DL-Lite: Tractable Description Logics for Ontologies

Presented by: Kyle Robinson

Authored by: Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
Motivation

- Previous efficient DLs lack sufficient modeling power
- DLs with powerful modeling capabilities have high complexity
- Larger ontologies becoming common
- High complexity DLs unacceptable
Overview

• DL-Lite Overview
• Description Logic Components
• Reasoning Services
• KB Normalization
• KB Satisfiability
• Query Reformulation and Answering
• Reformulation Example
• Conclusion
DL-Lite Overview

• Tailored for basic ontology languages
• Low reasoning complexity
• Ability to answer complex queries
• Utilize an efficient SQL DB as secondary storage for ABoxes
• Can represent ER models and basic UML class diagrams
• Focused on conjunctive queries
Concepts

• Represent domain in terms of concepts, roles
• Allows for atomic concepts, existential quantification of roles, and inverse roles
• Only allowed negation of basic concepts
• No disjunction
Assertions

• TBox assertions include concept subsumption and role functionality

\[ B \sqsubseteq C \]
\[ \text{(funct } R), \ (\text{funct } R^-) \]

inclusion assertions
functionality assertions

• ABox assertions allow for membership assertions

\[ B(a), R(a, b) \]

membership assertions
Conjunctive Queries

• X’s are distinguished variables
• Y’s are non-distinguished variables
• Conj(x,y), conjunction of atoms of the form B(a) or R(a,b)
Interpretation

• Atomic concepts mapped to subset of domain
• Roles mapped to a binary relation
• Complement, domain without basic concept
• Existential roles, there exists \(c, c'\) such that \(c\) and \(c'\) are in \(R\)
• Intersection

\[
\begin{align*}
A^\mathcal{I} & \subseteq \Delta \\
(-B)^\mathcal{I} & = \Delta \setminus B^\mathcal{I} \\
(C_1 \cap C_2)^\mathcal{I} & = C_1^\mathcal{I} \cap C_2^\mathcal{I} \\
R^\mathcal{I} & \subseteq \Delta \times \Delta \\
(\exists R)^\mathcal{I} & = \{c \mid \exists c'. (c, c') \in R^\mathcal{I}\} \\
(\exists R^-)^\mathcal{I} & = \{c \mid \exists c'. (c', c) \in R^\mathcal{I}\}
\end{align*}
\]
Example

Tbox:

\[\text{Professor} \sqsubseteq \exists \text{TeachesTo} \quad \exists \text{TeachesTo}^- \sqsubseteq \text{Student} \quad \text{Professor} \sqsubseteq \neg \text{Student} \quad \text{Student} \sqsubseteq \exists \text{HasTutor} \quad \exists \text{HasTutor}^- \sqsubseteq \text{Professor} \quad (\text{func}\text{t} \ \text{HasTutor}).\]

Abox:

\[\text{HasTutor}(\text{John}, \text{Mary})\]
Reasoning Services

• Query answering
  – Given a query return a set of tuples containing the answer

• Query containment
  – Check if the result of a query is the subset of another query

• KB satisfiability
  – Consistency check
KB Normalization

- Expand ABox with existential roles
- Normalized ABox’s are stored in a DBMS
- Rewrite conjunctive concepts occurring in TBox as separate assertions
- TBox now contains only positive/negative inclusions, and functionality assertions
- Close TBox by computing NIs between basic concepts

Conjunctive Concepts:
\[ B \subseteq C_1 \cap C_2 \]
\[ B \subseteq C_1, B \subseteq C_2. \]

Positive Inclusion:
\[ B_1 \subseteq B_2. \]

Negative Inclusion:
\[ B_2 \subseteq \neg B_3. \]

Closing:
\[ B_1 \subseteq \neg B_3. \]
Example

Tbox after normalization:

\[ \text{Professor} \sqsubseteq \exists \text{TeachesTo} \quad \text{Student} \sqsubseteq \exists \text{HasTutor} \\
\exists \text{TeachesTo}^- \sqsubseteq \text{Student} \quad \exists \text{HasTutor}^- \sqsubseteq \text{Professor} \\
\text{Professor} \sqsubseteq \neg \text{Student} \quad \exists \text{TeachesTo}^- \sqsubseteq \neg \text{Professor} \quad \exists \text{HasTutor}^- \sqsubseteq \neg \text{Student}. \]

Abox after normalization:

\[ \text{HasTutor}(\text{John, Mary}) \]

\[ \exists \text{HasTutor}^- \quad \text{(Mary)} \]

\[ \exists \text{HasTutor} \quad \text{(John)} \]
KB Satiﬁability

• Check if ABox contradicts a NI
• Check if ABox violated role functionalities
• Veriﬁes consistency of KB

(i) there exists a NI $B_1 \subseteq \neg B_2$ in $\mathcal{T}$ and a constant $a$ such that the assertions $B_1(a)$ and $B_2(a)$ belong to $A$;
(ii) there exists an assertion (funct $R$) (respectively, (funct $R^-$)) in $\mathcal{T}$ and three constants $a, b, c$ such that both $R(a, b)$ and $R(a, c)$ (resp., $R(b, a)$ and $R(c, a)$) belong to $A$. 
Query Reformulation and Answering

• No structure available that can be evaluated for every conjunctive query
• Reformulate query with Tbox assertions
• Query size does not depend on ABox
• Complexity of answering algorithm is polynomial
• Transform each query into a SQL query
PerfectRef Algorithm

Algorithm PerfectRef(q, T)
Input: conjunctive query q, DL-Lite TBox T
Output: set of conjunctive queries P
P := \{q\};
repeat
    P' := P;
    for each q ∈ P' do
        (a) for each g in q do
            for each PI I in T do
                if I is applicable to g
                then P := P ∪ \{q[g/gr(g, I)]\}
        (b) for each g₁, g₂ in q do
            if g₁ and g₂ unify
            then P := P ∪ \{τ(reduce(q, g₁, g₂))\};
    until P' = P;
return P
Inclusion Applicability

\[ gr(g, I) = R_2(x, \_), \text{ if } B_2 = \exists R_2; \]
\[ gr(g, I) = R_2(\_, x), \text{ if } B_2 = \exists R_2^-; \]
\[ gr(g, I) = A(x), \text{ if } B_2 = A, \text{ where } A \text{ is a basic concept.} \]
Example

Tbox after normalization:

\[ \text{Professor} \sqsubseteq \exists \text{TeachesTo} \quad \text{Student} \sqsubseteq \exists \text{HasTutor} \]
\[ \exists \text{TeachesTo}^\sim \sqsubseteq \text{Student} \quad \exists \text{HasTutor}^\sim \sqsubseteq \text{Professor} \]
\[ \text{Professor} \sqsubseteq \neg \text{Student} \quad \exists \text{TeachesTo}^\sim \sqsubseteq \neg \text{Professor} \quad \exists \text{HasTutor}^\sim \sqsubseteq \neg \text{Student}. \]

Abox after normalization:

\[ \text{HasTutor}(\text{John}, \text{Mary}) \]
\[ \exists \text{HasTutor}^\sim \text{ (Mary)} \]
\[ \exists \text{HasTutor} \text{ (John)} \]

Query:

\[ q(x) \leftarrow \text{TeachesTo}(x, y), \text{HasTutor}(y, -). \]
Conclusion

• An efficient description logic
• Language is simple but captures main notions of ontologies and conceptual modeling formalisms (ER and UML)
• Focuses on conjunctive queries
• DL-Lite allows
  – Concept subsumption
  – Disjointedness
  – Role-typing
  – Functionality restrictions