On the complexity of querying data through ontologies

Meghyn Bienvenu

CNRS, Université Paris Sud
Querying data through ontologies

Idea: exploit semantic information from ontology when querying data

Example application: querying patient data

- ontology describes medical terms and relationships between terms
  - Hodgkin’s lymphoma is a type of cancer
  - hypertension and high blood pressure are synonyms
- user formulates query using vocabulary of ontology
  - find patients suffering from cancer and high blood pressure
- system performs reasoning to find all (deducible) answers

In this talk:

- quick tour of the field
  - focus on description logic (DL) ontologies
- some pointers to current research
Syntax

Vocabulary
- atomic concepts (unary relations)  
  - Mother, Student
- atomic roles (binary relations) 
  - parentOf, partOf
- individuals (constants) 
  - marie, pierre

Complex concepts
- concept constructors: $\top$, $\neg C$, $C \cap D$, $\exists r.C$, $\geq n r.C$, ...
- examples with translation to FOL:
  - $Person \cap \neg Student$  
    - $Person(x) \land \neg Student(x)$
  - $\exists parentOf.\ Female$  
    - $\exists y.\ parentOf(x, y) \land Female(y)$
  - $\geq 2 parentOf.\ \top$  
    - $\exists y, z.\ parentOf(x, y) \land parentOf(x, z) \land y \neq z$

Complex roles:
- role constructors: $\neg$ (inverse), $\circ$ (composition), ...
Semantics

Interpretation \( \mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}) \)

- \( \Delta^\mathcal{I} \) is a non-empty set (universe)
- \( \cdot^\mathcal{I} \) is a function
  - individual \( a \mapsto \) an element \( a^\mathcal{I} \in \Delta^\mathcal{I} \)
  - atomic concept \( A \mapsto \) a unary relation \( A^\mathcal{I} \subseteq \Delta^\mathcal{I} \)
  - atomic role \( r \mapsto \) a binary relation \( r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \)

Extension to complex concepts and roles:

- \( \top^\mathcal{I} = \Delta^\mathcal{I} \) and \( \bot^\mathcal{I} = \emptyset \)
- \( (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I} \) and \( (C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I} \) and \( (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I} \)
- \( (\exists r.C)^\mathcal{I} = \{u \mid \text{there exists } v \text{ such that } (u, v) \in r^\mathcal{I} \text{ and } v \in C^\mathcal{I} \} \)
- \( (\leq n r.C)^\mathcal{I} = \{u \mid \text{at most } n \text{ } v \text{ such that } (u, v) \in r^\mathcal{I} \text{ and } v \in C^\mathcal{I} \} \)
- \( (r^-)^\mathcal{I} = \{(u, v) \mid (v, u) \in r^\mathcal{I} \} \)
Knowledge bases

DL knowledge base = TBox + ABox

<table>
<thead>
<tr>
<th>TBox (ontology)</th>
<th>Inclusions</th>
<th>Assertions</th>
<th>ABox (data)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( C \subseteq D )</td>
<td>( C^I \subseteq D^I )</td>
<td>( C(a) )</td>
</tr>
<tr>
<td></td>
<td>( R \subseteq S )</td>
<td>( R^I \subseteq S^I )</td>
<td>( r(a, b) )</td>
</tr>
<tr>
<td></td>
<td>( \exists \text{parentOf}. \top )</td>
<td>( (a^I, b^I) \in r^I )</td>
<td>( \text{mother}(\text{marie}) )</td>
</tr>
</tbody>
</table>

\( \mathcal{I} \) is a model of \( \mathcal{K} \) if \( \mathcal{I} \) satisfies all assertions and axioms in \( \mathcal{K} \)
Reasoning tasks

Classical reasoning tasks:

- **subsumption** does $T \models C \subseteq D$ ?
- **classification** find all $A, B$ such that $T \models A \subseteq B$
- **satisfiability** is $\mathcal{K} = (T, \mathcal{A})$ satisfiable ?
- **instance checking** does $(T, \mathcal{A}) \models C(b)$ ?

Relationships among these tasks:

- $T \models C \subseteq D$ iff $(T, \{C(a)\}) \models D(a)$
- $\mathcal{K}$ satisfiable iff $\mathcal{K} \not\models B(a)$ (where $B$ fresh concept, not in $\mathcal{K}$)

Variants: subsumption without TBox, satisfiability of a concept, ...
Short history of DLs

1985-1995  Negative results (undecidability, NP-hardness)
Tractable fragments $(\mathcal{FL}_0, \mathcal{AL})$ based upon $\sqcap$ and $\forall R. C$
Complexity: subsumption in PTIME (but no TBox!)
Algorithms: normalization + structural comparison
Short history of DLs

1985-1995  Negative results (undecidability, NP-hardness)
            Tractable fragments ($\mathcal{FL}_0, \mathcal{AL}$) based upon $\sqcap$ and $\forall R.C$
            Complexity: subsumption in PTIME (but no TBox!)
            Algorithms: normalization + structural comparison

1995-2005  Expressive logics like $\mathcal{SHOIQ}$ which offers:
            $\neg$, $\sqcup$, $\exists R.C$, $\geq n.C$, $\leq n.C$, $r^-$, $r \sqsubseteq s$, (trans $r$), $\{a\}$, ...
            Complexity: subsumption $\geq$ EXPTIME (with TBox)
            Algorithms: highly optimized tableaux reasoners
            Despite very high complexity, good performance!
Short history of DLs

1985-1995  Negative results (undecidability, NP-hardness)
Tractable fragments ($\mathcal{FL}_0, \mathcal{AL}$) based upon $\Box$ and $\forall R.C$
Complexity: subsumption in PTIME (but no TBox!)
Algorithms: normalization + structural comparison

1995-2005  Expressive logics like $\text{SHOIQ}$ which offers:
$\neg, \sqcup, \exists R.C, \geq n.C, \leq n.C, r^-, r \sqsubseteq s,$ (trans $r$), $\{a\},...$
Complexity: subsumption $\geq$ EXPTIME (with TBox)
Algorithms: highly optimized tableaux reasoners
Despite very high complexity, good performance!

2005-present  Lightweight description logics, motivated by applications
$DL-Lite$ family (OWL 2 QL) and $\mathcal{EL}$ family (OWL 2 EL)
Algorithms: query rewriting and/or saturation
Motivations for lightweight DLs

1. **Applications requiring more expressive queries**
   - **conjunctive queries** (like in databases)
     \[ q(x, z) = \text{Female}(x) \land \text{childOf}(x, y) \land \text{Female}(y) \land \text{childOf}(y, z) \land \text{Female}(z) \]
   - difficulty: not reducible to classical reasoning tasks

2. **Applications involving large ontologies and lots of data**
   - scalability is crucial!
Conjunctive queries

Conjunctive queries are an important subclass of first-order logic queries.

They correspond to select-project-join queries in relational DBs.

Formally: a conjunctive query (CQ) has the form

\[ q(x_1, \ldots, x_k) = \exists x_{k+1}, \ldots, x_m \alpha_1 \land \ldots \land \alpha_r \]

where \( \alpha_1, \ldots, \alpha_r \) are atomic formulae over the variables \( x_1, \ldots, x_m \).

**Semantics:** a tuple \((a_1, \ldots, a_k)\) of constants is a (certain) answer to \( q(x_1, \ldots, x_k) \) w.r.t. \( K \) iff \( I \models q[a_1, \ldots, a_k] \) for every model \( I \) of \( K \).
Complexity landscape: expressive DLs

<table>
<thead>
<tr>
<th></th>
<th>instance queries</th>
<th>conjunctive queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>combined</td>
<td>combined</td>
</tr>
<tr>
<td>(\mathcal{ALC}(\mathcal{H}))</td>
<td>EXP-complete</td>
<td>EXP-complete</td>
</tr>
<tr>
<td>(\mathcal{ALCI}, \mathcal{SH}, \mathcal{SHIQ})</td>
<td>EXP-complete</td>
<td>2EXP-complete</td>
</tr>
<tr>
<td>(\mathcal{SHIQ})</td>
<td>NEXP-complete</td>
<td>open</td>
</tr>
<tr>
<td>(\mathcal{SROIQ})</td>
<td>2NEXP-complete</td>
<td>open</td>
</tr>
</tbody>
</table>

\[\mathcal{ALC} : \sqcap, \sqcup, \neg, \exists r. C, \forall r. C \quad \mathcal{I} : r^- \quad \mathcal{H} : r \sqsubseteq s \quad \mathcal{O} : \{a\} \quad \mathcal{S} : (\text{trans } r) \quad \mathcal{R} : r \circ t \sqsubseteq s\]

- **combined complexity**: in terms of the TBox, ABox, and query
- **data complexity**: only in terms of the size of the ABox
  - appropriate when \(|\mathcal{A}| >> |\mathcal{T}|\)
Disjunction yields coNP-hardness

To illustrate the difficulty of answering CQs, we show coNP-hardness in data complexity for DLs with disjunction.

For our reduction, we use the coNP-complete problem 2+2UNSAT:

**Instance** propositional formula $\varphi = c_1 \land \ldots \land c_n$, where each $c_i = v_{i_1} \lor v_{i_2} \lor \neg v_{i_3} \lor \neg v_{i_4}$ (first two literals positive, last two negative), possibly using truth constants true, false

**Problem** decide if the formula $\varphi$ is satisfiable, return “yes” if not satisfiable, “no” if satisfiable
Disjunction yields coNP-hardness

Fixed TBox and query:

- \( \mathcal{T} = \{ V \sqsubseteq T \sqcup F \} \)
- \( q = \exists c, v_1, v_2, v_3, v_4 \ P_1(c, v_1) \land P_2(c, v_2) \land N_1(c, v_3) \land N_2(c, v_4) \land F(v_1) \land F(v_2) \land T(v_3) \land T(v_4) \)

Given a 2+2CNF \( \varphi = c_1 \land \ldots \land c_n \) over \( x_1, \ldots, x_m, \) true, false, we use the following ABox \( \mathcal{A}_\varphi \)

- for each clause \( c_i = v_{i_1} \lor v_{i_2} \lor \neg v_{i_3} \lor \neg v_{i_4} : \)
  \( P_1(c_i, v_{i_1}), P_2(c_i, v_{i_2}), N_1(c, v_{i_3}), N_2(c, v_{i_4}) \)
- for each variable \( x_j : V(x_j) \)
- \( T(\text{true}), F(\text{false}) \)

Can show: \( \mathcal{T}, \mathcal{A}_\varphi \models q_\varphi \) if and only if \( \varphi \) is unsatisfiable
Short history of DLs

1985-1995  Negative results (undecidability, NP-hardness)
  Tractable fragments (e.g. $\mathcal{AL}$) based upon $\sqcap$ and $\forall R. C$
  Complexity: subsumption in PTIME (without TBox)
  Algorithms: normalization + structural comparison

1995-2005  Expressive logics like $\mathcal{SHIQ}$ which offers:
  $\neg, \sqcup, \exists R. C, \geq n. C, \leq n. C, r^-, r \sqsubseteq s$, (trans $r$), ...
  Complexity: subsumption $\geq$ EXPTIME (with TBox)
  Algorithms: highly optimized tableaux reasoners
  Despite very high complexity, good performance!

2005-present Lightweight description logics
  $DL-Lite$ family (OWL 2 QL) and $\mathcal{EL}$ family (OWL 2 EL)
  Algorithms: query rewriting and/or saturation
The *DL-Lite* family

**Objective:** useful ontology language allowing efficient conjunctive query answering

**Idea:** exploit the efficiency of relational DB systems

**General approach:** query rewriting

- ABox is stored as a traditional database
- the input query is rewritten to integrate the relevant information from the TBox
- the new query is evaluated over the database
Syntax of *DL-Lite*

We present the dialect $DL-Lite_{\mathcal{R}}$ (which underlies OWL2 QL).

**Assertions:** $A(c)$, $r(c, d)$

**Inclusions:** $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$, $S_1 \sqsubseteq S_2$, $S_1 \sqsubseteq \neg S_2$ où

$$B := \top \mid A \mid \exists S \quad S := r \mid r^{-}$$

where $A$ is an atomic concept and $r$ an atomic role

Other *DL-Lite* dialects allow:

- functional roles (funct $S$)
- cardinality restrictions ($\geq q S$, $\leq q S$)
- Horn inclusions ($B_1 \sqcap \ldots \sqcap B_n \sqsubseteq (\neg)B_{n+1}$)
- roles which are symmetric, asymmetric, reflexive, or anti-reflexive
First-order rewritability

In DL-Lite, satisfiability and CQ answering are both first-order rewritable:

- given a TBox $\mathcal{T}$, we can compute a first-order query $\varphi_{\mathcal{T}}$ such that for every ABox $\mathcal{A}$, we have:

  $$(\mathcal{T}, \mathcal{A}) \models \bot \quad \text{iff} \quad \mathcal{I}_{\mathcal{A}} \models \varphi_{\mathcal{T}}$$

- given a TBox $\mathcal{T}$ and a CQ $q$, we can compute $q'$ such that for every ABox $\mathcal{A}$ and tuple of constants $\overline{a}$, we have:

  $$\mathcal{T}, \mathcal{A} \models q[\overline{a}] \quad \text{iff} \quad \mathcal{I}_{\mathcal{A}} \models q'[\overline{a}]$$

where $\mathcal{I}_{\mathcal{A}}$ denotes the interpretation based upon $\mathcal{A}$.

Result: both tasks are in $AC^0 \subseteq LOGSPACE \subseteq P$ for data complexity.

- same low data complexity as querying relational databases
Query rewriting by example

\[ T \]
- TeachingStaff ⊆ ∃teaches
- ∃teaches ⊆ TeachingStaff
- Professor ⊆ TeachingStaff
- Lecturer ⊆ TeachingStaff
- Lecturer ⊆ ¬Professor
- ∃teaches ⊆ Course

\[ A \]
- Professor(Sara)
- teaches(Paul, CS100)
- Lecturer(Alex)

\[ Q \]
- TeachingStaff(\(x\))
Query rewriting by example

\[ T \]
- TeachingStaff \sqsubseteq \exists \text{teaches}
- \exists \text{teaches} \sqsubseteq \text{TeachingStaff}
- \text{Professor} \sqsubseteq \text{TeachingStaff}
- \exists \text{teaches} \sqsubseteq \text{Lecturer} \sqsubseteq \neg \text{Professor}
- \exists \text{teaches} \sqsubseteq \text{Course}

\[ A \]
- Professor(Sara)
- teaches(Paul, CS100)
- Lecturer(Alex)

\[ q \]
- TeachingStaff(x)

\[ q' \]
- TeachingStaff(x) \lor \text{Professor}(x) \lor \text{Lecturer}(x) \lor \exists y. \text{teaches}(x, y)
Query rewriting by example

\[ T \]
- TeachingStaff \( \subseteq \exists \) teaches
- \( \exists \) teaches \( \subseteq \) TeachingStaff
- Professor \( \subseteq \) TeachingStaff
- Lecturer \( \subseteq \) TeachingStaff
- \( \exists \) teaches \( \subseteq \) Course

\[ A \]
- Professor(Sara)
- teaches(Paul, CS100)
- Lecturer(Alex)

\[ q \]
- TeachingStaff(x)

\[ q' \]
- TeachingStaff(x) \( \lor \) Professor(x) \( \lor \) Lecturer(x) \( \lor \) \( \exists \) y. teaches(x, y)

Answers: Sara, Paul, Alex
Nowadays, several different query rewriting algorithms exist:

- QuOnto, Requiem, Presto, Rapid, Nyaya, ...

All offer excellent theoretical guarantees (data complexity in $AC^0$)... but suffer from one major problem:

rewritten queries can be huge! \( (O(|T| \cdot |q|)^{|q|}) \)

Database systems handle poorly (if at all) such enormous queries.

**Question**: can this blowup be avoided?
Complexity of query rewriting

For plain DL-Lite (no role inclusions):
- polytime procedure for query rewriting [Kikot et al., DL’11]

For DL-Lite$_R$ (underlying OWL 2 QL):
- no polytime procedure for FO query rewriting (unless P=NP) [Kikot et al., DL’11]
- polynomial NR datalog rewriting possible (under some assumptions), but resulting program complex [Gottlob & Schwentick, DL’11, KR’12]
- analysis of when polynomial FO rewritings are possible [Kikot et al., KR’12]
The $\mathcal{EL}$ family

The logic $\mathcal{EL}$, and its extensions, are designed for applications requiring very large ontologies.

This family of DLs is well-suited for biomedical applications.

Examples of large biomedical ontologies:

- GO (Gene Ontology), around 20,000 concepts
- NCI (cancer ontology), around 30,000 concepts
- SNOMED (medical ontology), over 300,000 concepts (!)

Pericarditis $\sqsubseteq$ Inflammation $\sqcap$ $\exists$loc.Pericardium
Pericardium $\sqsubseteq$ Tissue $\sqcap$ $\exists$partOf.Heart $\sqsubseteq$ Inflammation $\sqsubseteq$ Disease
Disease $\sqsubseteq$ $\exists$loc.$\exists$partOf.Heart $\sqsubseteq$ HeartDisease
Syntax of $\mathcal{EL}$

The basic logic $\mathcal{EL}$ allows complex concepts of the following form:

$$C := \top | C_1 \cap C_2 | \exists R. C$$

Inclusions $C_1 \sqsubseteq C_2$ and assertions $A(c), R(c, d)$

Possible extensions:

- $\bot$ (to express disjoint classes)
- domain restrictions $\text{dom}(R) \sqsubseteq C$
- range restrictions $\text{range}(R) \sqsubseteq C$
- role inclusions $R_1 \circ \ldots \circ R_n \sqsubseteq R_{n+1}$ (transitivity: $R \circ R \sqsubseteq R$)

OWL 2 EL includes all these extensions.
Forward chaining and canonical models

\[ K = \begin{array}{|l|} \hline R(a, b) & C(b) \\ \hline \end{array} \quad \begin{array}{|l|} \hline C \subseteq E & D \subseteq \exists S. (B \cap D) \\ D \subseteq \exists R. (A \cap D) & \exists R. E \subseteq D \\ \hline \end{array} \]
Forward chaining and canonical models

\[ K = \begin{array}{cc}
R(a, b) & C(b) \\
\end{array} \]

\[ C \subseteq E \quad D \subseteq \exists S.(B \land D) \]

\[ D \subseteq \exists R.(A \land D) \quad \exists R.E \subseteq D \]

\[ a \xrightarrow{R} b \]

\[ K = R(a, b) \land C(b) \]

\[ C \subseteq E \quad D \subseteq \exists S.(B \land D) \\
D \subseteq \exists R.(A \land D) \quad \exists R.E \subseteq D \]

\[ a \xrightarrow{R} b \]
Forward chaining and canonical models

\[ \mathcal{K} = \begin{array}{c}
R(a, b) & C(b) \\
\end{array} \]

\[ C \subseteq E \quad D \subseteq \exists S.(B \cap D) \]

\[ \exists R. E \subseteq D \]

\[ D \subseteq \exists R.(A \cap D) \]

Diagram:

- \( a \cdot R \rightarrow b \)
- \( D \)
- \( C \subseteq E \)
- \( E \subseteq D \)
Forward chaining and canonical models

\[ K = \begin{array}{c}
R(a, b) & C(b) \\
\end{array} \]

\[ D \ni \exists R. (A \ni D) \]

\[ D \ni \exists S. (B \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ K = R(a, b) \quad C(b) \]

\[ D \ni \exists R. (A \ni D) \]

\[ D \ni \exists S. (B \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \ quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]

\[ D \quad C \quad E \]

\[ A \quad D \]

\[ R \]

\[ \exists R. (A \ni D) \]

\[ \exists R. E \ni D \]

\[ C \ni E \]

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

\[ a \quad b \]
Forward chaining and canonical models

$$\mathcal{K} = \begin{array}{c}
R(a, b) \quad C(b) \\
D \quad \exists R. \ (A \cap D) \quad \exists R. E \subseteq D
\end{array}$$

$$\begin{array}{c}
C \subseteq E \\
D \subseteq \exists S. (B \cap D)
\end{array}$$
Forward chaining and canonical models

\[ \mathcal{K} = \]

\[ R(a, b) \quad C(b) \]

\[ \exists R. (A \cap D) \]

\[ \exists S. (B \cap D) \]

\[ \exists R. E \subseteq D \]

\[ K = \]

\[ a \quad b \]

\[ A \quad D \]

\[ R \]

\[ S \]

\[ B \quad D \]
Forward chaining and canonical models

\[ \mathcal{K} = \begin{bmatrix} R(a, b) & C(b) \end{bmatrix} \]

\[ C \subseteq E \quad D \subseteq \exists S. (B \land D) \]

\[ D \subseteq \exists R. (A \land D) \quad \exists R.E \subseteq D \]
Forward chaining and canonical models

\[ \mathcal{K} = \begin{align*}
R(a, b) & \quad C(b) \\
D & \quad \exists R. (A \sqsubseteq D) \\
D & \quad \exists S. (B \sqsubseteq D) \\
E & \quad \exists R. (D \sqsubseteq E)
\end{align*} \]

Exhaustive application of inclusions

Result: canonical model \( \mathcal{I}_\mathcal{K} \)

- always gives the right answer to queries
\[ \mathcal{I}_\mathcal{K} \models q(\tilde{a}) \text{ iff } \mathcal{K} \models q(\tilde{a}) \]

- may be infinite

- forest structure (ABox + attached trees)
Compact representation of the canonical model

$\mathcal{K} = \begin{array}{c|c}
R(a, b) & C(b) \\
\hline
C \subseteq E & D \subseteq \exists S. (B \cap D) \\
D \subseteq \exists R. (A \cap D) & \exists R. E \subseteq D
\end{array}$

Idea: use the repetitions in $\mathcal{I}_\mathcal{K}$ to find a finite representation
Compact representation of the canonical model

$\mathcal{K} =$

$R(a, b)$ $C(b)$

$C \sqsubseteq E$ $D \sqsubseteq \exists S.(B \cap D)$

$D \sqsubseteq \exists R.(A \cap D)$ $\exists R.E \sqsubseteq D$

normalized TBox:
- only atomic concepts behind $\exists$
- conjunction only on the left-hand-side

$D \sqsubseteq \exists R.(A \cap D) \leadsto D \sqsubseteq \exists R.F \quad F \sqsubseteq A \quad F \sqsubseteq D \quad A \cap D \sqsubseteq F$

$D \sqsubseteq \exists S.(B \cap D) \leadsto D \sqsubseteq \exists S.G \quad G \sqsubseteq B \quad G \sqsubseteq D \quad B \cap D \sqsubseteq G$
Compact representation of the canonical model

\[ \mathcal{K} = \begin{array}{c}
R(a, b) \quad C(b)
\end{array} \]

\[ C \subseteq E \quad D \subseteq \exists S.G \quad G \subseteq B \quad G \subseteq D \]
\[ B \cap D \subseteq G \quad D \subseteq \exists R.F \quad A \cap D \subseteq F \]
\[ F \subseteq A \quad F \subseteq D \quad \exists R.E \subseteq D \]

At the start:
- ABox assertions
- an individual \( w_A \) with \( A(w_A) \) for each atomic concept \( A \)

Application of an inclusion on \( x \):
- if \( C(x) \) and \( C \subseteq A \): add \( A(x) \)
- if \( C(x) \) and \( C \subseteq \exists R.A \): add \( R(x, w_A) \)
- if \( C(x), D(x) \) and \( C \cap D \subseteq A \): add \( A(x) \)
Compact representation of the canonical model

\[ \mathcal{K} = \begin{array}{cc}
R(a, b) & C(b) \\
\end{array} \]

Result: \( C_{\mathcal{K}} \)

- Subsumption
  \[ \mathcal{K} \models A_1 \subseteq A_2 \quad \text{ssi} \quad C_{\mathcal{K}} \models A_2(w_{A_1}) \]

- Instance queries
  \[ \mathcal{K} \models A_1(c) \quad \text{ssi} \quad C_{\mathcal{K}} \models A_1(c) \]

Terminates in polynomial time
Compact representation of the canonical model

\[ \mathcal{K} = \begin{array}{c} R(a, b) \quad C(b) \end{array} \]

\[ \begin{array}{c} C \subset E & D \subset \exists S.G & G \subset B & G \subset D \\ B \cap D \subset G & D \subset \exists R.F & A \cap D \subset F \\ F \subset A & F \subset D & \exists R.E \subset D \end{array} \]

Result: \( C_\mathcal{K} \)

- **Subsumption**
  \( \mathcal{K} \models A_1 \sqsubseteq A_2 \text{ ssi } C_\mathcal{K} \models A_2(w_{A_1}) \)

- **Instance queries**
  \( \mathcal{K} \models A_1(c) \text{ ssi } C_\mathcal{K} \models A_1(c) \)

Terminates in polynomial time

can classify SNOMED in a few seconds!
Compact representation of the canonical model

\[ \mathcal{K} = R(a, b) \land C(b) \]

Result: \( C_{\mathcal{K}} \)

- **Subsumption**
  \( \mathcal{K} \models A_1 \sqsubseteq A_2 \) \iff \( C_{\mathcal{K}} \models A_2(w_{A_1}) \)

- **Instance queries**
  \( \mathcal{K} \models A_1(c) \) \iff \( C_{\mathcal{K}} \models A_1(c) \)

Terminates in polynomial time

What about conjunctive queries?
Answering conjunctive queries

$\mathcal{K} = \{ R(a, b), C(b) \}$

$C \subseteq E \quad D \subseteq \exists S.G \quad G \subseteq B \quad G \subseteq D$

$B \cap D \subseteq G \quad D \subseteq \exists R.F \quad A \cap D \subseteq F$

$F \subseteq A \quad F \subseteq D \quad \exists R.E \subseteq D$

$\mathcal{C}_\mathcal{K}$

$\mathcal{I}_\mathcal{K}$

$q = \exists x R(x, x)$

answer: yes

answer: no
Answering conjunctive queries

\[ \mathcal{K} = \begin{array}{c}
R(a, b) \\
C(b)
\end{array} \]

\[ C \subseteq E \quad D \subseteq \exists S.G \quad G \subseteq B \quad G \subseteq D \]
\[ B \cap D \subseteq G \quad D \subseteq \exists R.F \quad A \cap D \subseteq F \]
\[ F \subseteq A \quad F \subseteq D \quad \exists R.E \subseteq D \]

\[ \mathcal{C}_K \]
\[ \mathcal{I}_K \]

answers: \((a, b), (a, w_F), (w_F, w_F), (w_G, w_F), (w_D, w_F)\)

\[ q(x, y) = D(x) \land R(x, y) \]

\[ \neq \]

answers: \((a, b)\)
Answering conjunctive queries

**Problem:** false positives - query matches in $C_K$ that do not exist in $I_K$

**Solution:** modify $q$ to prevent such matches

For our examples:

$$\exists x \, R(x, x) \leadsto \exists x \, R(x, x) \land \bigwedge_A (x \neq w_A)$$

$$D(x) \land R(x, y) \leadsto D(x) \land R(x, y) \land \bigwedge_A (x \neq w_A \land y \neq w_A)$$

**Remark:** rewriting of $q$ is independent of both $T$ and $A$
The approach we have just seen is called “combined rewriting”.

This approach guarantees polynomial data complexity.

**Advantage**: more widely applicable than “pure” rewriting

**Disadvantage**: uses more space (if $|\mathcal{A}|$ is big...), modifies the data

**Note**: combined rewriting also interesting for DL-Lite
First-order rewritability in \( \mathcal{EL} \)

Combined approach requires ability to modify the data
  - not always possible / desirable! (e.g. information integration)

**Question**: can we identify queries which are FO-rewritable?

Some first results in this direction [Bienvenu et al., DL’12] for IQs:
  - always possible if TBox is acyclic
  - for general TBoxes: the problem of deciding FO-rewritability is PSPACE-hard, in \( \text{EXPTIME} \)
  - \( \text{EXPTIME} \)-hard if ABoxes have restricted signature

Non-uniform complexity analysis: consider specific TBox, query (see [Lutz and Wolter, KR’12] for more on this).
Expressive “lightweight” DLs

Interestingly, much more expressive DLs have polynomial data complexity.

Horn-\textit{SHIQ}: extends both DL-Lite and \textit{EL}

- classical reasoning is \textit{EXPTIME}-complete in combined complexity (like for full \textit{SHIQ})
- conjunctive query answering is \textit{P}-complete in data complexity (like for \textit{EL})

New querying algorithm for Horn-\textit{SHIQ} [Eiter et. al., AAAI’12] based upon datalog:

- can be seen as rewriting the query using the TBox, then evaluating it over completed ABox
Recap of complexity landscape

<table>
<thead>
<tr>
<th></th>
<th>instance queries</th>
<th>conjunctive queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>combined</td>
<td>combined</td>
</tr>
<tr>
<td>Plain database</td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>DL-Lite</td>
<td>in P</td>
<td>NP-complete</td>
</tr>
<tr>
<td>EL</td>
<td>P-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td>ELI, Horn-SH(O)IQ</td>
<td>EXP-complete</td>
<td>EXP-complete</td>
</tr>
<tr>
<td>Horn-SR(O)IQ</td>
<td>2EXP-complete</td>
<td>2EXP-complete</td>
</tr>
<tr>
<td>ALC((H))</td>
<td>EXP-complete</td>
<td>EXP-complete</td>
</tr>
<tr>
<td>ALCI / SH / SHIQ</td>
<td>EXP-complete</td>
<td>2EXP-complete</td>
</tr>
<tr>
<td>SHOIQ</td>
<td>NEXP-complete</td>
<td>open</td>
</tr>
<tr>
<td>SROIQ (OWL 2)</td>
<td>2NEXP-complete</td>
<td>open</td>
</tr>
</tbody>
</table>
Conclusion

Research in DLs has undergone big changes in recent years:

- new application: using ontologies to access data
- conjunctive query answering now a central reasoning task
- focus on new families of tractable DLs (DL-Lite, $\mathcal{EL}$)

Nowadays, complexity landscape quite well understood

- two measures: combined complexity and data complexity
- landscape for CQs more nuanced than for traditional reasoning tasks

Two main techniques used for lightweight DLs:

- query rewriting
- saturation (aka forward-chaining, chase)
Current work and future directions

Remains a lot of do in order to make query answering really practicable:

- **more refined complexity analysis** (beyond data complexity)

- **database-style optimizations**

- **benchmarks** for testing algorithms sorely lacking!

- **what about more expressive query languages?**
  - regular path queries [Bienvenu et al., DL’12], CQs extended with negation or inequalities (cf. [Rosati ’07])

- **querying inconsistent data**, cf. [Rosati, IJCAI’11] [Bienvenu, AAAI’12]
Textbook


However, most of what was covered in the talk isn't in this book.
References: DL-Lite family


References: DL-Lite family


  [Note: for Datalog +/-]
References: DL-Lite family


References: $\mathcal{EL}$ family


- Jing Mei, Shengping Liu, Guo Tong Xie, Aditya Kalyanpur, Achille Fokoue, Yuan Ni, Hanyu Li, Yue Pan: *A Practical Approach for Scalable Conjunctive Query Answering on Acyclic $\mathcal{EL}^+$ Knowledge Base*. Proceedings of ISWC (2009).
References: Horn-$SHI\mathcal{Q}$


References: Expressive DLs


Miscellaneous references

Non-uniform complexity of query answering:


Semantic indexing:


Query containment and minimization in DLs:

Path queries:


Negative results for richer query languages:


Inconsistency-tolerant query answering:

