Lightweight Description Logics: *DL-Lite*$_{\mathcal{A}}$ and $\mathcal{EL}^{++}$

Elena Botoeva [1]

KRDB Research Centre
Free University of Bozen-Bolzano

January 13, 2011
Departamento de Ciencias de la Computación
Universidad de Chile, Santiago, Chile

[1] Part of the slides is borrowed from Diego Calvanese

Outline

1. Description Logics
2. Description Logic *DL-Lite*$_A$
   - Syntax and Semantics of *DL-Lite*$_A$
   - Reasoning in *DL-Lite*$_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering
3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

Outline

Description Logics

- formal languages for representing knowledge bases
  - TBox represents implicit knowledge (a set of axioms)
  - ABox represents explicit knowledge (a set of individual assertions)

Description Logics

- formal languages for representing knowledge bases
  - TBox represents implicit knowledge (a set of axioms)
  - ABox represents explicit knowledge (a set of individual assertions)
- talk about
  - concepts

    Professor Student Course $\top$ $\bot$,
  - and roles

    teaches attends

Description Logics

- formal languages for representing knowledge bases
  - TBox represents implicit knowledge (a set of axioms)
  - ABox represents explicit knowledge (a set of individual assertions)

- talk about
  - concepts

    $\mathsf{Professor} \quad \mathsf{Student} \quad \mathsf{Course} \quad \top \quad \bot$,
  - and roles

    $\mathsf{teaches} \quad \mathsf{attends}$

- variable free syntax
  - for describing complex concepts

    $\mathsf{Professor} \sqcup \mathsf{Student} \quad \exists \mathsf{teaches}.\mathsf{PhDCourse} \quad \forall \mathsf{hasChild}.\mathsf{Male}$
  - for asserting implicit knowledge

    $\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course} \quad \mathsf{Professor} \sqcap \mathsf{Student} \sqsubseteq \bot$
  - for asserting explicit knowledge

    $\mathsf{Student}(\mathtt{john}) \quad \mathsf{attends}(\mathtt{john}, \mathtt{db})$

Why Description Logics?

- *Decidable fragments of FOL* ($\Rightarrow$ Well-defined semantics).
  DLs provide sound and complete reasoning services:
  - checking knowledge base consistency,
  - checking logical entailment,
  - answering conjunctive queries (unions of CQ).

Why Description Logics?

- *Decidable fragments of FOL* (⇒ Well-defined semantics).
  DLs provide sound and complete reasoning services:
  - checking knowledge base consistency,
  - checking logical entailment,
  - answering conjunctive queries (unions of CQ).
- *Modelling capabilities*. Description Logics (DLs) can express, e.g.:
  - Taxonomy of classes of objects,
  - UML class diagrams,
  - ER models, etc.

Why Description Logics?

- *Decidable fragments of FOL* (⇒ Well-defined semantics). DLs provide sound and complete reasoning services:
  - checking knowledge base consistency,
  - checking logical entailment,
  - answering conjunctive queries (unions of CQ).
- *Modelling capabilities*. Description Logics (DLs) can express, e.g.:
  - Taxonomy of classes of objects,
  - UML class diagrams,
  - ER models, etc.
- *DLs are widely used nowadays*:
  - underly OWL 2, the Semantic Web standard,
  - serve as conceptual layer in Ontology Based Data Access,
  - for formalizing bio-medical domain, etc.

Lightweight Description Logics

The majority of studied DLs is intractable:

- Satisfiability of the basic DL $\mathcal{ALC}$ is ExpTime-complete.
- Satisfiability of $\mathcal{SROIQ}$, the basis of OWL 2, is 2NExpTime-complete.

Lightweight Description Logics

The majority of studied DLs is intractable:

- Satisfiability of the basic DL $\mathcal{ALC}$ is ExpTime-complete.
- Satisfiability of $\mathcal{SROIQ}$, the basis of OWL 2, is 2NExpTime-complete.

Two families of DLs that provide tractable reasoning have been developed, *DL-Lite* family by Calvanese et al. [5], and $\mathcal{EL}$ family by Baader et al. [2].

- A common feature: no disjunction and no universal restrictions

  $\mathsf{Professor} \sqcup \mathsf{Student} \quad \forall \mathsf{hasChild}.\mathsf{Male}$

Outline

1. Description Logics
2. Description Logic $DL\text{-}Lite_A$
   - Syntax and Semantics of $DL\text{-}Lite_A$
   - Reasoning in $DL\text{-}Lite_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering
3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

DL-Lite and *DL-Lite*$_A$

- *DL-Lite* is a family of tractable logics [5] specifically tailored to efficiently deal with large amounts of data.
  - Reasoning in *DL-Lite* are *FOL-rewritable*, i.e., we can reduce them to the problem of *query evaluation in relational databases*.
    $\Rightarrow$ AC^0 in data complexity.

DL-Lite and *DL-Lite*$_A$

- *DL-Lite* is a family of tractable logics [5] specifically tailored to efficiently deal with large amounts of data.
  - Reasoning in *DL-Lite* are *FOL-rewritable*, i.e., we can reduce them to the problem of *query evaluation in relational databases*. $\Rightarrow \mathrm{AC}^0$ in data complexity.
- *DL-Lite*$_{\mathcal{A}}$ is the most expressive member of this family.

Outline

1. Description Logics
2. Description Logic *DL-Lite*$_A$
   - Syntax and Semantics of *DL-Lite*$_A$
   - Reasoning in *DL-Lite*$_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering
3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

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

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.

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

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept and role constructs:

$$
\begin{array}{lllll}
B & ::= & A & \mid & \exists R & \text{basic concept} \\
C & ::= & B & \mid & \neg B & \text{complex concept} \\
R & ::= & P & \mid & P^- & \text{basic role}
\end{array}
$$

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

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept and role constructs:

$$\begin{array}{llll} B & ::= & A \mid \exists R & \text{basic concept} \\ C & ::= & B \mid \neg B & \text{complex concept} \\ R & ::= & P \mid P^- & \text{basic role} \end{array}$$

- TBox and ABox assertions:

$B_1 \sqsubseteq B_2$	concept inclusion		
$B_1 \sqsubseteq \neg B_2$	disjointness of concepts	$A(a)$	membership
$R_1 \sqsubseteq R_2$	role inclusion	$P(a,b)$	assertions
$\mathrm{Dis}(R_1, R_2)$	disjointness of roles		
$\mathrm{Funct}(R)$	role functionality		

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

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept and role constructs:

$$
\begin{array}{lllll}
B & ::= & A \mid \exists R & & \text{basic concept} \\
C & ::= & B \mid \neg B & & \text{complex concept} \\
R & ::= & P \mid P^- & & \text{basic role}
\end{array}
$$

- TBox and ABox assertions:

$B_1 \sqsubseteq B_2$	concept inclusion
$B_1 \sqsubseteq \neg B_2$	disjointness of concepts
$R_1 \sqsubseteq R_2$	role inclusion
$\mathrm{Dis}(R_1, R_2)$	disjointness of roles
$\mathrm{Funct}(R)$	role functionality

$A(a)$	membership
$P(a, b)$	assertions

- A *DL-Lite*$_\mathcal{A}$ Knowledge Base $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where
  - $\mathcal{T}$ is a finite set of TBox axioms and
  - $\mathcal{A}$ is a finite set of membership assertions.

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

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept and role constructs:

$$
\begin{array}{rcll}
B & ::= & A \mid \exists R & \text{basic concept} \\
C & ::= & B \mid \neg B & \text{complex concept} \\
R & ::= & P \mid P^- & \text{basic role}
\end{array}
$$

- TBox and ABox assertions:

$B_1 \sqsubseteq B_2$	concept inclusion		
$B_1 \sqsubseteq \neg B_2$	disjointness of concepts	$A(a)$	membership
$R_1 \sqsubseteq R_2$	role inclusion	$P(a,b)$	assertions
$\mathrm{Dis}(R_1, R_2)$	disjointness of roles		
$\mathrm{Funct}(R)$	role functionality		

- A *DL-Lite*$_\mathcal{A}$ Knowledge Base $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where
  - $\mathcal{T}$ is a finite set of TBox axioms and
  - $\mathcal{A}$ is a finite set of membership assertions.

Note: for simplicity attributes, value-domain expressions and identification constraints are not presented.

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

Syntactic restriction to ensure tractability:

Functional roles cannot be specialized.

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

Syntactic restriction to ensure tractability:

Functional roles cannot be specialized.

I.e., it is not allowed to have things like:

$$R' \sqsubseteq R$$
$$\mathsf{Funct}(R)$$

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

Syntactic restriction to ensure tractability:

Functional roles cannot be specialized.

I.e., it is not allowed to have things like:

$$R' \sqsubseteq R$$
$$\text{Funct}(R)$$

Otherwise, the resulting logic is ExpTime-hard in the size of ontology[1].

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

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$:
  - for every concept name A, $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$;
  - for every role name P, $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$;
  - for every individual name a, $a^\mathcal{I} \in \Delta^\mathcal{I}$.

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

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$:
  - for every concept name A, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;
  - for every role name P, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$;
  - for every individual name a, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
- Concept and role constructs

$$
\begin{aligned}
(\neg B)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \\
(\exists R)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}}\} \\
(P^{-})^{\mathcal{I}} &= \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in P^{\mathcal{I}}\}
\end{aligned}
$$

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

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$:
  - for every concept name A, $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$;
  - for every role name P, $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$;
  - for every individual name a, $a^\mathcal{I} \in \Delta^\mathcal{I}$.
- Concept and role constructs

$$
\begin{aligned}
(\neg B)^\mathcal{I} &= \Delta^\mathcal{I} \setminus B^\mathcal{I} \\
(\exists R)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I}, (x, y) \in R^\mathcal{I}\} \\
(P^-)^\mathcal{I} &= \{(y, x) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (x, y) \in P^\mathcal{I}\}
\end{aligned}
$$

- TBox and ABox assertions

$$
\begin{array}{lll}
\mathcal{I} \models B \sqsubseteq C & \textit{iff} & B^\mathcal{I} \subseteq C^\mathcal{I} \\
\mathcal{I} \models R_1 \sqsubseteq R_2 & \textit{iff} & {R_1}^\mathcal{I} \subseteq {R_2}^\mathcal{I} \\
\mathcal{I} \models \mathrm{Dis}(R_1, R_2) & \textit{iff} & {R_1}^\mathcal{I} \cap {R_2}^\mathcal{I} = \emptyset \\
\mathcal{I} \models \mathrm{Funct}(R) & \textit{iff} & (x, y_1) \in R^\mathcal{I}, \\
 & & (x, y_2) \in R^\mathcal{I} \\
 & & \Rightarrow y_1 = y_2
\end{array}
$$

$$
\begin{array}{lll}
\mathcal{I} \models A(a) & \textit{iff} & a^\mathcal{I} \in A^\mathcal{I} \\
\mathcal{I} \models P(a, b) & \textit{iff} & (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}
\end{array}
$$

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

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$:
  - for every concept name A, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;
  - for every role name P, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$;
  - for every individual name a, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
- Concept and role constructs

$$\begin{aligned}
(\neg B)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \\
(\exists R)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}}\} \\
(P^{-})^{\mathcal{I}} &= \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in P^{\mathcal{I}}\}
\end{aligned}$$

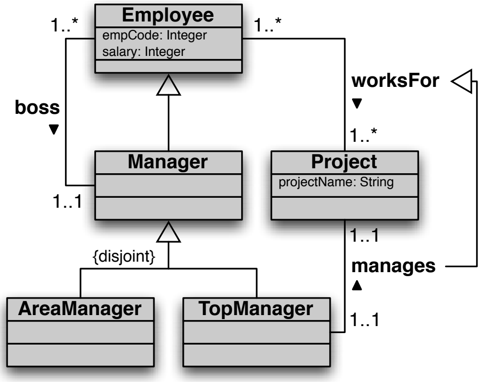
- TBox and ABox assertions

$$\begin{array}{lll}
\mathcal{I} \models B \sqsubseteq C & \textit{iff} & B^{\mathcal{I}} \subseteq C^{\mathcal{I}} \\
\mathcal{I} \models R_1 \sqsubseteq R_2 & \textit{iff} & {R_1}^{\mathcal{I}} \subseteq {R_2}^{\mathcal{I}} \\
\mathcal{I} \models \mathrm{Dis}(R_1, R_2) & \textit{iff} & {R_1}^{\mathcal{I}} \cap {R_2}^{\mathcal{I}} = \emptyset \\
\mathcal{I} \models \mathrm{Funct}(R) & \textit{iff} & (x, y_1) \in R^{\mathcal{I}}, \\
 & & (x, y_2) \in R^{\mathcal{I}} \\
 & & \Rightarrow y_1 = y_2
\end{array}$$

$$\begin{array}{lll}
\mathcal{I} \models A(a) & \textit{iff} & a^{\mathcal{I}} \in A^{\mathcal{I}} \\
\mathcal{I} \models P(a, b) & \textit{iff} & (a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}
\end{array}$$

- $\mathcal{I}$ is a *model* of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies all axioms of $\mathcal{T}$ and $\mathcal{A}$.

DL-Lite$_{\mathcal{A}}$ – Example



$$
\begin{array}{rcl}
\text{Manager} & \sqsubseteq & \text{Employee} \\
\text{AreaManager} & \sqsubseteq & \text{Manager} \\
\text{TopManager} & \sqsubseteq & \text{Manager} \\
\text{AreaManager} & \sqsubseteq & \neg\text{TopManager} \\
\exists\text{worksFor} & \sqsubseteq & \text{Employee} \\
\exists\text{worksFor}^{-} & \sqsubseteq & \text{Project} \\
\text{Employee} & \sqsubseteq & \exists\text{worksFor} \\
\text{Project} & \sqsubseteq & \exists\text{worksFor}^{-} \\
\multicolumn{3}{c}{\text{Funct}(\text{manages})} \\
\multicolumn{3}{c}{\text{Funct}(\text{manages}^{-})} \\
\text{manages} & \sqsubseteq & \text{worksFor} \\
 & \vdots &
\end{array}
$$

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

Outline

1. Description Logics
2. Description Logic $DL\text{-}Lite_A$
   - Syntax and Semantics of $DL\text{-}Lite_A$
   - Reasoning in $DL\text{-}Lite_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering
3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

Reasoning Problems

- The *Knowledge Base Satisfiability* problem is to check, given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model.

Reasoning Problems

- The *Knowledge Base Satisfiability* problem is to check, given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model.

- The *Query Answering* problem is to compute, given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K}$ and a query q (either a CQ or a UCQ) over $\mathcal{K}$, the set $ans(q, \mathcal{K})$ of certain answers.

Reasoning Problems

- The *Knowledge Base Satisfiability* problem is to check, given a *DL-Lite*$_{\mathcal{A}}$ KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model.
  - The *Concept Satisfiability* problem is to decide, given a TBox $\mathcal{T}$ and a concept C, whether there exist a model $\mathcal{I}$ of $\mathcal{T}$ such $C^{\mathcal{I}} \neq \emptyset$.
  - The *Concept Subsumption* problem is to decide, given a TBox $\mathcal{T}$ and concepts C_1 and C_2, whether for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that ${C_1}^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}$ ($\mathcal{T} \models C_1 \sqsubseteq C_2$).
  - The *Role Subsumption* problem is to decide, given a TBox $\mathcal{T}$ and roles R_1 and R_2, whether for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that ${R_1}^{\mathcal{I}} \subseteq {R_2}^{\mathcal{I}}$ ($\mathcal{T} \models R_1 \sqsubseteq R_2$).

- The *Query Answering* problem is to compute, given a *DL-Lite*$_{\mathcal{A}}$ KB $\mathcal{K}$ and a query q (either a CQ or a UCQ) over $\mathcal{K}$, the set $ans(q, \mathcal{K})$ of certain answers.
  - The *Concept Instance Checking* problem is to decide, given an object name a, a concept B, and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, whether $\mathcal{K} \models C(a)$.
  - The *Role Instance Checking* problem is to decide, given a pair (a, b), a role R, and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, whether $\mathcal{K} \models R(a, b)$.

First Order Logic Rewritability

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

- For each atomic concept A of the ontology:
  - define a unary relational table $\mathtt{tab}_A$
  - populate $\mathtt{tab}_A$ with each $\langle c \rangle$ such that $A(c) \in \mathcal{A}$
- For each atomic role P of the ontology,
  - define a binary relational table $\mathtt{tab}_P$
  - populate $\mathtt{tab}_P$ with each $\langle c_1, c_2 \rangle$ such that $P(c_1, c_2) \in \mathcal{A}$

We denote with $DB(\mathcal{A})$ the database obtained as above.

First Order Logic Rewritability

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

- For each atomic concept A of the ontology:
  - define a unary relational table $\texttt{tab}_A$
  - populate $\texttt{tab}_A$ with each $\langle c \rangle$ such that $A(c) \in \mathcal{A}$
- For each atomic role P of the ontology,
  - define a binary relational table $\texttt{tab}_P$
  - populate $\texttt{tab}_P$ with each $\langle c_1, c_2 \rangle$ such that $P(c_1, c_2) \in \mathcal{A}$

We denote with $DB(\mathcal{A})$ the database obtained as above.

Definition

KB satisfiability *(QA)* in *DL-Lite*$_{\mathcal{A}}$ is *FOL-rewritable* if,
for every $\mathcal{T}$ *(and every UCQ q)* there exists a FO query q',
such that for every nonempty $\mathcal{A}$ *(and every tuple of constants $\vec{a}$ from $\mathcal{A}$)*,
$\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable iff $q'()$ evaluates to false in $DB(\mathcal{A})$
($\vec{a} \in ans(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff $\vec{a}^{DB(\mathcal{A})} \in {q'}^{DB(\mathcal{A})}$).

First Order Logic Rewritability

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

- For each atomic concept A of the ontology:
  - define a unary relational table $\texttt{tab}_A$
  - populate $\texttt{tab}_A$ with each $\langle c \rangle$ such that $A(c) \in \mathcal{A}$
- For each atomic role P of the ontology,
  - define a binary relational table $\texttt{tab}_P$
  - populate $\texttt{tab}_P$ with each $\langle c_1, c_2 \rangle$ such that $P(c_1, c_2) \in \mathcal{A}$

We denote with $DB(\mathcal{A})$ the database obtained as above.

Definition

KB satisfiability *(QA)* in *DL-Lite*$_{\mathcal{A}}$ is *FOL-rewritable* if,
for every $\mathcal{T}$ *(and every UCQ q)* there exists a FO query q',
such that for every nonempty $\mathcal{A}$ *(and every tuple of constants $\vec{a}$ from $\mathcal{A}$)*,
$\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable iff $q'()$ evaluates to false in $DB(\mathcal{A})$
($\vec{a} \in ans(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff $\vec{a}^{DB(\mathcal{A})} \in q'^{DB(\mathcal{A})}$).

We show that KB satisfiability and QA in *DL-Lite*$_{\mathcal{A}}$ are FOL-rewritable.

Knowledge Base Satisfiability

Problem

Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, check whether there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$

Knowledge Base Satisfiability

Problem

Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, check whether there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$

- *Positive Inclusions* (PIs) are inclusions of the form

  $B_1 \sqsubseteq B_2, R_1 \sqsubseteq R_2$

- *Negative Inclusions* (NIs) are inclusions of the form

  $B_1 \sqsubseteq \neg B_2$, $\mathsf{Dis}(R_1, R_2)$, or $\mathsf{Funct}(R)$

Satisfiability of KBs with only PIs

Positive inclusions cannot introduce contradicting information:

Satisfiability of KBs with only PIs

Positive inclusions cannot introduce contradicting information:

Theorem

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $DL\text{-}Lite_{\mathcal{A}}$ KB such that $\mathcal{T}$ consists only of PIs. Then $\mathcal{K}$ is satisfiable.

Satisfiability of KBs with only PIs

Positive inclusions cannot introduce contradicting information:

Theorem

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{A}$ KB such that $\mathcal{T}$ consists only of PIs. Then $\mathcal{K}$ is satisfiable.

We can always build a model by adding missing tuples to satisfy PIs.

Source of Unsatisfiability

However, negative inclusions can cause a KB to be unsatisfiable:

Source of Unsatisfiability

However, negative inclusions can cause a KB to be unsatisfiable:

- $\mathcal{T}$: $\mathsf{Dis}(\mathsf{teaches}, \mathsf{attends})$
  $\mathcal{A}$: $\mathtt{teaches}(\mathtt{john}, \mathtt{db})$, $\mathtt{attends}(\mathtt{john}, \mathtt{db})$

Source of Unsatisfiability

However, negative inclusions can cause a KB to be unsatisfiable:

- $\mathcal{T}$: $\mathsf{Dis}(\mathsf{teaches}, \mathsf{attends})$
  $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{db})$, $\mathsf{attends}(\mathtt{john}, \mathtt{db})$
- $\mathcal{T}$: $\mathsf{Funct}(\mathsf{teaches}^{-})$
  $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{db})$, $\mathsf{teaches}(\mathtt{david}, \mathtt{db})$

Source of Unsatisfiability

However, negative inclusions can cause a KB to be unsatisfiable:

- $\mathcal{T}$: $\mathsf{Dis}(\mathsf{teaches}, \mathsf{attends})$
  $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{db}),\ \mathsf{attends}(\mathtt{john}, \mathtt{db})$
- $\mathcal{T}$: $\mathsf{Funct}(\mathsf{teaches}^-)$
  $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{db}),\ \mathsf{teaches}(\mathtt{david}, \mathtt{db})$
- $\mathcal{T}$: $\mathsf{Student} \sqsubseteq \neg\mathsf{Professor},\ \exists\mathsf{teaches} \sqsubseteq \mathsf{Professor}$
  $\mathcal{A}$: $\mathsf{Student}(\mathtt{john}),\ \mathsf{teaches}(\mathtt{john}, \mathtt{db})$

Source of Unsatisfiability

However, negative inclusions can cause a KB to be unsatisfiable:

- $\mathcal{T}$: $\mathsf{Dis}(\mathsf{teaches}, \mathsf{attends})$
  $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{db}),\ \mathsf{attends}(\mathtt{john}, \mathtt{db})$
- $\mathcal{T}$: $\mathsf{Funct}(\mathsf{teaches}^-)$
  $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{db}),\ \mathsf{teaches}(\mathtt{david}, \mathtt{db})$
- $\mathcal{T}$: $\mathsf{Student} \sqsubseteq \neg\mathsf{Professor},\ \exists\mathsf{teaches} \sqsubseteq \mathsf{Professor}$
  $\mathcal{A}$: $\mathsf{Student}(\mathtt{john}),\ \mathsf{teaches}(\mathtt{john}, \mathtt{db})$
    - *Interaction* of negative and positive inclusions has to be considered.
      $\Rightarrow$ calculate the *closure* of NIs w.r.t. PIs.

Knowledge Base Satisfiability

Given a *DL-Lite*$_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we check its satisfiability as follows:

Knowledge Base Satisfiability

Given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we check its satisfiability as follows:

Algorithm for checking KB satisfiability

1. Calculate the closure of NIs.
2. Translate the closure into a UCQ q_{unsat} asking for violation of some NI.
3. Evaluate encoding of q_{unsat} into SQL over $DB(\mathcal{A})$.
   - if $Eval(SQL(q_{unsat}), DB(\mathcal{A})) = \emptyset$, then the KB is satisfiable;
   - otherwise the KB is unsatisfiable.

Knowledge Base Satisfiability

Given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we check its satisfiability as follows:

Algorithm for checking KB satisfiability

1. Calculate the closure of NIs.
2. Translate the closure into a UCQ q_{unsat} asking for violation of some NI.
3. Evaluate encoding of q_{unsat} into SQL over $DB(\mathcal{A})$.
    - if $Eval(SQL(q_{unsat}), DB(\mathcal{A})) = \emptyset$, then the KB is satisfiable;
    - otherwise the KB is unsatisfiable.

Correctness of this procedure shows FOL-rewritability of KB satisfiability in *DL-Lite*.

Closure of Negative Inclusions

Closure of NIs cln$(\mathcal{T})$ w.r.t. PIs

- every NI is in *cln*$(\mathcal{T})$.

Closure of Negative Inclusions

Closure of NIs cln$(\mathcal{T})$ w.r.t. PIs

- every NI is in *cln*$(\mathcal{T})$.
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Student} \sqsubseteq \neg\text{Professor} \\ \mathcal{T}: & \exists\text{teaches} \sqsubseteq \text{Professor} \end{array}\right\} \Rightarrow$

Closure of Negative Inclusions

Closure of NIs $cln(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.
- $$\left.\begin{array}{ll} cln(\mathcal{T}): & \mathsf{Student} \sqsubseteq \neg\mathsf{Professor} \\ \mathcal{T}: & \exists\mathsf{teaches} \sqsubseteq \mathsf{Professor} \end{array}\right\} \Rightarrow$$
  add to $cln(\mathcal{T})$: $\mathsf{Student} \sqsubseteq \neg\exists\mathsf{teaches}$

Closure of Negative Inclusions

Closure of NIs cln($\mathcal{T}$) w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.

- $$\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Student} \sqsubseteq \neg\text{Professor} \\ \mathcal{T}: & \exists\text{teaches} \sqsubseteq \text{Professor} \end{array}\right\} \Rightarrow$$
  add to $cln(\mathcal{T})$: $\text{Student} \sqsubseteq \neg\exists\text{teaches}$

- $$\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Professor} \sqsubseteq \neg\exists\text{attends} \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$$

Closure of Negative Inclusions

Closure of NIs $cln(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.

- $$\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Student} \sqsubseteq \neg\text{Professor} \\ \mathcal{T}: & \exists\text{teaches} \sqsubseteq \text{Professor} \end{array}\right\} \Rightarrow$$
  add to $cln(\mathcal{T})$: Student $\sqsubseteq \neg\exists$teaches

- $$\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Professor} \sqsubseteq \neg\exists\text{attends} \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$$
  add to $cln(\mathcal{T})$: Professor $\sqsubseteq \neg\exists$registeredTo

Closure of Negative Inclusions

Closure of NIs cln$(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.
- $cln(\mathcal{T})$: Student $\sqsubseteq \neg$Professor
  $\mathcal{T}$: $\exists$teaches $\sqsubseteq$ Professor $\Rightarrow$
  add to $cln(\mathcal{T})$: Student $\sqsubseteq \neg\exists$teaches
- $cln(\mathcal{T})$: Professor $\sqsubseteq \neg\exists$attends
  $\mathcal{T}$: registeredTo $\sqsubseteq$ attends $\Rightarrow$
  add to $cln(\mathcal{T})$: Professor $\sqsubseteq \neg\exists$registeredTo
- $cln(\mathcal{T})$: Dis(teaches, attends)
  $\mathcal{T}$: registeredTo $\sqsubseteq$ attends $\Rightarrow$

Closure of Negative Inclusions

Closure of NIs $cln(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Student} \sqsubseteq \neg\text{Professor} \\ \mathcal{T}: & \exists\text{teaches} \sqsubseteq \text{Professor} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Student} \sqsubseteq \neg\exists\text{teaches}$
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Professor} \sqsubseteq \neg\exists\text{attends} \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Professor} \sqsubseteq \neg\exists\text{registeredTo}$
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Dis}(\text{teaches}, \text{attends}) \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Dis}(\text{teaches}, \text{registeredTo})$

Closure of Negative Inclusions

Closure of NIs $cln(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Student} \sqsubseteq \neg\text{Professor} \\ \mathcal{T}: & \exists\text{teaches} \sqsubseteq \text{Professor} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Student} \sqsubseteq \neg\exists\text{teaches}$
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Professor} \sqsubseteq \neg\exists\text{attends} \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Professor} \sqsubseteq \neg\exists\text{registeredTo}$
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Dis}(\text{teaches}, \text{attends}) \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Dis}(\text{teaches}, \text{registeredTo})$
- ...

Closure of Negative Inclusions

Closure of NIs $cln(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.
- $cln(\mathcal{T})$: $\mathsf{Student} \sqsubseteq \neg\mathsf{Professor}$ (or $\mathsf{Professor} \sqsubseteq \neg\mathsf{Student}$)
  $\mathcal{T}$: $\exists\mathsf{teaches} \sqsubseteq \mathsf{Professor}$ $\Rightarrow$
  add to $cln(\mathcal{T})$: $\mathsf{Student} \sqsubseteq \neg\exists\mathsf{teaches}$
- $cln(\mathcal{T})$: $\mathsf{Professor} \sqsubseteq \neg\exists\mathsf{attends}$ (or $\exists\mathsf{attends} \sqsubseteq \neg\mathsf{Professor}$)
  $\mathcal{T}$: $\mathsf{registeredTo} \sqsubseteq \mathsf{attends}$ $\Rightarrow$
  add to $cln(\mathcal{T})$: $\mathsf{Professor} \sqsubseteq \neg\exists\mathsf{registeredTo}$
- $cln(\mathcal{T})$: $\mathsf{Dis}(\mathsf{teaches}, \mathsf{attends})$ (or $\mathsf{Dis}(\mathsf{attends}, \mathsf{teaches})$)
  $\mathcal{T}$: $\mathsf{registeredTo} \sqsubseteq \mathsf{attends}$ $\Rightarrow$
  add to $cln(\mathcal{T})$: $\mathsf{Dis}(\mathsf{teaches}, \mathsf{registeredTo})$
- ...

Closure of Negative Inclusions

Closure of NIs $cln(\mathcal{T})$ w.r.t. PIs

- every NI is in $cln(\mathcal{T})$.
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Student} \sqsubseteq \neg\text{Professor} \\ \mathcal{T}: & \exists\text{teaches} \sqsubseteq \text{Professor} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Student} \sqsubseteq \neg\exists\text{teaches}$
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Professor} \sqsubseteq \neg\exists\text{attends} \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Professor} \sqsubseteq \neg\exists\text{registeredTo}$
- $\left.\begin{array}{ll} cln(\mathcal{T}): & \text{Dis}(\text{teaches}, \text{attends}) \\ \mathcal{T}: & \text{registeredTo} \sqsubseteq \text{attends} \end{array}\right\} \Rightarrow$
  add to $cln(\mathcal{T})$: $\text{Dis}(\text{teaches}, \text{registeredTo})$
- ...

Note: functionality does not interact with PIs and other NIs.

Note: the closure is finite since there are polynomially many different NIs.

Translation to FOL Queries

Having calculated $cln(\mathcal{T})$ we translate it to a $\text{UCQ}_{\neq}$ q_{unsat} as follows.

- Each NI α correspond to a CQ, $\delta(\alpha)$:
  - Student $\sqsubseteq \neg\exists$teaches $\Rightarrow$
    $\exists x.\text{Student}(x) \wedge \exists y.\text{teaches}(x, y)$.

Translation to FOL Queries

Having calculated $cln(\mathcal{T})$ we translate it to a UCQ$_{\neq}$ q_{unsat} as follows.

- Each NI α correspond to a CQ, $\delta(\alpha)$:
  - Student $\sqsubseteq \neg\exists$teaches $\Rightarrow$
    $\exists x.\text{Student}(x) \wedge \exists y.\text{teaches}(x, y)$.
  - Funct(teaches$^-$) $\Rightarrow$
    $\exists x_1, x_2, y.\text{teaches}(x_1, y) \wedge \text{teaches}(x_2, y) \wedge x_1 \neq x_2$.

Translation to FOL Queries

Having calculated $cln(\mathcal{T})$ we translate it to a $\text{UCQ}_{\neq}$ q_{unsat} as follows.

- Each NI α correspond to a CQ, $\delta(\alpha)$:
  - $\text{Student} \sqsubseteq \neg\exists\text{teaches} \Rightarrow$
    $\exists x.\text{Student}(x) \wedge \exists y.\text{teaches}(x, y)$.
  - $\text{Funct}(\text{teaches}^{-}) \Rightarrow$
    $\exists x_1, x_2, y.\text{teaches}(x_1, y) \wedge \text{teaches}(x_2, y) \wedge x_1 \neq x_2$.
  - $\text{Dis}(\text{attends}, \text{teaches}) \Rightarrow$
    $\exists x, y.\text{attends}(x, y) \wedge \text{teaches}(x, y)$.
- Then

$$q_{unsat} = \bigvee_{\alpha \in cln(\mathcal{T})} \delta(\alpha)$$

Query evaluation

Let q be a UCQ.

- We denote by $SQL(q)$ the encoding of q into an SQL query over $DB(\mathcal{A})$.
- We indicate with $Eval(SQL(q), DB(\mathcal{A}))$ the evaluation of $SQL(q)$ over $DB(\mathcal{A})$.

FOL-rewritability of satisfiability in *DL-Lite*$_\mathcal{A}$

Theorem

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_\mathcal{A}$ KB. Then, $\mathcal{K}$ is unsatisfiable iff $Eval(SQL(q_{unsat}, DB(\mathcal{A}))$ returns true.

In other words, satisfiability of a *DL-Lite*$_\mathcal{A}$ ontology can be reduced to FOL-query evaluation.

Query Answering

Problem

Query answering over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a form of *logical implication*:

find all tuples $\vec{c}$ of constants of $\mathcal{A}$ s.t. $\mathcal{K} \models q(\vec{c})$

We are interested in so called *certain answers*, i.e., the tuples that are answers to q in all models of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$:

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

Note: We have assumed that the answer $q^{\mathcal{I}}$ to a query q over an interpretation $\mathcal{I}$ is constituted by a set of tuples of constants of $\mathcal{A}$, rather than objects in $\Delta^{\mathcal{I}}$.

Query Answering over Satisfiable KBs

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

Algorithm for answering CQs over KBs

1. Using $\mathcal{T}$, rewrite q into a UCQ $r_{q,\mathcal{T}}$ (the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. Encode $r_{q,\mathcal{T}}$ into SQL and evaluate it over $\mathcal{A}$ managed in secondary storage via a RDBMS, to return $cert(q, \mathcal{K})$.

Query Answering over Satisfiable KBs

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

Algorithm for answering CQs over KBs

1. Using $\mathcal{T}$, rewrite q into a UCQ $r_{q,\mathcal{T}}$ (the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. Encode $r_{q,\mathcal{T}}$ into SQL and evaluate it over $\mathcal{A}$ managed in secondary storage via a RDBMS, to return $cert(q, \mathcal{K})$.

Correctness of this procedure shows FOL-rewritability of query answering in *DL-Lite*.

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

Query Rewriting

Consider the query $q(x) \leftarrow \text{Professor}(x)$

Intuition: Use the PIs as basic rewriting rules:

$$\text{AssistantProf} \sqsubseteq \text{Professor}$$

as a logic rule: $\text{Professor}(z) \leftarrow \text{AssistantProf}(z)$

Query Rewriting

Consider the query $\quad q(x) \leftarrow \text{Professor}(x)$

Intuition: Use the PIs as basic rewriting rules:

$$\text{AssistantProf} \sqsubseteq \text{Professor}$$

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

Basic rewriting step:

when an atom in the query unifies with the **head** of the rule,

substitute the atom with the **body** of the rule.

We say that the PI inclusion applies to the atom.

Query Rewriting

Consider the query $\quad \mathsf{q}(x) \leftarrow \mathsf{Professor}(x)$

Intuition: Use the PIs as basic rewriting rules:

$$\mathsf{AssistantProf} \sqsubseteq \mathsf{Professor}$$

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

Basic rewriting step:

when an atom in the query unifies with the **head** of the rule,

substitute the atom with the **body** of the rule.

We say that the PI inclusion applies to the atom.

In the example, the PI $\mathsf{AssistantProf} \sqsubseteq \mathsf{Professor}$ applies to the atom $\mathsf{Professor}(x)$. Towards the computation of the perfect rewriting, we add to the input query above, the query

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

Query Rewriting (cont'd)

Consider the query $q(x) \leftarrow \text{teaches}(x, y), \text{Course}(y)$

and the PI $\exists\text{teaches}^- \sqsubseteq \text{Course}$

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

The PI applies to the atom $\text{Course}(y)$, and we add to the perfect rewriting the query

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

Query Rewriting (cont'd)

Consider the query $q(x) \leftarrow teaches(x, y), Course(y)$

and the PI $\exists teaches^- \sqsubseteq Course$

as a logic rule: $Course(z_2) \leftarrow teaches(z_1, z_2)$

The PI applies to the atom $Course(y)$, and we add to the perfect rewriting the query

$$q(x) \leftarrow teaches(x, y), teaches(z_1, y)$$

Consider now the query $q(x) \leftarrow teaches(x, y)$

and the PI $Professor \sqsubseteq \exists teaches$

as a logic rule: $teaches(z, f(z)) \leftarrow Professor(z)$

The PI applies to the atom $teaches(x, y)$, and we add to the perfect rewriting the query

$$q(x) \leftarrow Professor(x)$$

Query Rewriting – Constants

Conversely, for the query $\quad q(x) \leftarrow \text{teaches}(x, \texttt{databases})$

and the same PI as before $\quad \text{Professor} \sqsubseteq \exists\text{teaches}$

as a logic rule: $\quad \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z)$

$\text{teaches}(x, \texttt{databases})$ does not unify with $\text{teaches}(z, f(z))$, since the skolem term $f(z)$ in the head of the rule does not unify with the constant $\texttt{databases}$.

Query Rewriting – Constants

Conversely, for the query $\quad q(x) \leftarrow \text{teaches}(x, \texttt{databases})$

and the same PI as before $\quad \text{Professor} \sqsubseteq \exists\text{teaches}$

as a logic rule: $\quad \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z)$

$\text{teaches}(x, \texttt{databases})$ does not unify with $\text{teaches}(z, f(z))$, since the skolem term $f(z)$ in the head of the rule does not unify with the constant $\texttt{databases}$.

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

Query Rewriting – Constants

Conversely, for the query $\quad q(x) \leftarrow \text{teaches}(x, \texttt{databases})$

and the same PI as before $\quad \text{Professor} \sqsubseteq \exists\text{teaches}$

as a logic rule: $\quad \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z)$

$\text{teaches}(x, \texttt{databases})$ does not unify with $\text{teaches}(z, f(z))$, since the skolem term $f(z)$ in the head of the rule does not unify with the constant $\texttt{databases}$.

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

The same holds for the following query, where y is distinguished, since unifying $f(z)$ with y would correspond to returning a skolem term as answer to the query:

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

Query Rewriting – Join variables

An analogous behavior to the one with constants and with distinguished variables holds when the atom contains join variables that would have to be unified with skolem terms.

Consider the query $\quad \text{q}(x) \leftarrow \text{teaches}(x, y), \text{Course}(y)$

and the PI $\quad \text{Professor} \sqsubseteq \exists\text{teaches}$

as a logic rule: $\quad \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z)$

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

Query Rewriting – Reduce step

Consider now the query $q(x) \leftarrow teaches(x, y), teaches(z, y)$

and the PI $Professor \sqsubseteq \exists teaches$

as a logic rule: $teaches(z, f(z)) \leftarrow Professor(z)$

This PI does not apply to $teaches(x, y)$ or $teaches(z, y)$, since y is in join, and we would again introduce the skolem term in the rewritten query.

Query Rewriting – Reduce step

Consider now the query $q(x) \leftarrow teaches(x, y), teaches(z, y)$

and the PI $Professor \sqsubseteq \exists teaches$

as a logic rule: $teaches(z, f(z)) \leftarrow Professor(z)$

This PI does not apply to $teaches(x, y)$ or $teaches(z, y)$, since y is in join, and we would again introduce the skolem term in the rewritten query.

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

$$q(x) \leftarrow teaches(x, y)$$

Now, we can apply the PI above, and add to the rewriting the query

$$q(x) \leftarrow Professor(x)$$

Query Rewriting Algorithm

Algorithm PerfectRef$(Q, \mathcal{T}_P)$
Input: union of conjunctive queries Q, set of $DL\text{-}Lite_{\mathcal{A}}$ 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

Observations:

- Termination follows from having only finitely many different rewritings.
- NIs or functionalities do not play any role in the rewriting of the query.

Query answering in *DL-Lite* – Example

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

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

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

ABox: $\mathsf{teaches}(\mathtt{john}, \mathtt{databases})$
$\mathsf{Professor}(\mathtt{mary})$

It is easy to see that evaluating the perfect rewriting over the ABox viewed as a database produces as answer $\{\mathtt{john}, \mathtt{mary}\}$.

Query answering in *DL-Lite*

Theorem

Let $\mathcal{T}$ be a DL-Lite TBox, $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$, q a CQ over $\mathcal{T}$, and let $r_{q,\mathcal{T}} = \mathrm{PerfectRef}(q, \mathcal{T}_P)$. Then, for each ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, we have that

$$cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = Eval(SQL(r_{q,\mathcal{T}}), DB(\mathcal{A})).$$

In other words, query answering over a satisfiable *DL-Lite* ontology is FOL-rewritable.

Query answering in *DL-Lite*

Theorem

Let $\mathcal{T}$ be a DL-Lite TBox, $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$, q a CQ over $\mathcal{T}$, and let $r_{q,\mathcal{T}} = \text{PerfectRef}(q, \mathcal{T}_P)$. Then, for each ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, we have that

$$cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = Eval(SQL(r_{q,\mathcal{T}}), DB(\mathcal{A})).$$

In other words, query answering over a satisfiable *DL-Lite* ontology is FOL-rewritable.

Notice that we did not mention NIs or functionality assertions of $\mathcal{T}$ in the result above. Indeed, *when the ontology is satisfiable, we can ignore NIs and functionalities and answer queries as if they were not specified in $\mathcal{T}$*.

Complexity of Reasoning in *DL-Lite*

Theorem

Checking satisfiability of DL-Lite$_\mathcal{A}$ *KBs is*

1. PTIME *in the size of the KB (combined complexity).*
2. AC^0 *in the size of the ABox (data complexity).*

Theorem

Query answering over DL-Lite$_\mathcal{A}$ *KBs is*

1. NP-*complete in the size of query and KB (combined comp.).*
2. PTIME *in the size of the KB.*
3. AC^0 *in the size of the ABox (data complexity).*

Outline

1. Description Logics

2. Description Logic $DL\text{-}Lite_A$
   - Syntax and Semantics of $DL\text{-}Lite_A$
   - Reasoning in $DL\text{-}Lite_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering

3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

$\mathcal{EL}$ and $\mathcal{EL}^{++}$

$\mathcal{EL}$ is another family of tractable logics [2, 3].

- it is expressive enough to model bio-medical ontologies like SNOMED;
- allows for horn inclusions and qualified existential restrictions:

$$\text{Heartdisease} \sqcap \exists \text{has-loc.HeartValve} \sqsubseteq \text{CriticalDisease}$$

Outline

1. Description Logics
2. Description Logic *DL-Lite*$_A$
   - Syntax and Semantics of *DL-Lite*$_A$
   - Reasoning in *DL-Lite*$_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering
3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

$\mathcal{EL}^{++}$ Syntax

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.

$\mathcal{EL}^{++}$ Syntax

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept constructs:

$$C, D \ ::= \ \top \mid \bot \mid A \mid \{a\} \mid C \sqcap D \mid \exists P.C$$

$\mathcal{EL}^{++}$ Syntax

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept constructs:

$$C, D \ ::= \ \top \mid \bot \mid A \mid \{a\} \mid C \sqcap D \mid \exists P.C$$

- TBox and ABox assertions:

$C \sqsubseteq D$	concept inclusion	$A(a)$	membership
$P_1 \circ \cdots \circ P_n \sqsubseteq P$	complex role inclusion	$P(a, b)$	assertions

$\mathcal{EL}^{++}$ Syntax

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept constructs:

$$C, D \;::=\; \top \mid \bot \mid A \mid \{a\} \mid C \sqcap D \mid \exists P.C$$

- TBox and ABox assertions:

$C \sqsubseteq D$	concept inclusion	$A(a)$	membership
$P_1 \circ \cdots \circ P_n \sqsubseteq P$	complex role inclusion	$P(a,b)$	assertions

- An $\mathcal{EL}^{++}$ Knowledge Base $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where
  - $\mathcal{T}$ is a finite set of TBox axioms and
  - $\mathcal{A}$ is a finite set of membership assertions.

$\mathcal{EL}^{++}$ Syntax

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept constructs:

$$C, D \;::=\; \top \mid \bot \mid A \mid \{a\} \mid C \sqcap D \mid \exists P.C$$

- TBox and ABox assertions:

$C \sqsubseteq D$	concept inclusion	$A(a)$	membership
$P_1 \circ \cdots \circ P_n \sqsubseteq P$	complex role inclusion	$P(a, b)$	assertions

- An $\mathcal{EL}^{++}$ Knowledge Base $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where
  - $\mathcal{T}$ is a finite set of TBox axioms and
  - $\mathcal{A}$ is a finite set of membership assertions.

Note: the concrete domain constructor, which is a part of $\mathcal{EL}^{++}$, is not presented here.

Note: complex role inclusions allow expressing transitivity of roles ($P \circ P \sqsubseteq P$) and role hierarchy ($P_1 \sqsubseteq P_2$).

$\mathcal{EL}^{++}$ Syntax

- Let N_A, N_P, N_a be sets of concept, role and individual names, respectively. Let $A \in N_A$, $P \in N_P$, $a \in N_a$.
- Concept constructs:

$$C, D \ ::= \ \top \mid \bot \mid A \mid \{a\} \mid C \sqcap D \mid \exists P.C$$

- TBox and ABox assertions:

$C \sqsubseteq D$	concept inclusion	$A(a)$	membership
$P_1 \circ \cdots \circ P_n \sqsubseteq P$	complex role inclusion	$P(a, b)$	assertions

- An $\mathcal{EL}^{++}$ Knowledge Base $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where
  - $\mathcal{T}$ is a finite set of TBox axioms and
  - $\mathcal{A}$ is a finite set of membership assertions.

$\mathcal{EL}$ concept constructs and assertions.

$\mathcal{EL}^{++}$ Semantics

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$:
  - for every concept name A, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;
  - for every role name P, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$;
  - for every individual name a, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

$\mathcal{EL}^{++}$ Semantics

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$:
  - for every concept name A, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;
  - for every role name P, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$;
  - for every individual name a, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
- Concept constructs

$$\begin{array}{rcl} (\top)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\ (\bot)^{\mathcal{I}} & = & \emptyset \\ (\{a\})^{\mathcal{I}} & = & \{a^{\mathcal{I}}\} \end{array} \qquad \begin{array}{rcl} (C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\exists P.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \exists y \in C^{\mathcal{I}}, (x,y) \in P^{\mathcal{I}}\} \end{array}$$

$\mathcal{EL}^{++}$ Semantics

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$:
  - for every concept name A, $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$;
  - for every role name P, $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$;
  - for every individual name a, $a^\mathcal{I} \in \Delta^\mathcal{I}$.

- Concept constructs

$$\begin{aligned} (\top)^\mathcal{I} &= \Delta^\mathcal{I} \\ (\bot)^\mathcal{I} &= \emptyset \\ (\{a\})^\mathcal{I} &= \{a^\mathcal{I}\} \end{aligned} \qquad \begin{aligned} (C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\ (\exists P.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} \mid \exists y \in C^\mathcal{I}, (x,y) \in P^\mathcal{I}\} \end{aligned}$$

- TBox and ABox assertions

$$\begin{array}{lll} \mathcal{I} \models C \sqsubseteq D & \textit{iff} & C^\mathcal{I} \subseteq D^\mathcal{I} \\ \mathcal{I} \models P_1 \circ \cdots \circ P_n \sqsubseteq P & \textit{iff} & {P_1}^\mathcal{I} \circ \cdots \circ {P_n}^\mathcal{I} \subseteq P^\mathcal{I} \\ \mathcal{I} \models A(a) & \textit{iff} & a^\mathcal{I} \in A^\mathcal{I} \\ \mathcal{I} \models P(a,b) & \textit{iff} & (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I} \end{array}$$

$\mathcal{EL}^{++}$ Semantics

- An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$:
  - for every concept name A, $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$;
  - for every role name P, $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$;
  - for every individual name a, $a^\mathcal{I} \in \Delta^\mathcal{I}$.
- Concept constructs

$$\begin{aligned} (\top)^\mathcal{I} &= \Delta^\mathcal{I} \\ (\bot)^\mathcal{I} &= \emptyset \\ (\{a\})^\mathcal{I} &= \{a^\mathcal{I}\} \end{aligned} \qquad \begin{aligned} (C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\ (\exists P.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} \mid \exists y \in C^\mathcal{I}, (x,y) \in P^\mathcal{I}\} \end{aligned}$$

- TBox and ABox assertions

$$\begin{array}{lll} \mathcal{I} \models C \sqsubseteq D & \textit{iff} & C^\mathcal{I} \subseteq D^\mathcal{I} \\ \mathcal{I} \models P_1 \circ \cdots \circ P_n \sqsubseteq P & \textit{iff} & {P_1}^\mathcal{I} \circ \cdots \circ {P_n}^\mathcal{I} \subseteq P^\mathcal{I} \\ \mathcal{I} \models A(a) & \textit{iff} & a^\mathcal{I} \in A^\mathcal{I} \\ \mathcal{I} \models P(a,b) & \textit{iff} & (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I} \end{array}$$

- $\mathcal{I}$ is a *model* of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies all axioms of $\mathcal{T}$ and $\mathcal{A}$.

$\mathcal{EL}^{++}$: Example [2]

$$
\begin{array}{rcl}
\text{Endocardium} & \sqsubseteq & \text{Tissue} \sqcap \exists \text{cont-in.HeartWall} \sqcap \exists \text{cont-in.HeartValve} \\
\text{HeartWall} & \sqsubseteq & \text{BodyWall} \sqcap \exists \text{part-of.Heart} \\
\text{HeartValve} & \sqsubseteq & \text{BodyValve} \sqcap \exists \text{part-of.Heart} \\
\text{Endocarditis} & \sqsubseteq & \text{Inflammation} \sqcap \exists \text{has-loc.Endocardium} \\
\text{Inflammation} & \sqsubseteq & \text{Disease} \sqcap \exists \text{acts-on.Tissue} \\
\text{Heartdisease} \sqcap & & \\
\exists \text{has-loc.HeartValve} & \sqsubseteq & \text{CriticalDisease} \\
\text{Heartdisease} & \equiv & \text{Disease} \sqcap \exists \text{has-loc.Heart} \\
\text{part-of} \circ \text{part-of} & \sqsubseteq & \text{part-of} \\
\text{part-of} & \sqsubseteq & \text{cont-in} \\
\text{has-loc} \circ \text{cont-in} & \sqsubseteq & \text{has-loc}
\end{array}
$$

[2] taken from [4]

Outline

1. Description Logics
2. Description Logic $DL\text{-}Lite_A$
   - Syntax and Semantics of $DL\text{-}Lite_A$
   - Reasoning in $DL\text{-}Lite_A$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering
3. Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$

Reasoning Problems

- The *Concept Subsumption* problem is to decide, given a TBox $\mathcal{T}$ and concepts C and D, whether for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Reasoning Problems

- The *Concept Subsumption* problem is to decide, given a TBox $\mathcal{T}$ and concepts C and D, whether for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

- The *Conjunctive Query Entailment* problem is to decide, given a *DL-Lite*$_{\mathcal{A}}$ KB $\mathcal{K}$ and a boolean query q over $\mathcal{K}$, whether $\mathcal{K} \models q$.

Reasoning Problems

- The *Concept Subsumption* problem is to decide, given a TBox $\mathcal{T}$ and concepts C and D, whether for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
  - The *Concept Satisfiability* problem is to decide, given a TBox $\mathcal{T}$ and a concept C, whether there exist a model $\mathcal{I}$ of $\mathcal{T}$ such $C^{\mathcal{I}} \neq \emptyset$.
  - The *Knowledge Base satisfiability* problem is to check, given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model.

- The *Conjunctive Query Entailment* problem is to decide, given a $DL\text{-}Lite_{\mathcal{A}}$ KB $\mathcal{K}$ and a boolean query q over $\mathcal{K}$, whether $\mathcal{K} \models q$.
  - The *Instance Checking* problem is to decide, given an object name a, a concept B, and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, whether $\mathcal{K} \models B(a)$.

Complexity of Reasoning in $\mathcal{EL}$

Theorem

Subsumption in $\mathcal{EL}^{++}$ can be decided in polynomial time (a polytime tableax for deciding subsumption).

Complexity of Reasoning in $\mathcal{EL}$

Theorem

Subsumption in $\mathcal{EL}^{++}$ can be decided in polynomial time (a polytime tableax for deciding subsumption).

Theorem

Entailment of conjunctive queries in $\mathcal{EL}^{++}$ (already in $\mathcal{EL}^{+}$) is undecidable. ([7],[6]).

Complexity of Reasoning in $\mathcal{EL}$

Theorem

Subsumption in $\mathcal{EL}^{++}$ can be decided in polynomial time (a polytime tableax for deciding subsumption).

Theorem

Entailment of conjunctive queries in $\mathcal{EL}^{++}$ (already in $\mathcal{EL}^{+}$) is undecidable. ([7],[6]).

Theorem

Entailment of unions of conjunctive queris in $\mathcal{EL}$ is:

1. PTime*-complete with respect to data complexity;*
2. PTime*-complete with respect to KB complexity;*
3. NP*-complete with respect to combined complexity.*

Thank you
for your attention!

A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev.
The *DL-Lite* family and relations.
J. of Artificial Intelligence Research, 36:1–69, 2009.

F. Baader, S. Brandt, and C. Lutz.
Pushing the $\mathcal{EL}$ envelope.
In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, 2005.

F. Baader, C. Lutz, and B. Suntisrivaraporn.
CEL—a polynomial-time reasoner for life science ontologies.
In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer, 2006.

F. Baader, C. Lutz, and B. Suntisrivaraporn.
Efficient reasoning in $\mathcal{EL}+$.

In *Proc. of the 19th Int. Workshop on Description Logic (DL 2006)*, volume 189 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, 2006.

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):385–429, 2007.

A. Krisnadhi and C. Lutz.
Data complexity in the $\mathcal{EL}$ family of description logics.
In *Proc. of the 14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007)*, pages 333–347, 2007.

R. Rosati.
On conjunctive query answering in $\mathcal{EL}$.

In *Proc. of the 20th Int. Workshop on Description Logic (DL 2007)*, volume 250 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, 2007.