec{X}) \wedge \ldots \wedge \neg p_{m+n}(\vec{X})$$

$\mathcal{Q}$ is safe iff every variable in a negative atoms also occurs in a positive atom

Theorem

- *Answering safe NBCQs in guarded Datalog$^{\pm}$ can be done in polynomial time in data complexity*
- *Answering safe NBCQs in linear Datalog$^{\pm}$ is FO-rewritable*

Thank you!

References

C. Beeri, A.Y. Levy, and M. C. Rousset

Rewriting Queries Using Views in Description Logics.
In Proc. PODS'97 (Symposium on Principles of Database Systems), pp. 99-108, 1997.

A. Cali, G. Gottlob, and T. Lukasiewicz

A general datalog-based framework for tractable query answering over ontologies.
In Proc. PODS'09 (Symposium on Principles of Database Systems), pp. 77-86, 2009.

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.

Tractable reasoning and efficient query answering in description logics: The DL-Lite family.
J. of Automated Reasoning 39(3), pp. 385-429, 2007.

A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati,

Linking Data to Ontologies.
J. Data Semantics 10, pp. 133-173, 2008.

R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev

Combined FO Rewritability for Conjunctive Query Answering in DL-Lite.
Description Logics'09 (International Workshop on Description Logics), 2009.

Further Reading

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.

Data complexity of query answering in description logics.
Proceedings of KR, 2006

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati,and M. Ruzzi.

Data integration through DL-Lite$_\mathcal{A}$ ontologies.
Proceeding of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)

C. Lutz, D. Toman, F. Wolter.

Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system,
Proceedings of IJCAI 2009.

H. Perez-Urbina, B. Motik, I. Horrocks

Tractable query answering and rewriting under description logic constraints,
J. Applied Logic, 2010

R. Rosati and A. Almatelli.

Improving query answering over DL-Lite ontologies.
Proceedings of KR 2010.