Ontology and Database Systems: Knowledge Representation and Ontologies

Part 5: Reasoning in the *DL-Lite* Family

Diego Calvanese

Faculty of Computer Science
European Master in Computational Logic

A.Y. 2015/2016

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Part 5

Reasoning in the *DL-Lite* family

Outline of Part 5

Outline of Part 5

unibz

Outline of Part 5

1. TBox reasoning
   - Preliminaries
   - Reducing to subsumption
   - Reducing to ontology unsatisfiability
2. TBox & ABox reasoning and query answering
3. Beyond *DL-Lite*

Remarks

In the following, we make some simplifying assumptions:

- We ignore the distinction between objects and values, since it is not relevant for reasoning. Hence we do not use value domains and attributes.
- We do not consider identification constraints.

Notation:

- When the distinction between *DL-Lite*$_{\mathcal{R}}$, *DL-Lite*$_{\mathcal{F}}$, or *DL-Lite*$_{\mathcal{A}}$ is not important, we use just *DL-Lite*.
- Q denotes a basic role, i.e., $Q \longrightarrow P \mid P^-$.
- R denotes a general role, i.e., $R \longrightarrow Q \mid \neg Q$.
- C denotes a general concept, i.e., $C \longrightarrow A \mid \neg A \mid \exists Q \mid \neg \exists Q$, where A is an atomic concept.

TBox Reasoning services

- **Concept Satisfiability:** C is satisfiable wrt $\mathcal{T}$, if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is not empty, i.e., $\mathcal{T} \not\models C \equiv \bot$
- **Subsumption:** C_1 is subsumed by C_2 wrt $\mathcal{T}$, if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, i.e., $\mathcal{T} \models C_1 \sqsubseteq C_2$.
- **Equivalence:** C_1 and C_2 are equivalent wrt $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$, i.e., $\mathcal{T} \models C_1 \equiv C_2$.
- **Disjointness:** C_1 and C_2 are disjoint wrt $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = \emptyset$, i.e., $\mathcal{T} \models C_1 \sqcap C_2 \equiv \bot$
- **Functionality implication:** A functionality assertion $(\textbf{funct}\ Q)$ is logically implied by $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$, we have that $(o, o_1) \in Q^{\mathcal{I}}$ and $(o, o_2) \in Q^{\mathcal{I}}$ implies $o_1 = o_2$, i.e., $\mathcal{T} \models (\textbf{funct}\ Q)$.

Analogous definitions hold for role satisfiability, subsumption, equivalence, and disjointness.

unibz

From TBox reasoning to ontology (un)satisfiability

Basic reasoning service:

- **Ontology satisfiability**: Verify whether an ontology $\mathcal{O}$ is satisfiable, i.e., whether $\mathcal{O}$ admits at least one model.

In the following, we show how to reduce TBox reasoning to ontology unsatisfiability:

1. We show how to reduce TBox reasoning services to concept/role subsumption.
2. We provide reductions from concept/role subsumption to ontology unsatisfiability.

unibz

Outline of Part 5

unibz

Concept/role satisfiability, equivalence, and disjointness

Theorem

1. C is unsatisfiable wrt $\mathcal{T}$ iff $\mathcal{T} \models C \sqsubseteq \neg C$.
2. $\mathcal{T} \models C_1 \equiv C_2$ iff $\mathcal{T} \models C_1 \sqsubseteq C_2$ and $\mathcal{T} \models C_2 \sqsubseteq C_1$.
3. C_1 and C_2 are disjoint iff $\mathcal{T} \models C_1 \sqsubseteq \neg C_2$.

Proof (sketch).

1. "$\Leftarrow$" if $\mathcal{T} \models C \sqsubseteq \neg C$, then $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, for every model $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{T}$; but this holds iff $C^{\mathcal{I}} = \emptyset$.

   "$\Rightarrow$" if C is unsatisfiable, then $C^{\mathcal{I}} = \emptyset$, for every model $\mathcal{I}$ of $\mathcal{T}$. Therefore $C^{\mathcal{I}} \subseteq (\neg C)^{\mathcal{I}}$.
2. Trivial.
3. Trivial. □

Analogous reductions for role satisfiability, equivalence and disjointness.

unibz

From implication of functionalities to subsumption

Theorem

$\mathcal{T} \models (\textbf{funct } Q)$ iff either

- $(\textbf{funct } Q) \in \mathcal{T}$ (only for *DL-Lite*$_\mathcal{F}$ or *DL-Lite*$_\mathcal{A}$), or
- $\mathcal{T} \models Q \sqsubseteq \neg Q$.

Proof (sketch).

"$\Leftarrow$" The case in which $(\textbf{funct } Q) \in \mathcal{T}$ is trivial.
Instead, if $\mathcal{T} \models Q \sqsubseteq \neg Q$, then $Q^\mathcal{I} = \emptyset$ and hence $\mathcal{I} \models (\textbf{funct } Q)$, for every model $\mathcal{I}$ of $\mathcal{T}$.

"$\Rightarrow$" When neither $(\textbf{funct } Q) \in \mathcal{T}$ nor $\mathcal{T} \models Q \sqsubseteq \neg Q$, we can construct a model of $\mathcal{T}$ that is not a model of $(\textbf{funct } Q)$. □

The interesting part of this result is the "only-if" direction, telling us that in DL-Lite functionality is implied only in trivial ways.

Outline of Part 5

From concept subsumption to ontology unsatisfiability

Theorem

$\mathcal{T} \models C_1 \sqsubseteq C_2$ iff the ontology $\mathcal{O}_{C_1 \sqsubseteq C_2} = \langle \mathcal{T} \cup \{\hat{A} \sqsubseteq C_1, \hat{A} \sqsubseteq \neg C_2\},\ \{\hat{A}(c)\}\rangle$ is unsatisfiable, where $\hat{A}$ is an atomic concept not in $\mathcal{T}$, and c is a constant.

Intuitively, C_1 is subsumed by C_2 iff the smallest ontology containing $\mathcal{T}$ and implying both $C_1(c)$ and $\neg C_2(c)$ is unsatisfiable.

Proof (sketch).

"$\Leftarrow$" Let $\mathcal{O}_{C_1 \sqsubseteq C_2}$ be unsatisfiable, and suppose that $\mathcal{T} \not\models C_1 \sqsubseteq C_2$. Then there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C_1^{\mathcal{I}} \not\subseteq C_2^{\mathcal{I}}$. Hence $C_1^{\mathcal{I}} \setminus C_2^{\mathcal{I}} \neq \emptyset$. We can extend $\mathcal{I}$ to a model of $\mathcal{O}_{C_1 \sqsubseteq C_2}$ by taking $c^{\mathcal{I}} = o$, for some $o \in C_1^{\mathcal{I}} \setminus C_2^{\mathcal{I}}$, and $\hat{A}^{\mathcal{I}} = \{c^{\mathcal{I}}\}$. This contradicts $\mathcal{O}_{C_1 \sqsubseteq C_2}$ being unsatisfiable.

"$\Rightarrow$" Let $\mathcal{T} \models C_1 \sqsubseteq C_2$, and suppose that $\mathcal{O}_{C_1 \sqsubseteq C_2}$ is satisfiable. Then there exists a model $\mathcal{I}$ be of $\mathcal{O}_{C_1 \sqsubseteq C_2}$. Then $\mathcal{I} \models \mathcal{T}$, and $\mathcal{I} \models C_1(c)$ and $\mathcal{I} \models \neg C_2(c)$, i.e., $\mathcal{I} \not\models C_1 \sqsubseteq C_2$. This contradicts $\mathcal{T} \models C_1 \sqsubseteq C_2$. □

From role subsumption to ont. unsatisfiability for *DL-Lite*$_{\mathcal{R}}$

Theorem

Let $\mathcal{T}$ be a *DL-Lite*$_{\mathcal{R}}$ TBox and R_1, R_2 two general roles.
Then $\mathcal{T} \models R_1 \sqsubseteq R_2$ iff the ontology
$\mathcal{O}_{R_1 \sqsubseteq R_2} = \langle \mathcal{T} \cup \{\hat{P} \sqsubseteq R_1, \hat{P} \sqsubseteq \neg R_2\},\ \{\hat{P}(c_1, c_2)\}\rangle$ is unsatisfiable,
where $\hat{P}$ is an atomic role not in $\mathcal{T}$, and c_1, c_2 are two constants.

Intuitively, R_1 is subsumed by R_2 iff the smallest ontology containing $\mathcal{T}$ and implying both $R_1(c_1, c_2)$ and $\neg R_2(c_1, c_2)$ is unsatisfiable.

Proof (sketch).

Analogous to the one for concept subsumption. $\square$

Notice that $\mathcal{O}_{R_1 \sqsubseteq R_2}$ *is inherently a DL-Lite*$_{\mathcal{R}}$ *ontology.*

From role subsumption to ont. unsatisfiability for $\textit{DL-Lite}_{\mathcal{F}}$

Theorem

Let $\mathcal{T}$ be a $\textit{DL-Lite}_{\mathcal{F}}$ TBox, and Q_1, Q_2 two basic roles such that $Q_1 \neq Q_2$. Then,

1. $\mathcal{T} \models Q_1 \sqsubseteq Q_2$ iff Q_1 is unsatisfiable iff
   either $\exists Q_1$ or $\exists Q_1^-$ is unsatisfiable wrt $\mathcal{T}$,
   which can again be reduced to ontology unsatisfiability.
2. $\mathcal{T} \models \neg Q_1 \sqsubseteq Q_2$ iff $\mathcal{T}$ is unsatisfiable.
3. $\mathcal{T} \models Q_1 \sqsubseteq \neg Q_2$ iff
   either $\exists Q_1$ and $\exists Q_2$ are disjoint, or $\exists Q_1^-$ and $\exists Q_2^-$ are disjoint, iff
   either $\mathcal{T} \models \exists Q_1 \sqsubseteq \neg \exists Q_2$, or $\mathcal{T} \models \exists Q_1^- \sqsubseteq \neg \exists Q_2^-$,
   which can again be reduced to ontology unsatisfiability.

Notice that an inclusion of the form $\neg Q_1 \sqsubseteq \neg Q_2$ is equivalent to $Q_2 \sqsubseteq Q_1$, and therefore is considered in the first item.

From role subsumption to ont. unsatisfiability for *DL-Lite*$_{\mathcal{A}}$

Theorem

Let $\mathcal{T}$ be a *DL-Lite*$_{\mathcal{A}}$ TBox, and Q_1, Q_2 two basic roles such that $Q_1 \neq Q_2$. Then,

1. $\mathcal{T} \models Q_1 \sqsubseteq Q_2$ iff
$\mathcal{O}_{Q_1 \sqsubseteq Q_2} = \langle \mathcal{T} \cup \{\hat{P} \sqsubseteq \neg Q_2\},\ \{Q_1(c_1, c_2), \hat{P}(c_1, c_2)\}\rangle$ is unsatisfiable,
where $\hat{P}$ is an atomic role not in $\mathcal{T}$, and c_1, c_2 are two constants.
2. $\mathcal{T} \models \neg Q_1 \sqsubseteq Q_2$ iff
$\mathcal{O}_{\neg Q_1 \sqsubseteq Q_2} = \langle \mathcal{T} \cup \{\hat{P} \sqsubseteq \neg Q_1, \hat{P} \sqsubseteq \neg Q_2\},\ \{\hat{P}(c_1, c_2)\}\rangle$ is unsatisfiable,
where $\hat{P}$ is an atomic role not in $\mathcal{T}$, and c_1, c_2 are two constants.
3. $\mathcal{T} \models Q_1 \sqsubseteq \neg Q_2$ iff
$\mathcal{O}_{Q_1 \sqsubseteq \neg Q_2} = \langle \mathcal{T},\ \{Q_1(c_1, c_2), Q_2(c_1, c_2)\}\rangle$ is unsatisfiable,
where c_1, c_2 are two constants.

Notice that an inclusion of the form $\neg Q_1 \sqsubseteq \neg Q_2$ is equivalent to $Q_2 \sqsubseteq Q_1$, and therefore is considered in the first item.

Summary

- The results above tell us that we can support TBox reasoning services by relying on the ontology (un)satisfiability service.
- Ontology satisfiability is a form of reasoning over both the TBox and the ABox of the ontology.
- In the following, we first consider other TBox & ABox reasoning services, in particular **query answering**, and then turn back to ontology satisfiability.

Outline of Part 5

1. TBox reasoning

2. TBox & ABox reasoning and query answering
    - TBox & ABox Reasoning services
    - Query answering
    - Query answering over satisfiable ontologies
    - Ontology satisfiability
    - Complexity of reasoning in *DL-Lite*

3. Beyond *DL-Lite*

Outline of Part 5

1. TBox reasoning
2. TBox & ABox reasoning and query answering
   - TBox & ABox Reasoning services
   - Query answering
   - Query answering over satisfiable ontologies
   - Ontology satisfiability
   - Complexity of reasoning in *DL-Lite*
3. Beyond *DL-Lite*

TBox and ABox reasoning services

- **Ontology Satisfiability:** Verify whether an ontology $\mathcal{O}$ is satisfiable, i.e., whether $\mathcal{O}$ admits at least one model.
- **Concept Instance Checking:** Verify whether an individual c is an instance of a concept C in an ontology $\mathcal{O}$, i.e., whether $\mathcal{O} \models C(c)$.
- **Role Instance Checking:** Verify whether a pair (c_1, c_2) of individuals is an instance of a role R in an ontology $\mathcal{O}$, i.e., whether $\mathcal{O} \models R(c_1, c_2)$.
- **Query Answering** Given a query q over an ontology $\mathcal{O}$, find all tuples $\vec{c}$ of constants such that $\mathcal{O} \models q(\vec{c})$.

unibz

Query answering and instance checking

For atomic concepts and roles, **instance checking is a special case of query answering**, in which the query is **boolean** and constituted by a single atom in the body.

- $\mathcal{O} \models A(c)$ iff $q() \leftarrow A(c)$ evaluated over $\mathcal{O}$ is true.
- $\mathcal{O} \models P(c_1, c_2)$ iff $q() \leftarrow P(c_1, c_2)$ evaluated over $\mathcal{O}$ is true.

unibz

From instance checking to ontology unsatisfiability

Theorem

Let $\mathcal{O} = \langle\mathcal{T},\mathcal{A}\rangle$ be a *DL-Lite* ontology, C a *DL-Lite* concept, and P an atomic role. Then:

- $\mathcal{O} \models C(c)$ iff $\mathcal{O}_{C(c)} = \langle\mathcal{T} \cup \{\hat{A} \sqsubseteq \neg C\},\ \mathcal{A} \cup \{\hat{A}(c)\}\rangle$ is unsatisfiable, where $\hat{A}$ is an atomic concept not in $\mathcal{O}$.
- $\mathcal{O} \models \neg P(c_1, c_2)$ iff $\mathcal{O}_{\neg P(c_1,c_2)} = \langle\mathcal{T},\ \mathcal{A} \cup \{P(c_1, c_2)\}\rangle$ is unsatisfiable.

Theorem

Let $\mathcal{O} = \langle\mathcal{T},\mathcal{A}\rangle$ be a *DL-Lite*$_\mathcal{F}$ ontology and P an atomic role.
Then $\mathcal{O} \models P(c_1, c_2)$ iff $\mathcal{O}$ is unsatisfiable or $P(c_1, c_2) \in \mathcal{A}$.

Theorem

Let $\mathcal{O} = \langle\mathcal{T},\mathcal{A}\rangle$ be a *DL-Lite*$_\mathcal{R}$ or *DL-Lite*$_\mathcal{A}$ ontology and P an atomic role.
Then $\mathcal{O} \models P(c_1, c_2)$ iff $\mathcal{O}_{P(c_1,c_2)} = \langle\mathcal{T} \cup \{\hat{P} \sqsubseteq \neg P\},\ \mathcal{A} \cup \{\hat{P}(c_1, c_2)\}\rangle$ is unsatisfiable, where $\hat{P}$ is an atomic role not in $\mathcal{O}$.

Outline of Part 5

1. TBox reasoning

2. TBox & ABox reasoning and query answering
   - TBox & ABox Reasoning services
   - Query answering
   - Query answering over satisfiable ontologies
   - Ontology satisfiability
   - Complexity of reasoning in *DL-Lite*

3. Beyond *DL-Lite*

Certain answers

We recall that

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

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

A.k.a. **certain answers** in databases, i.e., the tuples that are answers to q in **all** models of $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

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

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 $\Delta^{\mathcal{I}}$.

$\mathcal{Q}$-rewritability for *DL-Lite*

- We now study rewritability of query answering over *DL-Lite* ontologies.
- In particular we will show that $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ (and hence $\mathit{DL\text{-}Lite}_{\mathcal{F}}$ and $\mathit{DL\text{-}Lite}_{\mathcal{R}}$) enjoy FOL-rewritability of answering union of conjunctive queries.

Query answering vs. ontology satisfiability

- In the case in which an ontology is unsatisfiable, according to the "ex falso quod libet" principle, reasoning is trivialized.
- In particular, query answering is meaningless, since every tuple is in the answer to every query.
- We are not interested in encoding meaningless query answering into the perfect reformulation of the input query. Therefore, before query answering, we will always check ontology satisfiability to single out meaningful cases.

Thus, we proceed as follows:

1. We show how to do **query answering over satisfiable ontologies**.
2. We show how we can exploit the query answering algorithm also to check ontology satisfiability.

unibz

Positive vs. negative inclusions

We call positive inclusions (PIs) assertions of the form

$$A_1 \sqsubseteq A_2 \qquad \exists Q_1 \sqsubseteq A_2 \qquad Q_1 \sqsubseteq Q_2$$
$$A_1 \sqsubseteq \exists Q_2 \qquad \exists Q_1 \sqsubseteq \exists Q_2$$

We call negative inclusions (NIs) assertions of the form

$$A_1 \sqsubseteq \neg A_2 \qquad \exists Q_1 \sqsubseteq \neg A_2 \qquad Q_1 \sqsubseteq \neg Q_2$$
$$A_1 \sqsubseteq \neg \exists Q_2 \qquad \exists Q_1 \sqsubseteq \neg \exists Q_2$$

Outline of Part 5

unibz

Query answering over satisfiable ontologies

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

1. Using $\mathcal{T}$, **rewrite** q into a UCQ $r_{q,\mathcal{T}}$ (the perfect rewriting of q w.r.t. $\mathcal{T}$).
2. **Evaluate** $r_{q,\mathcal{T}}$ over $\mathcal{A}$ (simply viewed as data), to return $cert(q, \mathcal{O})$.

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

unibz

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

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

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

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

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

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

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

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

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

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

Query rewriting – Constants

Conversely, for the query $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, \mathtt{fl})$

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

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

$\mathsf{teaches}(x, \mathtt{fl})$ does not unify with $\mathsf{teaches}(z, f(z))$, since the **skolem term** $f(z)$ in the head of the rule **does not unify** with the constant $\mathtt{fl}$.
Remember: We adopt the **unique name assumption**.

In this case, we say that the PI does not apply to the atom $\mathsf{teaches}(x, \mathtt{fl})$.

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:

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

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

and the PI $\mathsf{Professor} \sqsubseteq \exists\mathsf{teaches}$

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

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

Query rewriting – Reduce step

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

and the PI $\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches}$

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

This PI does not apply to $\mathsf{teaches}(x, y)$ or $\mathsf{teaches}(z, y)$, 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 $\mathsf{teaches}(x, y)$ and $\mathsf{teaches}(z, y)$. This rewriting step is called **reduce**, and produces the query

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

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

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

unibz

Query rewriting – Summary

Reformulate the CQ q into a set of queries:

- Apply to q and the computed queries in all possible ways the PIs in $\mathcal{T}$:

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

  ('$\_$' 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.

- 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.

The UCQ resulting from this process is the **perfect rewriting** $r_{q,\mathcal{T}}$.

Query rewriting algorithm

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** $PR := PR \cup \{\, \textit{ApplyPI}(q, g, I) \,\}$;
    for each g_1, g_2 in q **do**
      if g_1 and g_2 unify **then** $PR := PR \cup \{\tau(\textit{Reduce}(q, g_1, g_2))\}$;
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: $\mathsf{q}(x) \leftarrow \mathsf{teaches}(x, y), \mathsf{Course}(y)$

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

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

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

unibz

Query answering in *DL-Lite* – An interesting example

TBox: $\mathsf{Person} \sqsubseteq \exists\mathsf{hasFather}$
$\exists\mathsf{hasFather}^- \sqsubseteq \mathsf{Person}$

ABox: $\mathsf{Person}(\mathtt{mary})$

Query: $\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, y_3)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, \_)$

$\Downarrow$ **Apply** $\mathsf{Person} \sqsubseteq \exists\mathsf{hasFather}$ to the atom $\mathsf{hasFather}(y_2, \_)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{Person}(y_2)$

$\Downarrow$ **Apply** $\exists\mathsf{hasFather}^- \sqsubseteq \mathsf{Person}$ to the atom $\mathsf{Person}(y_2)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(\_, y_2)$

$\Downarrow$ **Unify** atoms $\mathsf{hasFather}(y_1, y_2)$ and $\mathsf{hasFather}(\_, y_2)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2)$

$\Downarrow$

$\cdots$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, \_)$

$\Downarrow$ **Apply** $\mathsf{Person} \sqsubseteq \exists\mathsf{hasFather}$ to the atom $\mathsf{hasFather}(x, \_)$

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

unibz

Query answering over satisfiable *DL-Lite* ontologies

For an ABox $\mathcal{A}$ and a query q over $\mathcal{A}$, let $\mathit{Eval}_{\mathrm{CWA}}(q, \mathcal{A})$ denote the evaluation of q over $\mathcal{A}$ considered as a database (i.e., considered under the CWA).

Theorem

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

$$cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathit{Eval}_{\mathrm{CWA}}(\mathit{PerfectRef}(q, \mathcal{T}_P), \mathcal{A}).$$

As a consequence, query answering over a satisfiable *DL-Lite* ontology is FOL-rewritable.

Notice that we did not use NIs or functionality assertions of $\mathcal{T}$ in computing $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle$. Indeed, **when the ontology is satisfiable, we can ignore NIs and functionality assertions for query answering**.

Canonical model of a *DL-Lite* ontology

The proof of the previous result exploits a fundamental property of *DL-Lite*, that relies on the following notion.

Def.: Canonical model

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite* ontology. A model $\mathcal{I}_{\mathcal{O}}$ of $\mathcal{O}$ is called **canonical** if for every model $\mathcal{I}$ of $\mathcal{O}$ there is a homomorphism from $\mathcal{I}_{\mathcal{O}}$ to $\mathcal{I}$.

Theorem

Every satisfiable *DL-Lite* ontology has a **canonical model**.

Properties of the canonical models of a *DL-Lite* ontology:

- A canonical model is in general infinite.
- All canonical models are homomorphically equivalent, hence we can do as if there was a single canonical model.

Query answering in *DL-Lite* – Canonical model

From the definition of canonical model, and since homomorphisms are closed under composition, we get that:

To compute the certain answer to a query q over an ontology $\mathcal{O}$, one could in principle evaluate q over a canonical model $\mathcal{I}_\mathcal{O}$ of $\mathcal{O}$.

- This does not give us directly an algorithm for query answering over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, since $\mathcal{I}_\mathcal{O}$ may be infinite.
- However, one can show that evaluating q over $\mathcal{I}_\mathcal{O}$ amounts to evaluating the perfect rewriting $r_{q,\mathcal{T}}$ over $\mathcal{A}$.

Using RDBMS technology for query answering

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

- 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 have that query answering over satisfiable *DL-Lite* ontologies can be done effectively using RDBMS technology:

$$cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathit{Eval}(\mathit{SQL}(\mathit{PerfectRef}(q, \mathcal{T}_P)), \mathit{DB}(\mathcal{A}))$$

Where:

– $\mathit{Eval}(q_s, \mathit{DB})$ denotes the evaluation of an SQL query q_s over a database DB.
– $\mathit{SQL}(q)$ denotes the SQL encoding of a UCQ q.
– $\mathit{DB}(\mathcal{A})$ denotes the database obtained as above.

Outline of Part 5

unibz

Satisfiability of ontologies with only PIs

Let us now consider the problem of establishing whether an ontology is satisfiable.

A first notable result tells us that PIs alone cannot generate ontology unsatisfiability.

Theorem

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite* ontology where $\mathcal{T}$ contains **only PIs**.
Then, $\mathcal{O}$ is satisfiable.

Satisfiability of *DL-Lite*$_{\mathcal{A}}$ ontologies

Unsatisfiability in *DL-Lite*$_{\mathcal{A}}$ ontologies can be caused by **NIs** or by **functionality assertions**.

Example

TBox $\mathcal{T}$: $\text{Professor} \sqsubseteq \neg\text{Student}$
$\exists\text{teaches} \sqsubseteq \text{Professor}$
$(\textbf{funct}\ \text{teaches}^-)$

ABox $\mathcal{A}$: $\text{Student}(\texttt{john})$
$\text{teaches}(\texttt{john}, \texttt{fl})$
$\text{teaches}(\texttt{michael}, \texttt{fl})$

unibz

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

Satisfiability of a *DL-Lite*$_{\mathcal{A}}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is reduced to evaluating over *DB*$(\mathcal{A})$ a UCQ that asks for the **existence of objects violating the NI and functionality assertions**.

Let $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$.
We deal with NIs and functionality assertions differently.

For each NI $N \in \mathcal{T}$:

1. we construct a boolean CQ $q_N()$ such that

$$\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N() \quad \text{iff} \quad \langle \mathcal{T}_P \cup \{N\}, \mathcal{A} \rangle \text{ is unsatisfiable}$$

2. We check whether $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$ using *PerfectRef*, i.e., we compute *PerfectRef*$(q_N, \mathcal{T}_P)$, and evaluate it over *DB*$(\mathcal{A})$.

For each functionality assertion $F \in \mathcal{T}$:

1. we construct a boolean CQ $q_F()$ such that

$$\mathcal{A} \models q_F() \quad \text{iff} \quad \langle \{F\}, \mathcal{A} \rangle \text{ is unsatisfiable.}$$

2. We check whether $\mathcal{A} \models q_F()$, by simply evaluating q_F over *DB*$(\mathcal{A})$.

Checking violations of negative inclusions

For each **NI** N in $\mathcal{T}$ we compute a boolean CQ $q_N()$ according to the following rules:

$$
\begin{array}{rcl}
A_1 \sqsubseteq \neg A_2 & \rightsquigarrow & q_N() \leftarrow A_1(x), A_2(x) \\
\exists P \sqsubseteq \neg A \ \text{ or } \ A \sqsubseteq \neg \exists P & \rightsquigarrow & q_N() \leftarrow P(x,y), A(x) \\
\exists P^- \sqsubseteq \neg A \ \text{ or } \ A \sqsubseteq \neg \exists P^- & \rightsquigarrow & q_N() \leftarrow P(y,x), A(x) \\
\exists P_1 \sqsubseteq \neg \exists P_2 & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(x,z) \\
\exists P_1 \sqsubseteq \neg \exists P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(z,x) \\
\exists P_1^- \sqsubseteq \neg \exists P_2 & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(y,z) \\
\exists P_1^- \sqsubseteq \neg \exists P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(z,y) \\
P_1 \sqsubseteq \neg P_2 \ \text{ or } \ P_1^- \sqsubseteq \neg P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(x,y) \\
P_1^- \sqsubseteq \neg P_2 \ \text{ or } \ P_1 \sqsubseteq \neg P_2^- & \rightsquigarrow & q_N() \leftarrow P_1(x,y), P_2(y,x)
\end{array}
$$

unibz

Checking violations of negative inclusions – Example

PIs $\mathcal{T}_P$: $\exists$teaches $\sqsubseteq$ Professor
NIs N : Professor $\sqsubseteq$ $\neg$Student

Query q_N: $q_N() \leftarrow \text{Student}(x), \text{Professor}(x)$

Perfect Rewriting: $q_N() \leftarrow \text{Student}(x), \text{Professor}(x)$
$q_N() \leftarrow \text{Student}(x), \text{teaches}(x, \_)$

ABox $\mathcal{A}$: $\text{teaches}(\texttt{john}, \texttt{fl})$
$\text{Student}(\texttt{john})$

It is easy to see that $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$, and that the ontology $\langle \mathcal{T}_P \cup \{\text{Professor} \sqsubseteq \neg\text{Student}\},\ \mathcal{A} \rangle$ **is unsatisfiable**.

unibz

Boolean queries vs. non-boolean queries for NIs

To ensure correctness of the method, the queries used to check for the violation of a NI need to be **boolean**.

Example

TBox $\mathcal{T}$: $A_1 \sqsubseteq \neg A_0$ $\quad$ $\exists P \sqsubseteq A_1$
$A_1 \sqsubseteq A_0$ $\quad$ $A_2 \sqsubseteq \exists P^-$

ABox $\mathcal{A}$: $A_2(c)$

Since A_1, P, and A_2 are unsatisfiable, also $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable.

Consider the query corresponding to the NI $A_1 \sqsubseteq \neg A_0$.

$q_N() \leftarrow A_1(x), A_0(x)$

Then *PerfectRef*($q_N, \mathcal{T}_P$) is:

$q_N() \leftarrow A_1(x), A_0(x)$
$q_N() \leftarrow A_1(x)$
$q_N() \leftarrow P(x, \_)$
$q_N() \leftarrow A_2(\_)$

We have that $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$.

$q'_N(x) \leftarrow A_1(x), A_0(x)$

Then *PerfectRef*($q'_N, \mathcal{T}_P$) is

$q'_N(x) \leftarrow A_1(x), A_0(x)$
$q'_N(x) \leftarrow A_1(x)$
$q'_N(x) \leftarrow P(x, \_)$

$cert(q'_N, \langle \mathcal{T}_P, \mathcal{A} \rangle) = \emptyset$, hence $q'_N(x)$ does not detect unsatisfiability.

Checking violations of functionality assertions

For each **functionality assertion** F in $\mathcal{T}$ we compute a boolean FOL query $q_F()$ according to the following rules:

$$\begin{array}{lcl} (\textbf{funct } P) & \rightsquigarrow & q_F() \leftarrow P(x,y), P(x,z), y \neq z \\ (\textbf{funct } P^-) & \rightsquigarrow & q_F() \leftarrow P(x,y), P(z,y), x \neq z \end{array}$$

Example

Functionality F: $(\textbf{funct teaches}^-)$

Query q_F: $q_F() \leftarrow \textbf{teaches}(x,y), \textbf{teaches}(z,y), x \neq z$

ABox $\mathcal{A}$: $\mathsf{teaches}(\mathtt{john}, \mathtt{fl})$
$\mathsf{teaches}(\mathtt{michael}, \mathtt{fl})$

It is easy to see that $\mathcal{A} \models q_F()$, and that $\langle\{(\textbf{funct teaches}^-)\}, \mathcal{A}\rangle$, **is unsatisfiable**.

unibz

From satisfiability to query answering in $\textit{DL-Lite}_{\mathcal{A}}$

Lemma (Separation for $\textit{DL-Lite}_{\mathcal{A}}$)

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\textit{DL-Lite}_{\mathcal{A}}$ ontology, and $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$.
Then, $\mathcal{O}$ is unsatisfiable iff one of the following condition holds:

(a) There exists a NI $N \in \mathcal{T}$ such that $\langle \mathcal{T}_P, \mathcal{A} \rangle \models q_N()$.

(b) There exists a functionality assertion $F \in \mathcal{T}$ such that $\mathcal{A} \models q_F()$.

(a) relies on the properties that **NIs do not interact with each other**, and that **interaction between NIs and PIs** is captured **through** *PerfectRef*.

(b) exploits the property that **NIs and PIs do not interact with functionalities**: indeed, no functionality assertion is contradicted in a $\textit{DL-Lite}_{\mathcal{A}}$ ontology $\mathcal{O}$, beyond those explicitly contradicted by the ABox.

Notably, to check ontology satisfiability, each NI and each functionality assertion can be processed individually.

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

From the previous lemma and the theorem on query answering for satisfiable $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ ontologies, we get the following result.

Theorem

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ ontology, and $\mathcal{T}_P$ the set of PIs in $\mathcal{T}$.
Then, $\mathcal{O}$ is unsatisfiable iff one of the following condition holds:

(a) There exists a NI $N \in \mathcal{T}$ s.t. $\mathit{Eval}_{\mathrm{CWA}}(\mathit{PerfectRef}(q_N, \mathcal{T}_P), \mathcal{A})$ returns *true*.

(b) There exists a func. assertion $F \in \mathcal{T}$ s.t. $\mathit{Eval}_{\mathrm{CWA}}(q_F, \mathcal{A})$ returns *true*.

Note: All the queries $q_N()$ and $q_F()$ can be combined into a single UCQ.
Hence, satisfiability of a $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ ontology is reduced to evaluating a FOL-query over an ontology whose TBox consists of positive inclusions only (and hence is satisfiable).

Outline of Part 5

unibz

Complexity of query answering over satisfiable ontologies

Theorem

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

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

Proof (sketch).

1. **Guess** together the derivation of one of the CQs of the perfect rewriting, and an assignment to its existential variables. Checking the derivation and evaluating the guessed CQ over the ABox is then polynomial in combined complexity. NP-hardness follows from combined complexity of evaluating CQs over a database.
2. The number of CQs in the perfect rewriting is polynomial in the size of the TBox, and we can compute them in PTIME.
3. AC^0 is the data complexity of evaluating FOL queries over a DB. □

Complexity of ontology satisfiability

Theorem

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

1. $\textsc{PTime}$ in the size of the **ontology** (combined complexity).
2. AC^0 in the size of the **ABox** (data complexity).

Proof (sketch).

We observe that all the queries $q_N()$ and $q_F()$ checking for violations of negative inclusions N and functionality assertions F can be combined into a single UCQ whose size is linear in the TBox, and does not depend on the ABox. Hence, the result follows directly from the complexity of query answering over satisfiable ontologies. □

Complexity of TBox reasoning

Theorem

TBox reasoning over *DL-Lite*$_{\mathcal{A}}$ ontologies is $\text{PT}\textsc{ime}$ in the size of the **TBox** (schema complexity).

Proof (sketch).

Follows from the previous theorem, and from the fact that all TBox reasoning tasks can be reduced to ontology satisfiability.
Indeed, the size of the ontology constructed in the reduction is polynomial in the size of the input TBox. □

Outline of Part 5

Outline of Part 5

unibz

Beyond *DL-Lite*

We consider now DL languages that **extend DL-Lite with additional DL constructs** or with combinations of constructs that are not legal in *DL-Lite*.

We show that (essentially) all such extensions of *DL-Lite* make it lose its nice computational properties.

Specifically, we consider the following DL constructs:

Construct	Syntax	Example	Semantics
conjunction	$C_1 \sqcap C_2$	Doctor ⊓ Male	$C_1^\mathcal{I} \cap C_2^\mathcal{I}$
disjunction	$C_1 \sqcup C_2$	Doctor ⊔ Lawyer	$C_1^\mathcal{I} \cup C_2^\mathcal{I}$
qual. exist. restr.	$\exists Q.C$	∃child.Male	$\{a \mid \exists b.\, (a,b) \in Q^\mathcal{I} \land b \in C^\mathcal{I}\}$
qual. univ. restr.	$\forall Q.C$	∀child.Male	$\{a \mid \forall b.\, (a,b) \in Q^\mathcal{I} \to b \in C^\mathcal{I}\}$

Beyond $\textit{DL-Lite}_{\mathcal{A}}$: results on data complexity

	Lhs	Rhs	Funct.	Role incl.	Data complexity of query answering
0	$\textit{DL-Lite}_{\mathcal{A}}$		$\surd^*$	$\surd^*$	in AC^0
1	$A \mid \exists P.A$	A	–	–	NLogSpace-hard
2	A	$A \mid \forall P.A$	–	–	NLogSpace-hard
3	A	$A \mid \exists P.A$	$\surd$	–	NLogSpace-hard
4	$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTime-hard
5	$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTime-hard
6	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	$\surd$	–	PTime-hard
7	$A \mid \exists P.A \mid \exists P^-.A$	$A \mid \exists P$	–	–	PTime-hard
8	$A \mid \exists P \mid \exists P^-$	$A \mid \exists P \mid \exists P^-$	$\surd$	$\surd$	PTime-hard
9	$A \mid \neg A$	A	–	–	coNP-**hard**
10	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-**hard**
11	$A \mid \forall P.A$	A	–	–	coNP-**hard**

Notes:

- * with the "proviso" of not specializing functional properties.
- NLogSpace and PTime hardness holds already for instance checking.
- For coNP-hardness in line 10, a TBox with a single assertion $A_L \sqsubseteq A_T \sqcup A_F$ suffices! $\leadsto$ **No** hope of including **covering constraints**.

unibz

Observations

- *DL-Lite*-family is FOL-rewritable, hence AC^0 – holds also with n-ary relations $\rightsquigarrow$ *DLR-Lite*$_{\mathcal{F}}$ and *DLR-Lite*$_{\mathcal{R}}$.
- RDFS is a subset of *DL-Lite*$_{\mathcal{R}}$ $\rightsquigarrow$ is FOL-rewritable, hence AC^0.
- Horn-$\mathcal{SHIQ}$ [Hustadt *et al.*, 2005] is $\mathrm{PT{\small IME}}$**-hard** even for instance checking (line 8).
- DLP [Grosof *et al.*, 2003] is $\mathrm{PT{\small IME}}$**-hard** (line 4)
- $\mathcal{EL}$ [Baader *et al.*, 2005] is $\mathrm{PT{\small IME}}$**-hard** (line 4).
- Although used in ER and UML, no hope of including **covering constraints**, since we get coNP-hardness for trivial DLs (line 10).

Outline of Part 5

1. TBox reasoning
2. TBox & ABox reasoning and query answering
3. Beyond *DL-Lite*
   - Data complexity of query answering in DLs beyond *DL-Lite*
   - NLogSpace-hard DLs
   - PTime-hard DLs
   - coNP-hard DLs
   - Combining functionality and role inclusions
   - Unique name assumption

unibz

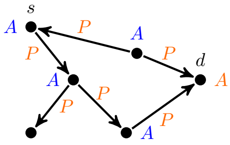
Qualified existential quantification in the lhs of inclusions

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

	Lhs	Rhs	$\mathcal{F}$	$\mathcal{R}$	Data complexity
1	$A \mid \exists P.A$	A	–	–	NLOGSPACE-hard

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

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



Result:
$\langle \mathcal{T}, \mathcal{A} \rangle \models A(s)$ iff d is reachable from s in the graph.

Note: Since the reduction has to show hardness in data complexity, the graph must be encoded in the ABox (while the TBox has to be fixed).

unibz

NLOGSPACE-hard cases

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

	Lhs	Rhs	$\mathcal{F}$	$\mathcal{R}$	Data complexity
1	$A \mid \exists P.A$	A	–	–	NLOGSPACE-hard

By reduction from reachability in directed graphs.

2	A	$A \mid \forall P.A$	–	–	NLOGSPACE-hard

Follows from 1 by replacing $\exists P.A_1 \sqsubseteq A_2$ with $A_1 \sqsubseteq \forall P^-.A_2$, and by replacing each occurrence of P^- with P', for a new role P'.

3	A	$A \mid \exists P.A$	√	–	NLOGSPACE-hard

Proved by simulating in the reduction $\exists P.A_1 \sqsubseteq A_2$ via $A_1 \sqsubseteq \exists P^-.A_2$ and $(\textbf{funct}\ P^-)$, and by replacing again each occurrence of P^- with P', for a new role P'.

unibz

Outline of Part 5

unibz

Path System Accessibility

To show PTIME-hardness, we use a reduction from a PTIME-complete problem. We use 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

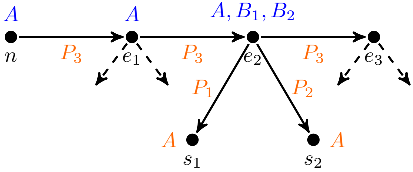
Given an instance PS of Path System Accessibility, deciding whether t is accessible, is **PTIME-complete**.

unibz

Reduction from Path System Accessibility

- Given an instance $PS = (N, E, S, t)$, we construct an ABox $\mathcal{A}$ that:
  - encodes the accessibility relation using three roles P_1, P_2, and P_3, and
  - asserts $A(s)$ for each source node $s \in S$.

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



- We construct a TBox $\mathcal{T}$ consisting of the inclusion assertions:

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

Result:

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

unibz

Outline of Part 5

coNP-hard cases

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

Query answering is coNP-hard in data complexity for:

	Lhs	Rhs	$\mathcal{F}$	$\mathcal{R}$	Data complexity
9	$A \mid \neg A$	A	–	–	coNP-hard
10	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-hard
11	$A \mid \forall P.A$	A	–	–	coNP-hard

All three cases are proved by adapting the proof of coNP-hardness of instance checking for $\mathcal{ALE}$ by [Donini *et al.*, 1994].

unibz

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: $\varphi = c_1 \land c_2 \land c_3$, with

$$
\begin{array}{lcl}
c_1 & = & v_1 \lor v_2 \lor \neg v_3 \lor \neg v_4 \\
c_2 & = & \textit{false} \lor \textit{false} \lor \neg v_1 \lor \neg v_4 \\
c_3 & = & \textit{false} \lor v_4 \lor \neg \textit{true} \lor \neg v_2
\end{array}
$$

2+2-SAT is NP-complete [Donini *et al.*, 1994].

Reduction from 2+2-SAT

We construct a TBox $\mathcal{T}$ and a query $q()$ over concepts L, T, F and roles P_1, P_2, N_1, N_2.

- TBox $\mathcal{T} = \{ \ L \sqsubseteq T \sqcup F \ \}$
- $q() \leftarrow P_1(c, v_1), P_2(c, v_2), N_1(c, v_3), N_2(c, v_4),$
  $F(v_1), F(v_2), T(v_3), T(v_4)$

Given a 2+2-CNF formula $\varphi = c_1 \wedge \cdots \wedge c_k$ over vars $v_1, \ldots, v_n$, *true*, *false*, we construct an ABox $\mathcal{A}_\varphi$ using individuals $\mathtt{c}_1, \ldots \mathtt{c}_k$, $\mathtt{v}_1, \ldots, \mathtt{v}_n$, $\mathtt{true}$, $\mathtt{false}$:

- for each propositional variable v_i: $L(\mathtt{v}_i)$
- for each clause $c_j = v_{j_1} \vee v_{j_2} \vee \neg v_{j_3} \vee \neg v_{j_4}$:
  $P_1(\mathtt{c}_j, \mathtt{v}_{j_1})$, $P_2(\mathtt{c}_j, \mathtt{v}_{j_2})$, $N_1(\mathtt{c}_j, \mathtt{v}_{j_3})$, $N_2(\mathtt{c}_j, \mathtt{v}_{j_4})$
- $T(\mathtt{true})$, $F(\mathtt{false})$

Note: the TBox $\mathcal{T}$ and the query q do not depend on φ, hence this reduction works for data complexity.

Reduction from 2+2-SAT (cont'd)

Lemma

$\langle \mathcal{T}, A_\varphi \rangle \not\models q()$ iff φ is satisfiable.

Proof (sketch).

"$\Rightarrow$" If $\langle \mathcal{T}, A_\varphi \rangle \not\models q()$, then there is a model $\mathcal{I}$ of $\langle \mathcal{T}, A_\varphi \rangle$ s.t. $\mathcal{I} \not\models q()$. We define a truth assignment $\alpha_\mathcal{I}$ by setting $\alpha_\mathcal{I}(v_i) = \mathit{true}$ iff $\mathrm{v}_i^\mathcal{I} \in T^\mathcal{I}$. Notice that, since $L \sqsubseteq T \sqcup F$, if $\mathrm{v}_i^\mathcal{I} \notin T^\mathcal{I}$, then $\mathrm{v}_i^\mathcal{I} \in F^\mathcal{I}$.
It is easy to see that, since $q()$ asks for a false clause and $\mathcal{I} \not\models q()$, for each clause c_j, one of the literals in c_j evaluates to *true* in $\alpha_\mathcal{I}$.
"$\Leftarrow$" From a truth assignment α that satisfies φ, we construct an interpretation $\mathcal{I}_\alpha$ with $\Delta^{\mathcal{I}_\alpha} = \{c_1, \ldots, c_k, v_1, \ldots, v_n, t, f\}$, and:

- $\mathrm{c}_j^{\mathcal{I}_\alpha} = c_j$, $\mathrm{v}_i^{\mathcal{I}_\alpha} = v_i$, $\mathtt{true}^{\mathcal{I}_\alpha} = t$, $\mathtt{false}^{\mathcal{I}_\alpha} = f$
- $T^{\mathcal{I}_\alpha} = \{v_i \mid \alpha(v_i) = \mathit{true}\} \cup \{t\}$, $F^{\mathcal{I}_\alpha} = \{v_i \mid \alpha(v_i) = \mathit{false}\} \cup \{f\}$

It is easy to see that $\mathcal{I}_\alpha$ is a model of $\langle \mathcal{T}, A_\varphi \rangle$ and that $\mathcal{I}_\alpha \not\models q()$. $\square$

Outline of Part 5

unibz

Combining functionalities and role inclusions

Let *DL-Lite*$_{\mathcal{FR}}$ be the DL that is the union of *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, i.e., the *DL-Lite* logic that allows for using both role functionality and role inclusions without any restrictions.

Due to the unrestricted interaction of functionality and role inclusions *DL-Lite*$_{\mathcal{FR}}$ is significantly more complicated than the logics of the *DL-Lite* family:

- One can force the unification of existentially implied objects (i.e., separation does not hold anymore).
- Additional constructs besides those present in *DL-Lite* can be simulated.
- The computational complexity of reasoning increases significantly.

Unification of existentially implied objects – Example

TBox $\mathcal{T}$: $A \sqsubseteq \exists P$ $\quad P \sqsubseteq S$
$\exists P^- \sqsubseteq A$ $\quad (\textbf{funct } S)$

ABox $\mathcal{A}$: $A(c_1),\ S(c_1, c_2),\ S(c_2, c_3),\ \ldots,\ S(c_{n-1}, c_n)$

$$
\begin{array}{rrcl}
A(c_1), & A \sqsubseteq \exists P & \models & P(c_1, x),\ \text{for some } x \\
P(c_1, x), & P \sqsubseteq S & \models & S(c_1, x) \\
S(c_1, x), \quad S(c_1, c_2), & (\textbf{funct } S) & \models & x = c_2 \\
P(c_1, c_2), & \exists P^- \sqsubseteq A & \models & A(c_2) \\
A(c_2), & A \sqsubseteq \exists P & \ldots & \\
 & & \models & A(c_n)
\end{array}
$$

Hence, we get:

- If we add $B(c_n)$ and $B \sqsubseteq \neg A$, the ontology becomes inconsistent.
- Similarly, the answer to the following query over $\langle \mathcal{T}, \mathcal{A} \rangle$ is *true*:

$$q() \leftarrow A(z_1), S(z_1, z_2), S(z_2, z_3), \ldots, S(z_{n-1}, z_n), A(z_n)$$

Unification of existentially implied objects

Note: The number of unification steps above **depends on the data**. Hence this kind of deduction cannot be mimicked by a FOL (or SQL) query, since it requires a form of **recursion**. As a consequence, we get:

Combining functionality and role inclusions is problematic.

It breaks **separability**, i.e., functionality assertions may force existentially quantified objects to be unified with existing objects.

Note: the problems are caused by the **interaction** among:

- an inclusion $P \sqsubseteq S$ between roles,
- a functionality assertion $(\textbf{funct } S)$ on the super-role, and
- a cycle of concept inclusion assertions $A \sqsubseteq \exists P$ and $\exists P^- \sqsubseteq A$.

Simulation of constructs using funct. and role inclusions

In fact, by exploiting the interaction between functionality and role inclusions, we can simulate typical DL constructs not present in *DL-Lite*:

- Simulation of $A \sqsubseteq \exists R.C$: (*Note:* this does not require functionality)

$$A \sqsubseteq \exists R_C \qquad R_C \sqsubseteq R \qquad \exists R_C^- \sqsubseteq C$$

- Simulation of $A_1 \sqcap A_2 \sqsubseteq C$:

$$\begin{array}{rclrcll} A_1 & \sqsubseteq & \exists R_1 & A_2 & \sqsubseteq & \exists R_2 & \\ R_1 & \sqsubseteq & R_{12} & R_2 & \sqsubseteq & R_{12} & (\textbf{funct } R_{12}) \\ \exists R_1^- & \sqsubseteq & \exists R_3^- & & & & \\ \exists R_3 & \sqsubseteq & C & & & & \\ R_3 & \sqsubseteq & R_{23} & R_2 & \sqsubseteq & R_{23} & (\textbf{funct } R_{23}^-) \end{array}$$

unibz

Simulation of constructs (cont'd)

Simulation of $A \sqsubseteq \forall R.C$:

We use **reification** of roles: $\bullet \xrightarrow{R} \bullet$ $\quad$ $\bullet \xleftarrow{S_1} \bullet \xrightarrow{S_2} \bullet$

$$
\begin{array}{lll}
S_{1,C} \sqsubseteq S_1 & S_{1,\neg C} \sqsubseteq S_1 & (\textbf{funct } S_1) \\
S_{2,C} \sqsubseteq S_2 & S_{2,\neg C} \sqsubseteq S_2 & (\textbf{funct } S_2) \\
\exists S_{1,C} \equiv \exists S_{2,C} & \exists S_{1,\neg C} \equiv \exists S_{2,\neg C} &
\end{array}
$$

$$\exists S_2 \sqsubseteq \exists S_{2,C} \sqcup \exists S_{2,\neg C}$$

$$
\begin{array}{ll}
\exists S_{2,C}^{-} \sqsubseteq C & \exists S_{2,\neg C}^{-} \sqsubseteq \neg C
\end{array}
$$

$$A \sqsubseteq \neg \exists S_{1,\neg C}^{-}$$

Complexity of *DL-Lite* with functionality and role inclusions

We can exploit the above constructions that simulate DL constructs to show lower bounds for reasoning with both functionality and role inclusions.

Theorem [Artale *et al.*, 2009]

For $DL\text{-}Lite_{\mathcal{FR}}$ ontologies:

- TBox reasoning is ExpTime-**complete** in the size of the **TBox**.
- Checking satisfiability of the ontology is
  - PTime-**complete** in the size of the **ABox** (data complexity).
  - ExpTime-**complete** in the size of the **ontology** (combined complexity).
- Query answering is
  - PTime-**complete** in the size of the **ABox** (data complexity).
  - ExpTime-**complete** in the size of the **ontology**.
  - in 2ExpTime in the size of the query and the ontology (combined com.).

unibz

Combining functionalities and role inclusions

We have seen that:

- By including in *DL-Lite* both functionality of roles and role inclusions without restrictions on their interaction, query answering becomes PTIME-hard.
- When the data complexity of query answering is NLOGSPACE or above, the DL does not enjoy FOL-rewritability.

As a consequence of these results, we get:

To preserve FOL-rewritability, the restriction on the interaction of functionality and role inclusions of $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ is necessary.

unibz

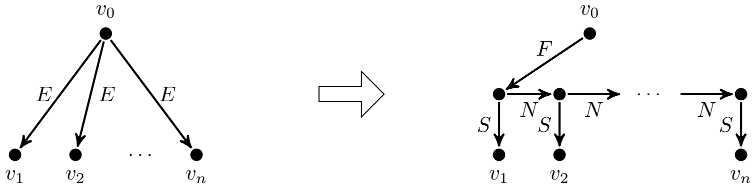
Outline of Part 5

Dropping the unique name assumption

Recall: the unique name assumption (UNA) states that different individuals must be interpreted as different domain objects.

We reconsider the complexity of query evaluation in *DL-Lite*$_\mathcal{F}$, and show that **without the UNA the data complexity increases**.

- We show how to reduce **reachability in directed graphs** to instance checking in *DL-Lite*$_\mathcal{F}$ without the UNA. This gives us an NLogSpace lower bound.
- We assume that the graph is represented through the first-child and next-sibling functional relations:



Dropping the unique name assumption (cont'd)

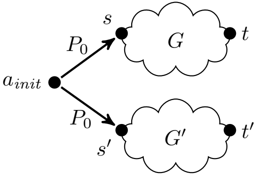
From G and two vertexes s and t of G, we define $\mathcal{O}_{una} = \langle \mathcal{T}_{una}, \mathcal{A}_G \rangle$:

- TBox uses an atomic concept A, and atomic roles P_0, P_F, P_N, P_S:

$$\mathcal{T}_{una} \;=\; \{(\textbf{funct}\ P_0)\} \cup \{(\textbf{funct}\ P_{\mathcal{R}}) \mid \mathcal{R} \in \{F, N, S\}\}.$$

- ABox is defined from G and the two vertexes s and t:

$$\begin{aligned}\mathcal{A}_G \;=\; & \{P_{\mathcal{R}}(a_1, a_2), P_{\mathcal{R}}(a'_1, a'_2) \mid (a_1, a_2) \in \mathcal{R},\ \text{for } \mathcal{R} \in \{F, N, S\}\} \cup \\ & \{A(t),\ P_0(a_{init}, s),\ P_0(a_{init}, s')\}\end{aligned}$$



This means that we encode in $\mathcal{A}_G$ two copies of G.

Note: $\mathcal{A}_G$ depends on G, but $\mathcal{T}_{una}$ does not.

We can show by induction on the length of paths from s that . . .

t is reachable from s in G if and only if $\mathcal{O}_{una} \models A(t')$.

Dropping the unique name assumption – Complexity

The previous reduction shows that instance checking in $\textit{DL-Lite}_{\mathcal{F}}$ (and hence also $\textit{DL-Lite}_{\mathcal{A}}$) without the UNA is NLOGSPACE-hard.

With a more involved reduction, one can show an even stronger lower bound, that turns out to be tight.

Theorem [Artale *et al.*, 2009]

Instance checking in $\textit{DL-Lite}_{\mathcal{F}}$ and $\textit{DL-Lite}_{\mathcal{A}}$ without the UNA is PTIME-complete in data complexity.

References I

[Artale *et al.*, 2009] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev.
The *DL-Lite* family and relations.
J. of Artificial Intelligence Research, 36:1–69, 2009.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz.
Pushing the $\mathcal{EL}$ envelope.
In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, 2005.

[Donini *et al.*, 1994] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf.
Deduction in concept languages: From subsumption to instance checking.
J. of Logic and Computation, 4(4):423–452, 1994.

[Grosof *et al.*, 2003] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker.
Description logic programs: Combining logic programs with description logic.
In *Proc. of the 12th Int. World Wide Web Conf. (WWW 2003)*, pages 48–57, 2003.

References II

[Hustadt *et al.*, 2005] Ullrich Hustadt, Boris Motik, and Ulrike Sattler.
Data complexity of reasoning in very expressive description logics.
In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005.