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 $EL^{++}$
   - Syntax and Semantics of $EL^{++}$
   - Reasoning in $EL$
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 \(\mathcal{EL}\)
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 ⊑ ⊤,
  - and roles
    - teaches ⊑ ⊓ Student ⊓ ⊥
    - attends

"Botoeva Lightweight Description Logics: DL-Lite_A and EL++"
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 ⊔ ⊤ ⊔ ⊥
  - and roles
    - teaches ⊔ attends

- variable free syntax
  - for describing complex concepts
    - Professor ⊔ Student ⊔ ∃ teaches.PhDCourse ⊔ ∀ hasChild.Male
  - for asserting implicit knowledge
    - ∃ teaches¬ ⊑ Course ⊔ Professor ⊔ Student ⊑ ⊥
  - for asserting explicit knowledge
    - Student(john) ⊔ attends(john, db)
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).
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).

- 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 $\text{ExpTime}$-complete.
- Satisfiability of $\mathcal{SROIQ}$, the basis of OWL 2, is $2\text{NExpTime}$-complete.
Lightweight Description Logics

The majority of studied DLs is intractable:

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

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

▶ A common feature: no disjunction and no universal restrictions

Professor $\sqcup$ Student $\quad \forall \text{hasChild}.\text{Male}$
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 $EL^{++}$
   - Syntax and Semantics of $EL^{++}$
   - Reasoning in $EL$
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. 
    ⇒ $AC^0$ 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*.  
    \[ \Rightarrow AC^0 \] in data complexity.

- **DL-Lite\(^A\)** is the most expressive member of this family.
Outline

1 Description Logics

2 Description Logic $DL-Lite_\mathcal{A}$
   - Syntax and Semantics of $DL-Lite_\mathcal{A}$
   - Reasoning in $DL-Lite_\mathcal{A}$
     - Knowledge Base Satisfiability
     - Conjunctive Query Answering

3 Description Logic $\mathcal{EL}^{++}$
   - Syntax and Semantics of $\mathcal{EL}^{++}$
   - Reasoning in $\mathcal{EL}$
**DL-Lite}_A Syntax**

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

- $B ::= \ A \ | \ \exists R \quad \text{basic concept}$
- $C ::= B \ | \ \neg B \quad \text{complex concept}$
- $R ::= P \ | \ P^\neg \quad \text{basic role}$
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:
  \[
  B ::= A | \exists R \quad \text{basic concept}
  
  C ::= B | \neg B \quad \text{complex concept}
  
  R ::= P | P^- \quad \text{basic role}
  \]

- TBox and ABox assertions:
  \[
  \begin{align*}
  B_1 \sqsubseteq B_2 & \quad \text{concept inclusion} \\
  B_1 \sqsubseteq \neg B_2 & \quad \text{disjointness of concepts} \\
  R_1 \sqsubseteq R_2 & \quad \text{role inclusion} \\
  \text{Dis}(R_1, R_2) & \quad \text{disjointness of roles} \\
  \text{Funct}(R) & \quad \text{role functionality}
  \end{align*}
  \]
  \[
  A(a) \quad \text{membership assertions}
  
  P(a, b) \quad \text{assertions}
  \]
**DL-Lite\(_A\) 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:
  
  \[
  \begin{align*}
  B & ::= A \mid \exists R & \text{basic concept} \\
  C & ::= B \mid \neg B & \text{complex concept} \\
  R & ::= P \mid P^- & \text{basic role}
  \end{align*}
  \]

- TBox and ABox assertions:
  
  \[
  \begin{align*}
  B_1 \sqsubseteq B_2 & \quad \text{concept inclusion} \\
  B_1 \sqsubseteq \neg B_2 & \quad \text{disjointness of concepts} \\
  R_1 \sqsubseteq R_2 & \quad \text{role inclusion} \\
  \text{Dis}(R_1, R_2) & \quad \text{disjointness of roles} \\
  \text{Funct}(R) & \quad \text{role functionality}
  \end{align*}
  \]

- A **DL-Lite\(_A\)** Knowledge Base \(K\) is a pair \(\langle T, A \rangle\) where
  
  - \(T\) is a finite set of TBox axioms and
  - \(A\) is a finite set of membership assertions.
**DL-Lite\_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:
  
  \[ B ::= A \mid \exists R \quad \text{basic concept} \]
  
  \[ C ::= B \mid \neg B \quad \text{complex concept} \]
  
  \[ R ::= P \mid P^{-} \quad \text{basic role} \]

- TBox and ABox assertions:
  
  \[ B_1 \sqsubseteq B_2 \quad \text{concept inclusion} \]
  
  \[ B_1 \sqsubseteq \neg B_2 \quad \text{disjointness of concepts} \]
  
  \[ R_1 \sqsubseteq R_2 \quad \text{role inclusion} \]
  
  \[ \text{Dis}(R_1, R_2) \quad \text{disjointness of roles} \]
  
  \[ \text{Funct}(R) \quad \text{role functionality} \]

- A **DL-Lite\_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.
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:

\[ R' \subseteq R \]

\[ \text{Funct}(R) \]
DL-Lite_A Syntax

Syntactic restriction to ensure tractability:

**Functional roles cannot be specialized.**

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

\[ R' \subseteq R \]
\[ \text{Funct}(R) \]

Otherwise, the resulting logic is \( \text{ExpTime} \)-hard in the size of ontology[1].
\textbf{DL-Lite}_A \textit{Semantics}

- An \textit{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}_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

\[
\begin{align*}
(\neg B)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \\
(\exists R)^{\mathcal{I}} &= \{ x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}} \} \\
(P^-)^{\mathcal{I}} &= \{ (y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in P^{\mathcal{I}} \} 
\end{align*}
\]
DL-Lite\(_A\) Semantics

- An *interpretation* \(\mathcal{I}\) is a pair \(⟨\Delta^\mathcal{I}, \cdot^\mathcal{I}⟩:\)
  - 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
  
  \[
  (¬B)^\mathcal{I} = \Delta^\mathcal{I} \setminus B^\mathcal{I} \\
  (\exists R)^\mathcal{I} = \{x \in \Delta^\mathcal{I} | \exists y \in \Delta^\mathcal{I}, (x, y) \in R^\mathcal{I}\} \\
  (P^-)^\mathcal{I} = \{(y, x) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} | (x, y) \in P^\mathcal{I}\}
  \]

- TBox and ABox assertions
  
  \[
  \mathcal{I} \models B \sqsubseteq C \quad \text{iff} \quad B^\mathcal{I} \subseteq C^\mathcal{I} \\
  \mathcal{I} \models R_1 \sqsubseteq R_2 \quad \text{iff} \quad R_1^\mathcal{I} \subseteq R_2^\mathcal{I} \\
  \mathcal{I} \models \text{Dis}(R_1, R_2) \quad \text{iff} \quad R_1^\mathcal{I} \cap R_2^\mathcal{I} = \emptyset \\
  \mathcal{I} \models \text{Funct}(R) \quad \text{iff} \quad (x, y_1) \in R^\mathcal{I}, (x, y_2) \in R^\mathcal{I} \Rightarrow y_1 = y_2 \\
  \mathcal{I} \models A(a) \quad \text{iff} \quad a^\mathcal{I} \in A^\mathcal{I} \\
  \mathcal{I} \models P(a, b) \quad \text{iff} \quad (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}
  \]
**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
  
  \[
  (-B)^\mathcal{I} = \Delta^\mathcal{I} \setminus B^\mathcal{I} \\
  (\exists R)^\mathcal{I} = \{ x \in \Delta^\mathcal{I} | \exists y \in \Delta^\mathcal{I}, (x, y) \in R^\mathcal{I}\} \\
  (P^\neg)^\mathcal{I} = \{ (y, x) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} | (x, y) \in P^\mathcal{I}\}
  \]

- TBox and ABox assertions

  \[
  \mathcal{I} \models B \sqsubseteq C \quad \text{iff} \quad B^\mathcal{I} \subseteq C^\mathcal{I} \\
  \mathcal{I} \models R_1 \sqsubseteq R_2 \quad \text{iff} \quad R_1^\mathcal{I} \subseteq R_2^\mathcal{I} \\
  \mathcal{I} \models \text{Dis}(R_1, R_2) \quad \text{iff} \quad R_1^\mathcal{I} \cap R_2^\mathcal{I} = \emptyset \\
  \mathcal{I} \models \text{Funct}(R) \quad \text{iff} \quad (x, y_1) \in R^\mathcal{I}, (x, y_2) \in R^\mathcal{I} \Rightarrow y_1 = y_2 \\
  \mathcal{I} \models A(a) \quad \text{iff} \quad a^\mathcal{I} \in A^\mathcal{I} \\
  \mathcal{I} \models P(a, b) \quad \text{iff} \quad (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}
  \]

- \(\mathcal{I}\) 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\(_A\) – Example**

Note: **DL-Lite\(_A\)** cannot capture completeness of a hierarchy. This would require **disjunction** (i.e., OR).
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 $EL^{++}$
   - Syntax and Semantics of $EL^{++}$
   - Reasoning in $EL$
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.
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 $\text{ans}(q, \mathcal{K})$ of certain answers.
Reasoning Problems

- The **Knowledge Base Satisfiability** problem is to check, given a $\mathcal{DL-Lite}_\mathcal{A}$ 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 $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$ ($\mathcal{T} \models C_1 \subseteq C_2$).
  - 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 $R_1^\mathcal{I} \subseteq R_2^\mathcal{I}$ ($\mathcal{T} \models R_1 \subseteq R_2$).

- The **Query Answering** problem is to compute, given a $\mathcal{DL-Lite}_\mathcal{A}$ KB $\mathcal{K}$ and a query $q$ (either a CQ or a UCQ) over $\mathcal{K}$, the set $\text{ans}(q, \mathcal{K})$ 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 $\text{tab}_A$
  - populate $\text{tab}_A$ 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 $\text{tab}_P$
  - populate $\text{tab}_P$ 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 $\text{tab}_A$
  - populate $\text{tab}_A$ 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 $\text{tab}_P$
  - populate $\text{tab}_P$ 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 $DL$-$\mathcal{A}$ 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 $\bar{a}$ from $\mathcal{A}$), $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable iff $q'(\cdot)$ evaluates to false in $DB(\mathcal{A})$ ($\bar{a} \in \text{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff $\bar{a}_{DB(\mathcal{A})} \in q'_{DB(\mathcal{A})}$).
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 $\text{tab}_A$
  - populate $\text{tab}_A$ 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 $\text{tab}_P$
  - populate $\text{tab}_P$ 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 $T$ (and every UCQ $q$) there exists a FO query $q'$, such that for every nonempty $\mathcal{A}$ (and every tuple of constants $\bar{a}$ from $\mathcal{A}$), $\langle T, \mathcal{A} \rangle$ is satisfiable iff $q'(\bar{a})$ evaluates to false in $DB(\mathcal{A})$ $(\bar{a} \in \text{ans}(q, \langle T, \mathcal{A} \rangle))$ iff $\bar{a}^{DB(\mathcal{A})} \in q'^{DB(\mathcal{A})})$.

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

**Problem**

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

Problem

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

- **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 $B_1 \sqsubseteq \neg B_2$, $\text{Dis}(R_1, R_2)$, or $\text{Funct}(R)$
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 $\mathcal{K} = \langle T, A \rangle$ be a $DL$-$Lite_A$ KB such that $T$ consists only of PIs. Then $\mathcal{K}$ is satisfiable.
Satisfiability of KBs with only PIs

Positive inclusions cannot introduce contradicting information:

**Theorem**

Let $\mathcal{K} = \langle T, A \rangle$ be a DL-Lite$_A$ KB such that $T$ 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:

- $\mathcal{T}$ : Dis(teaches, attends)
- $\mathcal{A}$ : teaches(john, db), attends(john, db)
Source of Unsatisfiability

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

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

- $\mathcal{T} : \text{Funct(teaches}^-)\text{)}$
  $\mathcal{A} : \text{teaches(john, db), teaches(david, db)}$
Source of Unsatisfiability

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

1. \( T : \text{Dis(teaches, attends)} \)
   
   \( A : \text{teaches(john, db), attends(john, db)} \)

2. \( T : \text{Funct(teaches}^-) \)
   
   \( A : \text{teaches(john, db), teaches(david, db)} \)

3. \( T : \text{Student } \sqsubseteq \neg \text{Professor}, \exists \text{teaches } \sqsubseteq \text{Professor} \)
   
   \( A : \text{Student(john), teaches(john, db)} \)
Source of Unsatisfiability

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

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

- $\mathcal{T} : \text{Funct(teaches$^{-}$)}$
  $\mathcal{A} : \text{teaches(john, db), teaches(david, db)}$

- $\mathcal{T} : \text{Student}$ $\sqsubseteq$ $\neg \text{Professor}$, $\exists \text{teaches} \sqsubseteq \text{Professor}$
  $\mathcal{A} : \text{Student(john), teaches(john, db)}$

  - Interaction of negative and positive inclusions has to be considered.
    $\Rightarrow$ calculate the closure of NIs w.r.t. Pls.
Knowledge Base Satisfiability

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

Given a $DL$-$Lite_\mathcal{A}$ KB $\mathcal{K} = \langle T, \mathcal{A} \rangle$, 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 $DL$-Lite$_A$ KB $\mathcal{K} = \langle T, A \rangle$, 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 $DB(A)$.
   - if $\text{Eval}(SQL(q_{unsat}), DB(A)) = \emptyset$, 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 $cln(T)$ w.r.t. PIs

- every NI is in $cln(T)$. 

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

Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

- $cln(\mathcal{T}) : \quad \text{Student} \sqsubseteq \neg \text{Professor}$

- $\mathcal{T} : \quad \exists \text{teaches} \sqsubseteq \text{Professor}$

$$\Rightarrow$$

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

Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

$$\text{cln}(\mathcal{T}) : \quad \text{Student} \sqsubseteq \neg \text{Professor} \quad \text{∃teaches} \sqsubseteq \text{Professor}$$

$\mathcal{T} : \quad \text{∃teaches} \sqsubseteq \text{Professor}$

add to $\text{cln}(\mathcal{T}) : \quad \text{Student} \sqsubseteq \neg \exists \text{teaches}$

Closed: $\text{cln}(\mathcal{T})$ is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

- $\text{cln}(T) : \quad \begin{align*} &\text{Student} \sqsubseteq \neg \text{Professor} \\ &\text{T} : \quad \exists\text{teaches} \sqsubseteq \text{Professor} \quad \} \quad \Rightarrow \\ &\text{add to } \text{cln}(T) : \quad \text{Student} \sqsubseteq \neg \exists\text{teaches} \end{align*}$

- $\text{cln}(T) : \quad \begin{align*} &\text{Professor} \sqsubseteq \neg \exists\text{attends} \\ &\text{T} : \quad \text{registeredTo} \sqsubseteq \exists\text{attends} \quad \} \quad \Rightarrow \end{align*}$

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

Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

\[
cln(\mathcal{T}) : \quad \begin{align*}
\text{Student} & \sqsubseteq \neg \text{Professor} \\
\exists \text{teaches} & \sqsubseteq \text{Professor} \\
\end{align*} \quad \Rightarrow \\
\text{T} : \quad \exists \text{teaches} & \sqsubseteq \text{Professor} \\
\text{add to } cln(\mathcal{T}) : \quad \text{Student} & \sqsubseteq \neg \exists \text{teaches}
\]

\[
cln(\mathcal{T}) : \quad \begin{align*}
\text{Professor} & \sqsubseteq \neg \exists \text{attends} \\
\text{registeredTo} & \sqsubseteq \text{attends} \\
\end{align*} \quad \Rightarrow \\
\text{T} : \quad \text{registeredTo} & \sqsubseteq \text{attends} \\
\text{add to } cln(\mathcal{T}) : \quad \text{Professor} & \sqsubseteq \neg \exists \text{registeredTo}
\]
Closure of Negative Inclusions

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

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

- $\text{cln}(\mathcal{T}) : \begin{cases} \text{Student} \sqsubseteq \neg \text{Professor} \\ \mathcal{T} : \exists \text{teaches} \sqsubseteq \text{Professor} \end{cases} \Rightarrow$

  add to $\text{cln}(\mathcal{T})$ : $\text{Student} \sqsubseteq \neg \exists \text{teaches}$

- $\text{cln}(\mathcal{T}) : \begin{cases} \text{Professor} \sqsubseteq \neg \exists \text{attends} \\ \mathcal{T} : \text{registeredTo} \sqsubseteq \exists \text{attends} \end{cases} \Rightarrow$

  add to $\text{cln}(\mathcal{T})$ : $\text{Professor} \sqsubseteq \neg \exists \text{registeredTo}$

- $\text{cln}(\mathcal{T}) : \begin{cases} \text{Dis(teaches, attends)} \\ \mathcal{T} : \text{registeredTo} \sqsubseteq \exists \text{attends} \end{cases} \Rightarrow$

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

Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

- $\text{cln}(\mathcal{T}) : \text{Student} \sqsubseteq \lnot \text{Professor}$ \[ \text{\exists teaches} \sqsubseteq \text{Professor} \]  
  add to $\text{cln}(\mathcal{T}) : \text{Student} \sqsubseteq \lnot \exists \text{teaches}$

- $\text{cln}(\mathcal{T}) : \text{Professor} \sqsubseteq \lnot \exists \text{attends}$ \[ \text{registeredTo} \sqsubseteq \text{attends} \]  
  add to $\text{cln}(\mathcal{T}) : \text{Professor} \sqsubseteq \lnot \exists \text{registeredTo}$

- $\text{cln}(\mathcal{T}) : \text{Dis(teaches, attends)}$ \[ \text{registeredTo} \sqsubseteq \text{attends} \]  
  add to $\text{cln}(\mathcal{T}) : \text{Dis(teaches, registeredTo)}$

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

Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

- $\text{cln}(\mathcal{T}) : \quad \text{Student} \sqsubseteq \neg \text{Professor}$
  $\mathcal{T} : \quad \exists \text{teaches} \sqsubseteq \text{Professor}$
  add to $\text{cln}(\mathcal{T}) : \quad \text{Student} \sqsubseteq \neg \exists \text{teaches}$

- $\text{cln}(\mathcal{T}) : \quad \text{Professor} \sqsubseteq \neg \exists \text{attends}$
  $\mathcal{T} : \quad \text{registeredTo} \sqsubseteq \text{attends}$
  add to $\text{cln}(\mathcal{T}) : \quad \text{Professor} \sqsubseteq \neg \exists \text{registeredTo}$

- $\text{cln}(\mathcal{T}) : \quad \text{Dis(teaches, attends)}$
  $\mathcal{T} : \quad \text{registeredTo} \sqsubseteq \text{attends}$
  add to $\text{cln}(\mathcal{T}) : \quad \text{Dis(teaches, registeredTo)}$

- ...

Note: functionality does not interact with PIs and other NIs.
Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

- $\text{cln}(\mathcal{T}) : \begin{align*}
    \text{Student} & \sqsubseteq \neg \text{Professor} \quad \text{(or Professor} \sqsubseteq \neg \text{Student)} \\
    \mathcal{T} : \begin{cases}
        \exists \text{teaches} \sqsubseteq \text{Professor} \\
        \text{add to } \text{cln}(\mathcal{T}) : \text{Student} \sqsubseteq \neg \exists \text{teaches}
    \end{cases}
\end{align*}$

- $\text{cln}(\mathcal{T}) : \begin{align*}
    \text{Professor} & \sqsubseteq \neg \exists \text{attends} \quad \text{(or } \exists \text{attends} \sqsubseteq \neg \text{Professor)} \\
    \mathcal{T} : \begin{cases}
        \text{registeredTo} \sqsubseteq \text{attends} \\
        \text{add to } \text{cln}(\mathcal{T}) : \text{Professor} \sqsubseteq \neg \exists \text{registeredTo}
    \end{cases}
\end{align*}$

- $\text{cln}(\mathcal{T}) : \begin{align*}
    \text{Dis(teaches, attends)} \quad \text{(or } \text{Dis(attends, teaches)}) \\
    \mathcal{T} : \begin{cases}
        \text{registeredTo} \sqsubseteq \text{attends} \\
        \text{add to } \text{cln}(\mathcal{T}) : \text{Dis(teaches, registeredTo)}
    \end{cases}
\end{align*}$

- ...

---

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

Note: the closure is finite since there are polynomially many different NIs.
Closure of Negative Inclusions

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

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

- $\text{cln}(T)$: $\text{Student} \sqsubseteq \neg \text{Professor}$
  $T$: $\exists \text{teaches} \sqsubseteq \text{Professor}$
  add to $\text{cln}(T)$: $\text{Student} \sqsubseteq \neg \exists \text{teaches}$

- $\text{cln}(T)$: $\text{Professor} \sqsubseteq \neg \exists \text{attends}$
  $T$: $\text{registeredTo} \sqsubseteq \text{attends}$
  add to $\text{cln}(T)$: $\text{Professor} \sqsubseteq \neg \exists \text{registeredTo}$

- $\text{cln}(T)$: $\text{Dis(teaches, attends)}$
  $T$: $\text{registeredTo} \sqsubseteq \text{attends}$
  add to $\text{cln}(T)$: $\text{Dis(teaches, registeredTo)}$

- ...

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

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

  $\vdash \neg \exists x.\text{Student}(x)$
Translation to FOL Queries

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

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

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

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

  - $\text{Funct(teaches}^{-}) \Rightarrow$
    $$\exists x_1, x_2, y. \text{teaches}(x_1, y) \land \text{teaches}(x_2, y) \land x_1 \neq x_2.$$  

  - $\text{Dis(attends, teaches)} \Rightarrow$
    $$\exists x, y. \text{attends}(x, y) \land \text{teaches}(x, y).$$

- Then
  $$q_{\text{unsat}} = \bigvee_{\alpha \in \text{cln}(T)} \delta(\alpha)$$
Query evaluation

Let $q$ be a UCQ.

- We denote by $SQL(q)$ the encoding of $q$ into an SQL query over $DB(A)$.

- We indicate with $\text{Eval}(SQL(q), DB(A))$ the evaluation of $SQL(q)$ over $DB(A)$. 
FOL-rewritability of satisfiability in $DL$-$Lite_\mathcal{A}$

**Theorem**

Let $\mathcal{K} = \langle T, \mathcal{A} \rangle$ be a $DL$-$Lite_\mathcal{A}$ KB. Then, $\mathcal{K}$ is unsatisfiable iff $\text{Eval}(\text{SQL}(q_{\text{unsat}}, DB(\mathcal{A})))$ 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 $\mathcal{K} = \langle T, A \rangle$ is a form of *logical implication*:

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

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

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

*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 $A$, 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 $\text{cert}(q, \mathcal{K})$ 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 $\text{cert}(q, \mathcal{K})$. 

Correctness of this procedure shows FOL-rewritability of query answering in DL-Lite.
Query Answering over Satisfiable KBs

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

**Algorithm for answering CQs over KBs**

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

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

$\leadsto$ 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 \( q(x) \leftarrow \text{Professor}(x) \)

**Intuition:** Use the PIs as basic rewriting rules:

\[
\begin{align*}
\text{AssistantProf} & \sqsubseteq \text{Professor} \\
\text{as a logic rule:} & \quad \text{Professor}(z) \leftarrow \text{AssistantProf}(z)
\end{align*}
\]

**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 \( q(x) \leftarrow \text{Professor}(x) \)

**Intuition:** Use the PIs as basic rewriting rules:

\[
\text{AssistantProf} \sqsubseteq \text{Professor}
\]

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

**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 \( \text{AssistantProf} \sqsubseteq \text{Professor} \) applies to the atom \( \text{Professor}(x) \). Towards the computation of the perfect rewriting, we add to the input query above, the query

\[
q(x) \leftarrow \text{AssistantProf}(x)
\]
Query Rewriting (cont’d)

Consider the query

\[ q(x) \leftarrow \text{teaches}(x, y), \text{Course}(y) \]

and the PI

\[ \exists \text{teaches}^\neg \sqsubseteq \text{Course} \]

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

\[ q(x) \leftarrow \text{teaches}(x, y), \text{Course}(y) \]

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

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

Consider now the query

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

and the PI

\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]

as a logic rule:

\[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

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

\[ q(x) \leftarrow \text{Professor}(x) \]
Query Rewriting – Constants

Conversely, for the query

\[ q(x) \leftarrow \text{teaches}(x, \text{databases}) \]

and the same PI as before

\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]

as a logic rule:

\[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

teaches\((x, \text{databases})\) does not unify with teaches\((z, f(z))\), since the skolem term \(f(z)\) in the head of the rule does not unify with the constant \text{databases}.\]
Query Rewriting – Constants

Conversely, for the query

\[ q(x) \leftarrow \text{teaches}(x, \text{databases}) \]

and the same PI as before

as a logic rule:

\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]

\[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

\text{teaches}(x, \text{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 \( \text{databases} \).

In this case, the PI does not apply to the atom \( \text{teaches}(x, \text{databases}) \).
Query Rewriting – Constants

Conversely, for the query

\( q(x) \leftarrow \text{teaches}(x, \text{databases}) \)

and the same PI as before

\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]

as a logic rule:

\[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

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

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

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

\[ q(x) \leftarrow \text{teaches}(x, y), \text{Course}(y) \]

and the PI

\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]

as a logic rule:

\[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

The PI above does not apply to the atom \( \text{teaches}(x, y) \).
Query Rewriting – Reduce step

Consider now the query
\[ q(x) \leftarrow \text{teaches}(x, y), \text{teaches}(z, y) \]

and the PI
\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]
as a logic rule:
\[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

This PI does not apply to \( \text{teaches}(x, y) \) or \( \text{teaches}(z, y) \), 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
\[ q(x) \leftarrow \text{teaches}(x, y), \text{teaches}(z, y) \]

and the PI
\[ \text{Professor} \sqsubseteq \exists \text{teaches} \]
as a logic rule: \[ \text{teaches}(z, f(z)) \leftarrow \text{Professor}(z) \]

This PI does not apply to \text{teaches}(x, y) or \text{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 \text{teaches}(x, y) and \text{teaches}(z, y). This rewriting step is called reduce, and produces the query
\[ q(x) \leftarrow \text{teaches}(x, y) \]

Now, we can apply the PI above, and add to the rewriting the query
\[ q(x) \leftarrow \text{Professor}(x) \]
Query Rewriting Algorithm

Algorithm PerfectRef\((Q, T_P)\)

Input: union of conjunctive queries \(Q\), set of DL-Lite\(_A\) PIs \(T_P\)

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 \(T_P\) 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 \textit{DL-Lite} – Example

\textbf{TBox:} \quad \text{Professor} \sqsubseteq \exists \text{teaches} \\
\quad \exists \text{teaches}^- \sqsubseteq \text{Course}

\textbf{Query:} \quad q(x) \leftarrow \text{teaches}(x, y), \text{Course}(y)

\textbf{Perfect Rewriting:} \quad q(x) \leftarrow \text{teaches}(x, y), \text{Course}(y) \\
\quad q(x) \leftarrow \text{teaches}(x, y), \text{teaches}(-, y) \\
\quad q(x) \leftarrow \text{teaches}(x, -) \\
\quad q(x) \leftarrow \text{Professor}(x)

\textbf{ABox:} \quad \text{teaches}(\text{john}, \text{databases}) \\
\quad \text{Professor}(\text{mary})

It is easy to see that evaluating the perfect rewriting over the ABox viewed as a database produces as answer \{\text{john}, \text{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 $r_{q,\mathcal{T}} = \text{PerfectRef}(q, \mathcal{T}_P)$. Then, for each ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, we have that

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

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

**Theorem**

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

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

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 $T$ 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 $T$.*
Complexity of Reasoning in \( DL-Lite \)

**Theorem**

Checking satisfiability of \( DL-Lite_A \) KBs is

1. \textbf{PTIME} in the size of the \textit{KB} (combined complexity).
2. \textbf{AC}^0 in the size of the \textit{ABox} (data complexity).

**Theorem**

Query answering over \( DL-Lite_A \) KBs is

1. \textbf{NP-complete} in the size of query and \textit{KB} (combined comp.).
2. \textbf{PTIME} in the size of the \textit{KB}.
3. \textbf{AC}^0 in the size of the \textit{ABox} (data complexity).
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 $EL^{++}$
   - Syntax and Semantics of $EL^{++}$
   - Reasoning in $EL$
\(\mathcal{EL}\) 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 $EL^{++}$
   - Syntax and Semantics of $EL^{++}$
   - Reasoning in $EL$
\textbf{\(\mathcal{EL}^{++}\) Syntax}

Let \(N_A, N_P, N_a\) be sets of concept, role and individual names, respectively. Let \(A \in N_A, P \in N_P, a \in N_a\).
**Ł++ 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:

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

**Note:** The concrete domain constructor, which is a part of Ł++, 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:
  \[
  C, D ::= \top | \bot | A | \{a\} | C \cap D | \exists P.C
  \]

- TBox and ABox assertions:
  
  \[
  C \sqsubseteq D \quad \text{concept inclusion} \quad \quad A(a) \quad \text{membership}
  \]
  \[
  P_1 \odot \cdots \odot P_n \sqsubseteq P \quad \text{complex role inclusion} \quad \quad P(a, b) \quad \text{assertions}
  \]
\( \mathcal{EL}^{++} \) 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:
  \[
  C, D ::= \top | \bot | A | \{a\} | C \cap D | \exists P. C
  \]

- TBox and ABox assertions:
  \[
  C \sqsubseteq D \quad \text{concept inclusion} \quad A(a) \quad \text{membership}
  \]
  \[
  P_1 \circ \cdots \circ P_n \sqsubseteq P \quad \text{complex role inclusion} \quad P(a, b) \quad \text{assertions}
  \]

- 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.
**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:**
  
  \[ C, D ::= \top | \bot | A | \{a\} | C \cap D | \exists P.C \]

- **TBox and ABox assertions:**
  
  \[ C \sqsubseteq D \text{ concept inclusion } A(a) \text{ membership } \]
  \[ P_1 \circ \cdots \circ P_n \sqsubseteq P \text{ complex role inclusion } P(a, b) \text{ assertions } \]

- An \( \mathcal{EL}^{++} \) Knowledge Base \( K \) is a pair \( \langle T, A \rangle \) where
  
  - \( T \) is a finite set of TBox axioms and
  - \( A \) 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:

\[
C, D ::= \top \mid \bot \mid A \mid \{a\} \mid C \cap D \mid \exists P.C
\]

- TBox and ABox assertions:

\[
C \sqsubseteq D \quad \text{concept inclusion} \quad A(a) \quad \text{membership assertion}
\]

\[
P_1 \circ \cdots \circ P_n \sqsubseteq P \quad \text{complex role inclusion} \quad P(a, b) \quad \text{assertions}
\]

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

\( \mathcal{EL} \) concept constructs and assertions.
\[ \mathcal{EL}^{++} \text{ Semantics} \]

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 \( a \), \( a^\mathcal{I} \in \Delta^\mathcal{I} \).
**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

\[
\begin{align*}
(\top)^\mathcal{I} &= \Delta^\mathcal{I} \\
(\bot)^\mathcal{I} &= \emptyset \\
\{a\}^\mathcal{I} &= \{a^\mathcal{I}\} \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(\exists P.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} \mid \exists y \in C^\mathcal{I}, (x, y) \in P^\mathcal{I}\}
\end{align*}
\]
\( \mathcal{EL}^{++} \) 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
  \[
  \begin{align*}
  (\top)^\mathcal{I} & = \Delta^\mathcal{I} \\
  (\bot)^\mathcal{I} & = \emptyset \\
  (\{a\})^\mathcal{I} & = \{a^\mathcal{I}\}
  \end{align*}
  \]
  \[
  (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}
  \]
  \[
  (\exists P.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y \in C^\mathcal{I}, (x, y) \in P^\mathcal{I} \}
  \]

- TBox and ABox assertions
  \[
  \begin{align*}
  \mathcal{I} \models C \sqsubseteq D & \quad \text{iff} \quad C^\mathcal{I} \subseteq D^\mathcal{I} \\
  \mathcal{I} \models P_1 \circ \cdots \circ P_n \sqsubseteq P & \quad \text{iff} \quad \prod_{i=1}^{n} P_i^\mathcal{I} \subseteq P^\mathcal{I} \\
  \mathcal{I} \models A(a) & \quad \text{iff} \quad a^\mathcal{I} \in A^\mathcal{I} \\
  \mathcal{I} \models P(a, b) & \quad \text{iff} \quad (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}
  \end{align*}
  \]
\textbf{\(\mathcal{EL}^{++}\) 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

\[
\begin{align*}
(\top)^\mathcal{I} & = \Delta^\mathcal{I} \\
(\bot)^\mathcal{I} & = \emptyset \\
\{a\}^\mathcal{I} & = \{a^\mathcal{I}\}
\end{align*}
\]

\[
\begin{align*}
(C \sqcap D)^\mathcal{I} & = C^\mathcal{I} \cap D^\mathcal{I} \\
(\exists P.C)^\mathcal{I} & = \{x \in \Delta^\mathcal{I} | \exists y \in C^\mathcal{I}, (x, y) \in P^\mathcal{I}\}
\end{align*}
\]

- TBox and ABox assertions

\[
\begin{align*}
\mathcal{I} \models C \sqsubseteq D & \quad \text{iff} \quad C^\mathcal{I} \subseteq D^\mathcal{I} \\
\mathcal{I} \models P_1 \circ \cdots \circ P_n \sqsubseteq P & \quad \text{iff} \quad P_1^\mathcal{I} \circ \cdots \circ P_n^\mathcal{I} \subseteq P^\mathcal{I} \\
\mathcal{I} \models A(a) & \quad \text{iff} \quad a^\mathcal{I} \in A^\mathcal{I} \\
\mathcal{I} \models P(a, b) & \quad \text{iff} \quad (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}
\end{align*}
\]

- \(\mathcal{I}\) 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 \(^2\)

Endocardium \sqsubseteq Tissue \sqcap \exists \text{cont-in}.\text{HeartWall} \sqcap \exists \text{cont-in}.\text{HeartValve}

HeartWall \sqsubseteq BodyWall \sqcap \exists \text{part-of}.\text{Heart}

HeartValve \sqsubseteq BodyValve \sqcap \exists \text{part-of}.\text{Heart}

Endocarditis \sqsubseteq \text{Inflammation} \sqcap \exists \text{has-loc}.\text{Endocardium}

Inflammation \sqsubseteq \text{Disease} \sqcap \exists \text{acts-on}.\text{Tissue}

\text{Heartdisease} \sqsubseteq

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

\text{Heartdisease} \equiv \text{Disease} \sqcap \exists \text{has-loc}.\text{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}

\(^2\)taken from [4]

Botoeva Lightweight Description Logics: \( DL\text{-}\text{Lite}_\mathcal{A} \) and \( \mathcal{EL}^{++} \)
Outline

1 Description Logics

2 Description Logic $DL-Lite_\mathcal{A}$
   - Syntax and Semantics of $DL-Lite_\mathcal{A}$
   - Reasoning in $DL-Lite_\mathcal{A}$
     - 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 $DL$-$Lite_A$ KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model.

- The *Conjunctive Query Entailment* problem is to decide, given a $DL$-$Lite_A$ 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 $\mathcal{K} = \langle \mathcal{T}, A \rangle$, 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 tableaux 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 tableaux 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 queries in $\mathcal{EL}$ is:

1. $\text{PTime}$-complete with respect to data complexity;
2. $\text{PTime}$-complete with respect to KB complexity;
3. $\text{NP}$-complete with respect to combined complexity.
Thank you for your attention!


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

Tractable reasoning and efficient query answering in description logics: The DL-Lite family.

A. Krisnadhi and C. Lutz.
Data complexity in the EL family of description logics.

R. Rosati.
On conjunctive query answering in EL.