Camilo Thorne, Diego Calvanese

KRDB Centre
Free University of Bozen-Bolzano
Via della Mostra 4
39100 - Italy
Ontology-based systems [Staab&Studer 2004] aim at accessing and querying (possibly from the web) repositories of heterogeneous data.

Examples: data integration systems, knowledge portals, ontology-based systems for semantically annotated data, etc. [Staab&Studer]

We denote such systems as **ontology-based data access systems** (OBDASs) [Calvanese et al. 2005]

In such scenarios, an ontology layer on top of a data layer provides a global **conceptual model** of potentially incomplete sources over which formal queries (SQL, SPARQL, etc.) are formulated.

The semantics of such systems can be characterized in terms of **FO interpretations**

Querying takes place under the **open world assumption** (OWA)

---

**The Problem**

---

Exploring Controlled English OBDA (2)
Ontology Languages

We will focus on ontologies represented in fragments of the W3C ontology language OWL.

Significant fragments of OWL correspond to widely used conceptual modelling formalisms such as UML class diagrams.
We will focus on ontologies represented in fragments of the W3C ontology language OWL.

Significant fragments of OWL correspond to widely used conceptual modelling formalisms such as UML class diagrams.

The **Employee** ontology characterizes the domain of employees, specifying

(i) the classes, relations and attributes (= the terminology) into which the domain is structured

(ii) the constraints (IS-A, participation, cardinality) all (incomplete) sources satisfy
To improve the usability of interfaces to ontologies and OBDASs, controlled languages [Bernstein et al. 2005, Sowa 2004] have been proposed.

They have been shown to outperform (in such terms) interfaces based on keywords or visual query languages [Bernstein et al. 2007].

They provide a trade-off between the rigor of formal ontology/query languages and NL.

This is related to work on NLIs to databases [Androstopoulos 1995] and CL interfaces to databases [Wintner et al. 2006].
To improve the usability of interfaces to ontologies and OBDASs, **controlled languages** [Bernstein et al. 2005, Sowa 2004] have been proposed. They have been shown to outperform (in such terms) interfaces based on keywords or visual query languages [Bernstein et al. 2007]. They provide a trade-off between the rigor of formal ontology/query languages and NL. This is related to work on NLIs to databases [Androstopoulos 1995] and CL interfaces to databases [Wintner et al. 2006].

Declarations translate **compositionally** into ontologies and questions into formal queries. Their semantic complexity [Pratt 2003] reduces to the computational properties of the OBDASs. ⇒ We should study the **computational complexity** of CLs w.r.t. OBDA.
Ontology Languages

**Semantic Web Language**
(OWL)

```xml
<owl:Class rdf:about="#Employee">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#develops"/>
      <owl:someValuesFrom rdf:resource="#Project"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Every employee develops some project

OWL is a **machine-readable** language (embedded in RDF and XML)
CLs are **human-readable**, yet as unambiguous as DLs

**Description Logics**

+ **CL**

```xml
Employee ⊑ ∃develops:Project
```

Exploring Controlled English OBDA (5)
Outline

1. The Problem
   (i) Ontology languages
   (ii) Controlled Languages

1. OBDA and Query Answering
   (i) ALCI ontologies and conjunctive queries
   (ii) Certain answers and query answering
   (iii) DL-Lite ontologies

2. Controlled Languages
   (i) DL-English and Lite-English
   (ii) The \{IS-A_i\}_{i\in[0,7]} fragments

3. Computational Complexity
   (i) Expressing query answering
   (ii) Tree-shaped conjunctive queries
   (iii) Data complexity of QA

4. Conclusions and further work
In \textit{ALCI}, \textbf{roles} \( R \) and \textbf{concepts} \( C \) are formed according to the syntax

\[
R \rightarrow P \mid P^-
\]
\[
C \rightarrow T \mid A \mid \exists R: C \mid \neg C \mid C \cap C'
\]
**ALCI Ontologies**

In **ALCI**, roles $R$ and concepts $C$ are formed according to the syntax

$$
R \rightarrow P \mid P^-
$$

$$
C \rightarrow T \mid A \mid \exists R : C \mid \neg C \mid C \cap C'
$$

An **assertion** is an expression $C \sqsubseteq C'$

A **terminology** (TBox) $\mathcal{T}$ is a set of assertions

An **ontology** is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A}$ is a set of ground facts (ABox)
**ALCI Ontologies**

In **ALCI**, **roles** $R$ and **concepts** $C$ are formed according to the syntax

$$
R \rightarrow P | P^- \\
C \rightarrow T | A | \exists R:C | \neg C | C \cap C'
$$

An **assertion** is an expression $C \sqsubseteq C'$

A **terminology** (TBox) $\mathcal{T}$ is a set of assertions

An **ontology** is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A}$ is a set of ground facts (ABox)

Semantics is given by FO **interpretations** $\mathcal{D} := \langle \Delta, .^D \rangle$

$$
\begin{align*}
A^D & \subseteq \Delta \\
T^D & := \Delta \\
(\exists R:C)^D & := \{d \mid \text{exists } d' \text{ s.t. } \langle d, d' \rangle \in R^D \text{ and } d' \in C^D \} \\
(\neg C)^D & := \Delta - C^D \\
(C \cap C')^D & := C^D \cap C'^D \\
P^D & \subseteq \Delta \times \Delta \\
(R^-)^D & := \{\langle d, d' \rangle \mid \langle d', d \rangle \in R^D \}
\end{align*}
$$

$\mathcal{D} \models C \sqsubseteq C'$ iff $C^D \subseteq C'^D$

$\mathcal{D} \models \langle \mathcal{T}, \mathcal{A} \rangle$ iff

i. $\mathcal{D} \models \mathcal{T}$

ii. $\mathcal{D} \models \mathcal{A}$

$\text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle) := \{\mathcal{D} \mid \mathcal{D} \models \langle \mathcal{T}, \mathcal{A} \rangle\}$
A **conjunctive query** (CQ) is a query of the form

\[ q(\vec{x}) \leftarrow \exists \vec{y} \Phi(\vec{x}, \vec{y}) \]

where \( q(\vec{x}) \) is the **head**, \( \vec{x} \) is a sequence of \( n \) **distinguished variables** and \( \exists \vec{y} \Phi(\vec{x}, \vec{y}) \) is a conjunction of existentially quantified atoms called **body**.
A **conjunctive query** (CQ) is a query of the form

\[ q(\vec{x}) \leftarrow \exists \vec{y} \Phi(\vec{x}, \vec{y}) \]

where \( q(\vec{x}) \) is the **head**, \( \vec{x} \) is a sequence of \( n \) **distinguished variables** and \( \exists \vec{y} \Phi(\vec{x}, \vec{y}) \) is a conjunction of existentially quantified atoms called **body**

They correspond to SQL SELECT-PROJECT-JOIN queries
A **conjunctive query** (CQ) is a query of the form

\[ q(\vec{x}) \leftarrow \exists \vec{y} \Phi(\vec{x}, \vec{y}) \]

where \( q(\vec{x}) \) is the **head**, \( \vec{x} \) is a sequence of \( n \) **distinguished variables** and \( \exists \vec{y} \Phi(\vec{x}, \vec{y}) \) is a conjunction of existentially quantified atoms called **body**

They correspond to SQL SELECT-PROJECT-JOIN queries

**EXAMPLE:**

Which manager is a project manager that manages some project?

\[ q(x) \leftarrow Manager(x) \land ProjectManager(x) \land \exists y (manages(x, y) \land Project(y)) \]

**SELECT** Manager.MName  
FROM Manager, ProjectManager, manages, Project  
WHERE Manager.MName = ProjectManager.MName  
AND Manager.MName = manages.MName  
AND Project.PName = manages.PName
**Certain Answers Semantics**

In OBDASs, CQs are formulated over the atomic concepts and roles of the ontology. The **certain answers** of a CQ $q$ over an ontology $\langle T, A \rangle$ are:

$$\text{cert}(q, A, T) := \{ \vec{d} \mid \langle T, A \rangle \models q(\vec{d}) \}$$

NB: It is essentially a FO entailment problem!

⇒ asking $q$ to an ontology = asking $q$ to all the models of the ontology
Certain Answers Semantics

In OBDASs, CQs are formulated over the atomic concepts and roles of the ontology. The **certain answers** of a CQ $q$ over an ontology $\langle T, A \rangle$ are:

$$\text{cert}(q, A, T) := \{ \vec{d} \mid \langle T, A \rangle \models q(\vec{d}) \}$$

**NB:** It is essentially a FO **entailment** problem!

$\Rightarrow$ asking $q$ to an ontology $= \text{ asking } q$ to **all** the models of the ontology

Inspired by [Vardi 1982] we consider different computational complexity measures:

- if $A$ is the only input $\Rightarrow$ **data complexity**
- if $q$ is the only input $\Rightarrow$ query complexity
- if $T$ is the only input $\Rightarrow$ schema complexity
- if both $q$ and $\langle T, A \rangle$ are inputs $\Rightarrow$ combined complexity
Certain Answers Semantics

In OBDASs, CQs are formulated over the atomic concepts and roles of the ontology.

The **certain answers** of a CQ $q$ over an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ are:

$$\text{cert}(q, \mathcal{A}, \mathcal{T}) := \{ \bar{d} \mid \langle \mathcal{T}, \mathcal{A} \rangle \models q(\bar{d}) \}$$

NB: It is essentially a FO **entailment** problem!

$\Rightarrow$ asking $q$ to an ontology $\Rightarrow$ asking $q$ to **all** the models of the ontology.

Inspired by [Vardi 1982] we consider different computational complexity measures:

- if $\mathcal{A}$ is the only input $\Rightarrow$ **data complexity**
- if $q$ is the only input $\Rightarrow$ query complexity
- if $\mathcal{T}$ is the only input $\Rightarrow$ schema complexity
- if both $q$ and $\langle \mathcal{T}, \mathcal{A} \rangle$ are inputs $\Rightarrow$ combined complexity

NB: The **query answering problem** (QA) is the associated decision problem.

$\Rightarrow$ by restricting (or expanding) the expressivity of $\mathcal{T}$, we obtain different computational properties.
**DL-Lite Ontologies**

A fragment of $\mathcal{ALCI}$ optimized for data access in OBDASs is $\textit{DL-Lite}$. In $\textit{DL-Lite}$ concepts are partitioned into \textbf{right} and \textbf{left} concepts:

\begin{align*}
R & \rightarrow P \mid P^- \\
C_l & \rightarrow A \mid \exists R : \top \\
C_r & \rightarrow C_l \mid \neg C_l \mid C_r \cap C_r' \mid \exists R : C_r
\end{align*}

Assertions (in TBoxes) are now of the form $C_l \sqsubseteq C_r$

QA (w.r.t. CQs) is optimal $\Rightarrow$ \textit{LogSpace} in data complexity

QA for $\mathcal{ALCI}$ is intractable $\Rightarrow$ \textit{coNP}-complete in data complexity

$\Rightarrow$ \textit{DL-Lite scales to data}!
**DL-Lite Ontologies**

*DL-Lite* captures the main features of conceptual data models (UML class diagrams, ER-diagrams, etc.)

**NB:** in *DL-Lite* we cannot capture completeness of the hierarchy
We want to express in CL ontology languages and queries. CLs allow for a compositional semantics by which they map into some logic formalism.

Compositionality motivates us to consider their semantic complexity [Pratt & Third 2005]. Semantic complexity is defined as the reasoning problems associated to their logic formalisms.

In the particular setting of OBDAS, this amounts to considering the different reasoning problems relevant for ontologies. We are particularly interested in the query answering problem: how difficult is it to access data from an ontology with CL? does this task scale to data?
Expressing \textit{ALCI} ontologies with DL-English

Following DL conventions [Baader et al. 2004] we associate

- word categories \textbf{N}, \textbf{Adj} and \textbf{IV} to atomic concepts
- category \textbf{TV} to role names
- recursive constituents to arbitrary concepts

\[
\begin{aligned}
\text{Every} & : \lambda C. \lambda C'. C \subseteq C' \\
\text{Nom} & : C \\
\text{VP} & : C' \\
\text{Everybody who} & : \lambda C. \lambda C'. C \subseteq C' \\
\text{VP} & : C \\
\text{VP} & : C'
\end{aligned}
\]

No manager who manages some project that does not make some money is shrewd.

\[
\begin{aligned}
\text{Manager} \sqcap \exists \text{manages} : (\text{Project} \sqcap \neg (\exists \text{make} : \text{Money})) \sqsubseteq \neg \text{Shrewd}
\end{aligned}
\]

Nobody manages only projects

\[
\begin{aligned}
\forall \text{manages} : \text{Project} \sqsubseteq \bot
\end{aligned}
\]

Anybody who manages some project manages some big project or small project

\[
\begin{aligned}
\exists \text{manages} : \text{Project} \sqsubseteq \exists \text{manages} : ((\text{Project} \sqcap \text{Big}) \sqcup ((\text{Project} \sqcap \text{Small})
\end{aligned}
\]

All DL-English (complete) sentences translate into an \textit{ALCI} assertion and conversely
Expressing \textit{ALCI} ontologies with DL-English

\[
\begin{align*}
S & \rightarrow \text{NP} \text{ VP} & \text{VP} & \rightarrow \text{TV} \text{ NP} & \text{VP} & \rightarrow \text{is a} \text{ Nom} & \text{VP} & \rightarrow \text{is TV by} \text{ NP} & \text{NP} & \rightarrow \text{Det Nom} & \\
\text{VP} & \rightarrow \text{is Adj} & \text{VP} & \rightarrow \text{IV} & \text{VP} & \rightarrow \text{is Neg TV by} \text{ NP} & \text{NP} & \rightarrow \text{Pro Relp VP} & \text{NP} & \rightarrow \text{Pro} \\
\text{VP} & \rightarrow \text{does Neg IV} & \text{VP} & \rightarrow \text{is Neg a Nom} & \text{Nom} & \rightarrow \text{Nom Relp VP} & \text{Nom} & \rightarrow \text{Adj Nom} & \text{Nom} & \rightarrow \text{N} \\
\text{VP} & \rightarrow \text{is Neg Adj} & \text{VP} & \rightarrow \text{VP} \text{ Crd VP} & \text{Nom} & \rightarrow \text{Nom Crd VP} & \text{Nom} & \rightarrow \text{N} \\
\end{align*}
\]

\[
\begin{align*}
\tau(\text{VP}) & := \tau(\text{NP})(\tau(\text{TV})) & \tau(\text{VP}) & := \tau(\text{Crd})(\tau(\text{VP}))(\tau(\text{VP})) & \tau(\text{S}) & := \tau(\text{NP})(\tau(\text{VP})) \\
\tau(\text{NP}) & := \tau(\text{Neg})(\tau(\text{NP})(\tau(\text{TV}))) & \tau(\text{VP}) & := \tau(\text{Neg})(\tau(\text{Adj})) & \\
\tau(\text{NP}) & := \tau(\text{Det})(\tau(\text{Nom})) & \tau(\text{VP}) & := \tau(\text{Neg})(\tau(\text{IV})) & \\
\tau(\text{Nom}) & := \tau(\text{Nom})(\tau(\text{Relp})(\tau(\text{VP}))) & \tau(\text{NP}) & := \tau(\text{Pro})(\tau(\text{Relp})(\tau(\text{VP}))) & \\
\tau(\text{Nom}) & := \tau(\text{Adj})(\tau(\text{Nom})) & \tau(\text{Nom}) & := \tau(\text{N}) & \\
\end{align*}
\]

\[
\begin{align*}
\text{Pro} & \rightarrow \text{anybody} & \tau(\text{Pro}) & := \lambda C. \lambda C'. C \sqsubseteq C': (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t) \\
\text{Pro} & \rightarrow \text{somebody} & \tau(\text{Pro}) & := \lambda R. \exists R: (e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t) \\
\text{Pro} & \rightarrow \text{nobody} & \tau(\text{Pro}) & := \lambda C. \lambda C'. C \sqsubseteq \neg C': (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t) \\
\text{Pro} & \rightarrow \text{only} & \tau(\text{Pro}) & := \lambda C. \lambda R. \forall R: C: (e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)) \\
\text{Pro} & \rightarrow \text{everybody} & \tau(\text{Pro}) & := \lambda C. \top \sqsubseteq C: (e \rightarrow t) \rightarrow t \\
\text{Pro} & \rightarrow \text{nobody} & \tau(\text{Pro}) & := \lambda C. C \sqsubseteq \bot: (e \rightarrow t) \rightarrow t \\
\text{Det} & \rightarrow \text{some} & \tau(\text{Det}) & := \lambda C. \lambda R. \exists R: C: (e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)) \\
\text{Det} & \rightarrow \text{every} & \tau(\text{Det}) & := \lambda C. \lambda C'. C \sqsubseteq C': (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t) \\
\text{Det} & \rightarrow \text{no} & \tau(\text{Det}) & := \lambda C. \lambda C'. C \sqsubseteq \neg C': (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t) \\
\end{align*}
\]
Expressing $ALCI$ ontologies with DL-English

A successful derivation for the VP "loves only men"
$\Rightarrow$ types unify
Failed derivation for the VP "loves every man"
⇒ types $e \to (e \to t)$ and $e \to t$ do not unify.
Expressing *DL-Lite* ontologies with Lite-English

In Lite-English, DL-English **Noms** and **VPs** are *constrained* to match left (= subject **Noms**) and right concepts (= predicate **VPs**)

The only negation allowed is introduced by "no"

* DL-Lite is expressed by Lite-English [Bernardi et al. 2007]
# Related Declarative CLs

<table>
<thead>
<tr>
<th>CL (English)</th>
<th>Maps to</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE [Fuchs 2005]</td>
<td>FO</td>
<td>KR/User specifications</td>
</tr>
<tr>
<td>ACE-OWL [Kaaljurand 2007]</td>
<td>OWL-DL</td>
<td>Ontology authoring + querying</td>
</tr>
<tr>
<td>PENG [Schwitter 2003]</td>
<td>OWL-DL</td>
<td>Ontology authoring + querying</td>
</tr>
<tr>
<td>SOS [Schwitter 2008]</td>
<td>OWL-DL</td>
<td>Ontology authoring + querying</td>
</tr>
<tr>
<td>CLCE [Sowa 2004]</td>
<td>FOL</td>
<td>Knowledge representation</td>
</tr>
<tr>
<td>AECMA [Unwalla 2005]</td>
<td>no</td>
<td>User specifications</td>
</tr>
<tr>
<td>English Query (EQ) [Blum 1999]</td>
<td>SQL</td>
<td>DB querying/management</td>
</tr>
<tr>
<td>OWL-CNL [Schwitter 2006]</td>
<td>OWL-DL</td>
<td>Ontology authoring</td>
</tr>
<tr>
<td>Easy English [Bernth 1998]</td>
<td>no</td>
<td>User specifications</td>
</tr>
<tr>
<td>λ-SQL [Winter 2006]</td>
<td>SQL</td>
<td>DB querying</td>
</tr>
<tr>
<td>nRQL [Schwitter 2008]</td>
<td>FO queries</td>
<td>Ontology querying</td>
</tr>
<tr>
<td>Rabbit [Schwitter 2008]</td>
<td>OWL</td>
<td>Ontology authoring</td>
</tr>
<tr>
<td>ACE-PQL [Bernstein 2005]</td>
<td>PQL</td>
<td>Ontology querying</td>
</tr>
<tr>
<td>QE-III [Clifford 1987]</td>
<td>IL</td>
<td>DB querying</td>
</tr>
</tbody>
</table>

(an overview of some controlled fragments of English)
A compositional translation $\tau(\cdot)$ maps a fragment of NL into a fragment of logic $\Rightarrow$ FO + the $\lambda$-abstraction, $\lambda$-application, types and $\beta$-reduction of higher order logic (HOL) [Montague 1970]

Such logic expressions are known as meaning representations (MRs)

Modulo $\tau(\cdot)$ we can speak about the semantic complexity of a fragment of English [Pratt 2003]
Expressing QA

A **compositional translation** $\tau(\cdot)$ maps a fragment of NL into a fragment of logic $\Rightarrow$ FO + the $\lambda$-abstraction, $\lambda$-application, types and $\beta$-reduction of higher order logic (HOL) [Montague 1970]

Such logic expressions are known as **meaning representations** (MRs)

Modulo $\tau(\cdot)$ we can speak about the **semantic complexity** of a fragment of English [Pratt 2003]

Let $\mathcal{L}$ be an ontology language, $\mathcal{Q}$ a query language, to **express QA in controlled English**

(i) define a grammar $G_\mathcal{L}$ with $\tau(\cdot)$ s.t. $\tau(L(G_\mathcal{L})) = \mathcal{L}$

(ii) define a grammar $G_\mathcal{Q}$ with $\tau'(\cdot)$ s.t. $\tau'(L(G_\mathcal{Q})) = \mathcal{Q}$

Such ontology/query language expressions become the meaning representations (MRs) of the CL utterances
A CQ that expresses an $ALCI$ concept is called a tree-shaped conjunctive query (TCQ)

To express them in CL we use, as function words,

- the determiner "some" and the pronouns "something, somebody" (existential)
- relative pronouns and $VP$-coordination (conjunction)
- interrogative pronouns such as "which, what, who," (etc.)
Expressing QA

A CQ that expresses an $\mathcal{ALCI}$ concept is called a **tree-shaped conjunctive query** (TCQ)

To express them in CL we use, as function words,
- the determiner ”some” and the pronouns ”something, somebody” (existential)
- relative pronouns and **VP**-coordination (conjunction)
- interrogative pronouns such as ”which, what, who,” (etc.)

**EXAMPLE:**

Which manager is a project manager that manages some project that is developed by some employee?

\[
q(x) \leftarrow \text{Manager}(x) \land \text{ProjectManager}(x) \land \exists y (\text{manages}(x,y) \land \text{Project}(y) \\
\exists z (\text{develops}(z,y) \land \text{Employee}(z) \land \text{Project}(y)))
\]

\[
\lambda x^e. \text{Manager}(x) \land \text{ProjectManager}(x) \land \exists y (\text{manages}(x,y) \land \text{Project}(y) \\
\exists z (\text{develops}(z,y) \land \text{Employee}(z) \land \text{Project}(y)) : e \rightarrow t
\]
The Family \{IS-A_i\}_{i \in [0,7]} of CLs

We are interested in refining our analysis regarding ontology languages. We want to single out

- the **maximal** CLs that are tractable w.r.t. data complexity
- the **minimal** CLs that are intractable w.r.t. data complexity
The Family \{IS-A_i\}_{i\in[0,7]} of CLs

We are interested in refining our analysis regarding ontology languages. We want to single out:

- the \textbf{maximal} CLs that are tractable w.r.t. data complexity
- the \textbf{minimal} CLs that are intractable w.r.t. data complexity

We adopt as strategy \textbf{restricting} on DL-English. Utterances in each fragment translate into assertions $C_i \subseteq C_r$. Hence, we partition [Bernardi et al. 2007] \textbf{Nom} and \textbf{VP} into:

- \textbf{left} components: $\textbf{Nom}_l$, $\textbf{VP}_l$
- \textbf{right} components: $\textbf{Nom}_r$, $\textbf{VP}_r$

\Rightarrow this allows for a fine-grained data complexity analysis.
### The Family \(\{IS-A_i\}_{i \in [0,7]}\) of CLs

<table>
<thead>
<tr>
<th>Fragment</th>
<th>Assertions</th>
<th>Sample Sentence(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS-A_0</td>
<td>(A \subseteq A_1 \sqcap \cdots \sqcap A_n)</td>
<td>(\Rightarrow) Every project manager is a manager and is an employee.</td>
</tr>
<tr>
<td>IS-A_1</td>
<td>(A \subseteq \forall P: A')</td>
<td>(\Rightarrow) Every project manager manages only projects.</td>
</tr>
<tr>
<td>IS-A_2</td>
<td>(A_1 \sqcap \cdots \sqcap A_n \subseteq \forall P: (A_1 \sqcap \cdots \sqcap A_m))</td>
<td>(\Rightarrow) Every good manager manages only good projects.</td>
</tr>
<tr>
<td>IS-A_3</td>
<td>(\exists P: A \subseteq A_1 \sqcap \cdots \sqcap A_n)</td>
<td>(\Rightarrow) Anybody who manages some project is an employee and is a manager.</td>
</tr>
<tr>
<td></td>
<td>(\exists P^-: A \subseteq A_1 \sqcap \cdots \sqcap A_n)</td>
<td>(\Rightarrow) Anything that is managed by some important manager is a big project.</td>
</tr>
<tr>
<td></td>
<td>(A \subseteq \exists P)</td>
<td>(\Rightarrow) Every manager manages something.</td>
</tr>
<tr>
<td>IS-A_4</td>
<td>(A_1 \sqcap \cdots \sqcap A_n \subseteq A_1 \sqcap \cdots \sqcap A_m)</td>
<td>(\Rightarrow) Every cruel manager is a bad manager.</td>
</tr>
<tr>
<td></td>
<td>(\exists P: (A_1 \sqcap \cdots \sqcap A_n) \subseteq A_1 \sqcap \cdots \sqcap A_m)</td>
<td>(\Rightarrow) Anybody who manages some bankrupt project is a bad manager.</td>
</tr>
<tr>
<td>IS-A_5</td>
<td>(\forall P: A \subseteq A_1 \sqcap \cdots \sqcap A_n)</td>
<td>(\Rightarrow) Anybody who manages only projects is a manager and a project manager.</td>
</tr>
<tr>
<td>IS-A_6</td>
<td>(A \subseteq A_1 \sqcup \cdots \sqcup A_n)</td>
<td>(\Rightarrow) Every manager is a project manager or is an area manager.</td>
</tr>
<tr>
<td>IS-A_7</td>
<td>(\neg A \subseteq A_1 \sqcap \cdots \sqcap A_n)</td>
<td>(\Rightarrow) Anybody who is not an area manager is an employee who is a project manager.</td>
</tr>
</tbody>
</table>
The Family \{\{S-A_i\}i\in[0,7]\} of CLs

<table>
<thead>
<tr>
<th>Concept $C_f$</th>
<th>Constituent $\alpha_f$</th>
<th>Grammar Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
<tr>
<td>$\exists P: A$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
<tr>
<td>$\forall P: A$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
<tr>
<td>$\exists P$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
<tr>
<td>$A_1 \sqcap \cdots \sqcap A_n$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
<tr>
<td>$A_1 \sqcup \cdots \sqcup A_n$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
<tr>
<td>$\neg A$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
<td>$\lambda C_i. \lambda C_r. C_i \subseteq C_r$</td>
</tr>
</tbody>
</table>

Exploring Controlled English OBDA (23)
## Complexity (w.r.t. TCQs)

<table>
<thead>
<tr>
<th>IS-A⁰</th>
<th>SAT (KB)</th>
<th>QA (data)</th>
<th>QA (combined)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>in PTime</td>
<td>in LogSpace</td>
<td>in PTime</td>
</tr>
<tr>
<td>IS-A¹</td>
<td>in PTime</td>
<td>NLogSpace-complete</td>
<td>in PSpace</td>
</tr>
<tr>
<td>IS-A²</td>
<td>in PTime</td>
<td>PTime-complete</td>
<td>in PSpace</td>
</tr>
<tr>
<td>IS-A³</td>
<td>in PTime</td>
<td>PTime-complete</td>
<td>in NExpTime (*)</td>
</tr>
<tr>
<td>IS-A⁴</td>
<td>in PTime</td>
<td>PTime-complete</td>
<td>in PSpace</td>
</tr>
<tr>
<td>IS-A⁵</td>
<td>in PTime</td>
<td>coNP-complete</td>
<td>in NExpTime (*)</td>
</tr>
<tr>
<td>IS-A⁶</td>
<td>in PTime</td>
<td>coNP-complete</td>
<td>coNP-complete</td>
</tr>
<tr>
<td>IS-A⁷</td>
<td>in PTime</td>
<td>coNP-complete</td>
<td>coNP-complete</td>
</tr>
</tbody>
</table>

Only the first four exhibit **tractable** data complexity [Lutz & Krisnadhi 2007, Rosati 2007, Krötsh & Rudolph 2007]

Intractability is caused by our being able to express the **partitioning** of a domain [Calvanese et al. 2006, Ortiz et al. 2008]
Complexity (w.r.t. TCQs)

<table>
<thead>
<tr>
<th>SAT (KB)</th>
<th>QA (data)</th>
<th>QA (combined)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS-A₀</td>
<td>in PTime</td>
<td>in LogSpace</td>
</tr>
<tr>
<td>IS-A₁</td>
<td>in PTime</td>
<td>NLogSpace-complete</td>
</tr>
<tr>
<td>IS-A₂</td>
<td>in PTime</td>
<td>PTime-complete</td>
</tr>
<tr>
<td>IS-A₃</td>
<td>in PTime</td>
<td>PTime-complete</td>
</tr>
<tr>
<td>IS-A₄</td>
<td>in PTime</td>
<td>PTime-complete</td>
</tr>
<tr>
<td>IS-A₅</td>
<td>in PTime</td>
<td>coNP-complete</td>
</tr>
<tr>
<td>IS-A₆</td>
<td>in PTime</td>
<td>coNP-complete</td>
</tr>
<tr>
<td>IS-A₇</td>
<td>in PTime</td>
<td>coNP-complete</td>
</tr>
</tbody>
</table>

Only the first four exhibit **tractable** data complexity [Lutz & Krisnadhi 2007, Rosati 2007, Krötsh & Rudolph 2007]

Intractability is caused by our being able to express the **partitioning** of a domain [Calvanese et al. 2006, Ortiz et al. 2008]

NB: A **maximal tractable** CL w.r.t. data complexity is obtained by **eliminating negation** from DL-English

⇒ we express the DL $\mathcal{ELI}$ \quad $C \rightarrow T \mid A \mid \exists R:C \mid C \sqcap C'$

⇒ medical ontologies (e.g. GALEN) express mostly $\mathcal{ELI}$ assertions
**Complexity (w.r.t. TCQs)**

<table>
<thead>
<tr>
<th></th>
<th>SAT (KB)</th>
<th>QA (data)</th>
<th>QA (combined)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL-Lite</td>
<td>in PTime</td>
<td>in LogSpace</td>
<td>in PSpace (*)</td>
</tr>
<tr>
<td>ALCI</td>
<td>ExpTime-complete</td>
<td>coNP-complete</td>
<td>ExpTime-complete</td>
</tr>
<tr>
<td>ALCQI</td>
<td>ExpTime-complete</td>
<td>coNP-complete</td>
<td>ExpTime-complete</td>
</tr>
<tr>
<td>SHIF</td>
<td>ExpTime-complete</td>
<td>coNP-complete</td>
<td>ExpTime-complete</td>
</tr>
<tr>
<td>SHOIN</td>
<td>NExpTime-complete</td>
<td>coNP-hard</td>
<td>NExpTime-complete</td>
</tr>
<tr>
<td>SHROIQ</td>
<td>NExpTime-hard</td>
<td>coNP-hard</td>
<td>NExpTime-hard</td>
</tr>
</tbody>
</table>

[Baader et al. 2004, Calvanese et al, 2005]

*DL-Lite* = Lite-English

*ALCI* = DL-English

*SHIF[D]* = ACE-OWL-Lite = OWL-Lite

*SHOIN[D]* = ACE-OWL-DL = OWL-DL

*SHROIQ[D]* = ACE-OWL = OWL 1.1.
Conclusions and further work

We have argued in favor of analysing the data complexity of CLs. This measure is relevant in the context of accessing information with CLs in ontology-based systems. To do so, we have proposed to express in CL QA over ontologies.

By considering the spectrum of CLs lying between $\mathcal{ALC}I$ and $DL-Lite$, the $\{\text{IS-A}_i\}_{i \in [0,7]}$ fragments, we can see

- which fragments are maximal (w.r.t. tractability) and minimal (w.r.t.) intractability
- how each NL construct contributes to computational properties

$\Rightarrow$ a path that remains to be explored is to consider more expressive interrogative CLs.

- adding full negation, anaphora and comparatives may yield intractability of QA (over $DL-Lite$ ontologies)
- SQL aggregation functions does not (over $DL-Lite$ ontologies)